INEX
Initiative for the Evaluation of XML retrieval

INEX 2005 Workshop Pre-Proceedings

November 28-30, 2005
Schloss Dagstuhl
International Conference and Research
Center for Computer Science

http://inex.is.informatik.uni-duisburg.de/2005/

DELOS
Network of Excellence on Digital Libraries

Norbert Fuhr
Mounia Lalmas
Saadia Malik
Gabriella Kazai

Table of Contents

Methodology

Multiple tracks

Ad-hoc track

Relevance feedback track

Natural language query track

Heterogeneous track

Interactive track

Document mining track

Multimedia track

APPENDIX

Ad-hoc track

Heterogeneous track

Multimedia track

Interactive track

Document mining track

Relevance feedback track

Natural language query track

Organisers

Project leaders

Norbert Fuhr (University of Duisburg-Essen)
Mounia Lalmas (Queen Mary University of London)

Contact persons

Saadia Malik (University of Duisburg-Essen)
Zoltan Szlavik (Queen Mary University of London)

Topic format specification

Börkur Sigurbjörnsson (University of Amsterdam)
Andrew Trotman (University of Otago)

Online relevance assessment tool

Benjamin Piwowarski (Universitad de Chile)

Metrics

Gabriella Kazai (Queen Mary University of London)
Arjen P. de Vries (CWI)
Paul Ogilvie (Carnegie Mellon University)
Benjamin Piwowarski (Universitad de Chile)

Relevance feedback task

Yosi Mass (IBM Research Lab)
Carolyn Crouch (University of Minnesota Duluth)

Natural language processing task

Shlomo Geva (Queensland University of Technology)
Alan Woodley (Queensland University of Technology)

Heterogeneous collection track

Ray Larson (University of California, Berkeley)

Interactive track

Birger Larsen (Royal School of Library and Information Science)
Anastasios Tombros (Queen Mary University of London)
Saadia Malik (University of Duisburg-Essen)

Document mining track

Ludovic Denoyer (Université Paris 6)
Anne-Marie Vercoustre (Inria-Rocquencourt)
Patrick Gallinari (Université Paris 6)

XML multimedia track

Roelof van Zwol (Utrecht University)
Gabriella Kazai (Queen Mary University of London)
Mounia Lalmas (Queen Mary University of London)

Preface

Welcome to the 4th workshop of the Initiative for the Evaluation of XML Retrieval (INEX)!

Now, in its fourth year, INEX is an established evaluation forum for XML information retrieval (IR), with over 50 participating organisations worldwide. Its aim is to provide an infrastructure, in the form of a large XML test collection and appropriate scoring methods, for the evaluation of XML IR systems.

XML IR plays an increasingly important role in many information access systems (e.g. digital libraries, web, intranet) where content is more and more a mixture of text, multimedia, and metadata, formatted according to the adopted W3C standard for information repositories, the so-called eXtensible Markup Language (XML). The ultimate goal of such systems is to provide the right content to their end-users. However, while many of today's information access systems still treat documents as single large (text) blocks, XML offers the opportunity to exploit the internal structure of documents in order to allow for more precise access, thus providing more specific answers to user requests. Providing effective access to XML-based content is therefore a key issue for the success of these systems.

2005 was an exciting year for INEX, and brought with it a lot of changes and new aspects to the evaluation. In total seven research tracks were included in INEX 2005, which studied different aspects of XML information access: Ad-hoc, Interactive, Multimedia, Relevance Feedback, Heterogeneous, Document Mining and Natural Language (NLP). The Multimedia and Document Mining tracks were new for the 2005 campaign; the other tracks reached their second year. The interactive track expanded in the numbers of tasks offered and in the number of participating groups; the track tries to answer some fundamental questions of XML IR. The heterogeneous track expanded by studying new collections with different DTDs and their effect on XML IR system effectiveness. The relevance feedback track investigated approaches for queries that also include structural hints (rather than content-only queries in 2004). The NLP track included a new task in 2005 that allows new participants with NLP expertise to join the INEX workshop without the need to develop a search engine, and thus encouraging wider accessibility. The consolidation of the existing tracks, and the expansion to new areas offered by the two new tracks, allows INEX to grow in reach.

INEX 2005 has also introduced a new relevance assessment procedure and new evaluation metrics.

The aim of the INEX 2005 workshop is to bring together researchers in the field of XML IR who participated in the INEX 2005 evaluation campaign. During the past year participating organisations contributed to the building of a large-scale XML test collection by creating topics, performing retrieval runs and providing relevance assessments. The workshop concludes the results of this large-scale effort, summarises and addresses encountered issues and devises a work plan for the future evaluation of XML retrieval systems.

Acknowledgements

INEX is funded by the DELOS Network of Excellence on Digital Libraries, to which we are very thankful. We would also like to thank the IEEE Computer Society for providing us the XML document collection.

We gratefully thank organisers of the various tracks for their great work in setting up the new tracks, and carrying on and refining the existing tracks.

As always, special thanks go to the participating organisations and people for their contributions and hard work throughout the year! Also, many thanks to everyone who has contributed to the lively email discussions on the various mailing lists – resulting in a record number of emails, and a reputation of the INEX community as one of the most vivacious.

We hope you have enjoyed the INEX 2005 campaign and have fruitful and stimulating discussions at the workshop.

Norbert Fuhr, University of Duisburg-Essen
Mounia Lalmas, Queen Mary University of London
Saadia Malik, University of Duisburg-Essen
Gabriella Kazai, Queen Mary University of London
November 2005

Schloss Dagstuhl

Schloss Dagstuhl or Dagstuhl manor house was built in 1760 by the then reigning prince Count Anton von Öttingen-Soetern-Hohenbaldern. After the French Revolution and occupation by the French in 1794, Dagstuhl was temporarily in the possession of a Lorraine ironworks.

In 1806 the manor house along with the accompanying lands was purchased by the French Baron Wilhelm de Lasalle von Louisenthal.

In 1959 the House of Lasalle von Louisenthal died out, at which time the manor house was then taken over by an order of Franciscan nuns, who set up an old-age home there.

In 1989 the Saarland government purchased the manor house for the purpose of setting up the International Conference and Research Center for Computer Science.

The first seminar in Dagstuhl took place in August of 1990. Every year approximately 2,000 research scientists from all over the world attend the 30-35 Dagstuhl Seminars and an equal number of other events hosted at the center.

http://www.dagstuhl.de/

EPRUM metrics and INEX 2005
DRAFT

Benjamin Piwowarski

Centre for Web Research, Universidad de Chile
bpiwowar@dcc.uchile.cl

Abstract. Standard Information Retrieval (IR) metrics are not well suited for new paradigms like XML or Web IR in which retrievable information units are document elements or sets of related document. These units are neither predefined nor independent, and the elements returned by IR systems may overlap and contain near misses. Part of the problem stems from the classical hypothesis on the user behaviour that do not take into account the structural or logical context of document elements or the possibility of navigation between units. This paper proposes a more realistic user model which encompasses a large variety of user behaviours, makes explicit the hypothesis underlying the user on explicit formal grounds. Based on this user model, we propose an extension of the probabilistic precision-recall metric which allows coping with the different problems encountered with these new IR paradigms. In this paper, we present the EPRUM metric used for evaluating the official submissions of INEX 2005. We also discuss the implication of such a metric on several key problems of XML Information Retrieval: the notion of the ideal list, the problem of the overlap.

1 Introduction

This document describes the EPRUM metric in the context of XML Retrieval. EPRUM is a metric that aims at providing a unique and comprehensive framework for the evaluation of XML Retrieval systems[1], by defining a precise user model and an extension of the notion of precision at a given recall level. As Generalised Recall [3] and Precision-Recall with User Modelling [4], EPRUM is based on a probabilistic model of the user and of the relevance that are directly used while computing precision and recall. This user model has parameters that can be tuned so that they mimic the "average" user behaviour.

The EPRUM user and relevance model also has consequences (1) on the interpretation of the INEX scale and (2) on the definition of what is an ideal run and its relation with the user model. With respect to the latter we define precisely what is user satisfaction and what is its relationship with the INEX scale. With respect to the former, diverging from our initial algorithm [3], we follow the one described in [1].

A note about relevance: we distinguish between the relevance of an element (the element contains some relevant material) and the idealism of an element (the fact

[1] but not limited to: the relevance model could be used in standard information retrieval and its user model could be reused in passage retrieval, web retrieval, video retrieval, etc.

that the element is the unit the user wants to see, i.e. that it belongs to the ideal recall base). In order to compute the ideal set, we used the algorithm described by G. Kazai in [1].

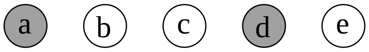
2 The EPRUM metric

EPRUM is an extension of precision-recall. Precision is defined *as the ratio of the minimum number of ranks that a user has to consult in the list returned by an ideal system and by the evaluated system,* given that the user wants to see a given amount of ideal units. At a given recall level l ($0 < l \leq 1$), precision is defined formally as:

$$\text{Precision}(l) = \mathbb{E}\left[\begin{array}{c}\text{Achievement indicator}\\ \text{for a recall } l\end{array} \times \frac{\begin{array}{c}\text{Minimum number of consulted list items}\\ \text{for achieving a recall } l \text{ (over all lists)}\end{array}}{\begin{array}{c}\text{Minimum number of consulted list items}\\ \text{for achieving a recall } l \text{ (system list)}\end{array}}\right]$$

It is easy to see that this is just an alternative definition of the precision at a given recall level. In classical IR, if a system retrieves $A+B$ documents, where A is the number of relevant documents and B the number of not relevant documents, then an ideal system would achieve the same recall with a list reduced to A documents. The above definition would result in a precision $\frac{A}{A+B}$ which is the exact definition of precision – the ratio of the number of relevant documents to the number of retrieved documents. The achievement indicator is used to set the precision to 0 if the recall level cannot be achieved; this is also the classical definition of precision-recall.

The following example illustrates the definition of the EPRUM metric; let the list returned by a system be the following:



where gray nodes are ideal units while white nodes are not ideal. The standard definition of precision would assign a precision of respectively 1, 0.25 and 0 for recalls of 1, 2 and 3 (or more). With the definition we chose, we get the same values (just forget about the mathematical expectation for now!):

Recall 1 The minimum number of elements the user has to consult, over all possible lists, is 1. The value is the same for the system list and the user was able to see one element. Precision is 1.

Recall 2 The minimum number of elements the user has to consult, over all possible lists, is 2. For the evaluated system, the user will have to consult the list until d - that is, the minimum number of items that she has to consult is 4. Precision is 0.5.

Recall 3 In this case, the same process would give us a precision of 3/5 (because the user has to consult the whole list), but has the recall cannot be attained by the user, the achievement indicator is 0 and hence precision is also 0.

As shown in this example, this definition of precision-recall gives the same results as the standard definition. The interest of this formulation is that we can define and use more complex user and relevance models, and starting from the same definition, derive a generalisation of precision-recall. It is possible to proof that, using the final formula of EPRUM and settings its parameters so as to mimic the standard user behaviour in "flat" IR, we get exactly the same result as trec_eval.

3 What is needed to compute EPRUM?

EPRUM can be computed given three different sets of parameters:

1. The probability that a user *consults* an element of the corpus. In standard IR, we say that a user consults a document when she clicks on the link in the list returned by the system. This probability reflects the fact that a user will have to click from a result in the list returned by a the IR system and will eventually have look at the element(s) that are associated with the list item. In the context of XML Retrieval, we have to distinguish two cases: the Fetch&Browse task and the others. In the case of the Focussed task, we suppose that a user will *always* consider an element after having clicked on its surrogate in the list. In the case of the Fetch&Browse task, the user model is more complex and is described latter.
2. The probability that a user *browses* from a considered element to any neighbour element. That is, a user, when considering an element, will most probably look around to its close context (i.e., in an XML documents this would be the previous siblings, next siblings, ancestors, etc.). This behaviour is stochastic, that is defined by a probability, since we don't expect **all** the users to behave the same. The probability of browsing from a considered element x to an element y could be measured, in a user experiment, by the proportion of users that would see y after having considered x.
3. The probability that a user finds a unit ideal. This probability is closely related to the concept of quantisation but has a well defined meaning in EPRUM: In a user experiment, its value would be the proportion of users that would find the given unit ideal if they had exactly the same information need.

Unfortunately, we still don't have enough user data to compute even an approximated user model. Nevertheless it is possible to define simple yet realistic behaviours. In INEX 2005, we chose user models close to the ones implied by xCG (where only elements overlapping with an ideal unit can be rewarded) and defined the following user models:

1. For the consideration,
   Focussed In the focussed task, we made the hypothesis, like for standard IR, that a user always considers elements pointed by list item. That is, if the third list item is element-x then the user will consult the element-x (she will see the content of this element). The probability of considering an element for the focussed task is either 0 (the element in the not in the i first ranks) or 1 (the element is in the first i list elements):

$$\mathrm{P}\left(C_{i,x'}\right) = \begin{cases} 1 & \text{if } x' \text{ is within the first } i \text{ elements of the list} \\ 0 & \text{otherwise} \end{cases}$$

Fetch&Browse Here, an item in the list is not anymore only one element but a set of elements from the same document. We view this task as follows: The user clicks on the document in the list. She is presented a document where system selected elements are highlighted and ordered – imagine that there is button that can focus the user window on each selected element in turn. The user then begins to see the first ranked element, then the second, etc. for a given article.

We make the hypothesis that the probability that the user keeps on consulting the list of elements depends on the amount of irrelevant material contained in the previous consulted elements – this is somehow similar to the T2I (tolerance to irrelevance) user model [2]. That is, the probability that the user keeps on going after having consulted an element in the document directly depends on the element overlap with the ideal elements. For an element ranked i within a document group, the probability that the user considers it is defined as:

$$\mathrm{P}\left(C_{i,x_j}\right) = \mathrm{P}\left(C_{i,x_{j-1}}\right) \times \left(k + (1-k) \times \frac{\text{size of intersection with ideal elements}}{\text{size of the element}}\right)$$

where $\mathrm{P}\left(C_{i,x_{-1}}\right) = 1$ by definition.

The coefficient k is the minimum probability for a user to consider the next element in the list. For INEX 2005, we set k to 0.8. For example, if the three first elements, say of size 10 characters, returned for an article have no intersection with an ideal element, the probability that the user considers the second one is $0.8 + 0.2 \times 0 = 0.8$, that she considers the third one is $0.8 \times (0.8 + 0.2 \times 0) = 0.64$, etc. Note that a run that returns only elements within (or equal to) ideal targets have their probability of considering an element always equal to 1. In this case (and only in this case), the order among elements within the article doesn't change the performance of the system with respect to this instantiation of EPRUM parameters.

2. For the browsing or navigational behaviour, we chose a simple user model – the user, from a considered element, can go up or down. We call this behaviour "hierarchic": The proportion of users that navigate from an element to one of its descendant, or from an element of its ancestor, is equal to the ratio of the sizes of the elements:

$$\mathrm{P}(x \longrightarrow y) = \begin{cases} \frac{\text{size of } y}{\text{size of } x} & \text{if } y \text{ is an ancestor of } x \\ \frac{\text{size of } y}{\text{size of } x} & \text{if } x \text{ is an ancestor of } y \\ 0 & \text{otherwise} \end{cases}$$

For instance, 30 % of the users would go from a section of size 10 its enclosing paragraph of size 3. Note that more realistic user models, like the T2I one, could be used. We chose this simple model because submitted runs were optimised for the inex_eval or the XCG metrics which have an implicit user definition which is close to the hierarchic behaviour.

3. For the idealism of an element, we used the Exh quantisation

$$\mathrm{P}(x \text{ is ideal}) = \begin{cases} 0 & \text{if "too small"} \\ 0 & \text{if exhaustivity is 0} \\ 0.5 & \text{if exhaustivity is 1} \\ 1 & \text{if exhaustivity is 2} \end{cases}$$

   Note that when an element has a probability 0 of being ideal, it does not mean that a system returning this element will not be rewarded because a user can still browse to the ideal element. Note also that no “too small” element can be an ideal unit. We interpret the probability that an element is ideal as the percentage of users that would be satisfied by the element.

4 Examples

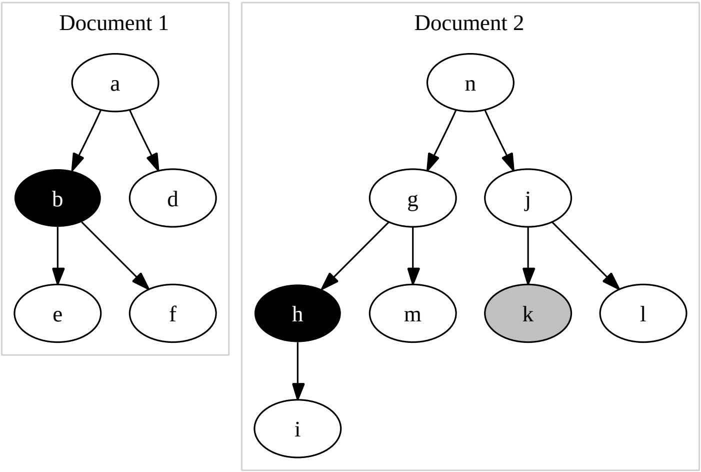


Fig. 1. The example database, composed of two documents and twelve elements. Two elements are highly exhaustive (b and h, with a black background) for the query and one of them is fairly exhaustive (k). The size of each element is 1 + the size of its children (in an imaginary unit: this could be for instance the number of words divided by 100): the size of e (f, h, m, k or l) is 1, the size of b is 3, the size of a is 5, etc. The probability of navigating from an element to the other being the ratio of sizes, the probability to navigate from f to b is for instance $\frac{1}{3}$.

We present in this section the evaluation of four lists for the Focussed (and SVCAS, VVCAS) and Fetch&Browse tasks. We used a small database where only two (or three) elements are ideal, as illustrated in Fig. 1.

The precision can be rewritten, for a given recall value r (r is the number of ideal units the user wants to see):

$$\begin{aligned}\text{Precision}(r) = \mathbb{E}&\left[\begin{array}{c}\text{Minimum number of consulted list items}\\ \text{for achieving a recall } l \text{ (over all lists)}\end{array}\right]\Bigg\}(E1)\\ &\times\,\mathbb{E}\left[\frac{\begin{array}{c}\text{Achievement indicator}\\ \text{for a recall } l\end{array}}{\begin{array}{c}\text{Minimum number of consulted list items}\\ \text{for achieving a recall } l \text{ (system list)}\end{array}}\right]\Bigg\}(E2)\end{aligned}$$

It can be shown that:

$$(E1) = \sum_{\text{rank } i} i\left(\mathrm{P}(F_i^* \geq r) - \mathrm{P}(F_{i-1}^* \geq r)\right)$$

$$(E2) = \sum_{\text{rank } i} \frac{1}{r}\left(\mathrm{P}(F_i \geq r) - \mathrm{P}(F_{i-1} \geq r)\right)$$

where F_i (resp. F_i^*) is the number of ideal elements found by the user after she consulted the i first ranks of the system returned list (resp. the ideal list). If we consider the classical case, where an ideal element is or not retrieved at each rank, then $\mathrm{P}(F_i \geq r)$ is either 0 or 1. In this case, it is easy to see that the expected value E1 (resp. E2) is the actual value (or inverse value) of the rank where the r^{th} ideal element has been retrieved.

4.1 Focussed and VVCAS, SVCAS

We use the following lists:

A List b,h,k: This is the ideal list, composed of the ideal elements - with the most ideal first.

B List k, h, b: This is the ideal list, but ordered by increasing order of relevance

C List f, h, k: The list **A** but with b (first element) replaced by one of its child

D List h, f, k: The list **D**, swapping the two first elements

We assume that the probability that the user has seen more than one ideal element *before* beginning to consult the list is 0; that is, $\mathrm{P}(F_0 \geq r) = 0$ for $r > 0$. We then distinguish two cases:

1. If the user is satisfied with an element of exhaustivity at least 2 (75 % of the users – there is a justification that we don't present here):

   Recall 1 (level 1/2) $(E1)$ is 1; $(E2)$ is resp. 1, $\frac{1}{2}$, $1 \times (\frac{1}{3} - 0) + \frac{1}{2} \times (1 - \frac{1}{3}) = \frac{2}{3}$, and 1 for lists **A**, **B**, **C** and **D**. Precision is 1, $\frac{1}{2}$, $\frac{2}{3}$, and 1.

   Recall 2 (level 1) $(E1)$ is 2; $(E2)$ is resp. $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{2} \times (\frac{1}{3} - 0) = \frac{1}{6}$, and $\frac{1}{6}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, $\frac{2}{3}$, $\frac{1}{3}$ and $\frac{1}{3}$.

2. If the user is satisfied with an element of exhaustivity at least 1 (25 % of the users):

Table 1. Evolution of the different probabilities, with respect to the different lists (A, B, C, D) for the Focussed, VVCAS, and SVCAS tasks. The three columns below probability P$(S_{i,x})$ correspond respectively to the probability that element a, b, or c is seen by the user after rank i. The probability P$_2$ (resp. P$_1$) is the probability that the user found at least ... ideal elements after rank i, given that she only is satisfied with elements at least highly (resp. fairly) exhaustive.

	List (b,h,k): **A**								List (k,h,b): **B**							
	P$(S_{i,x})$			P$_2(F_i \geq)$		P$_1(F_i \geq)$			P$(S_{i,x})$			P$_2(F_i \geq)$		P$_2(F_i \geq)$		
rank	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**
1	1	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0
2	1	1	0	1	1	1	1	0	0	1	1	1	0	1	1	0
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	List (f,h,k) **C**								List (b,i,k): **D**							
	P$(S_{i,x})$			P$_2(F_i \geq)$		P$_1(F_i \geq)$			P$(S_{i,x})$			P$_2(F_i \geq)$		P$_1(F_i \geq)$		
rank	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**
1	$\frac{1}{3}$	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	1	0	0	1	0	1	0	0
2	$\frac{1}{3}$	1	0	1	$\frac{1}{3}$	1	$\frac{1}{3}$	0	1	$\frac{1}{3}$	0	1	$\frac{1}{3}$	1	$\frac{1}{3}$	0
3	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$

Recall 1 (level 1/3) $(E1)$ is 1; $(E2)$ is resp. 1, 1, $1 \times (\frac{1}{3} - 0) + \frac{1}{2} \times (1 - \frac{1}{3}) = \frac{2}{3}$, and 1 for lists **A**, **B**, **C** and **D**. Precision is 1, 1, $\frac{2}{3}$, and 1.

Recall 2 (level 2/3) $(E1)$ is 2; $(E2)$ is resp. $\frac{1}{2}$, $\frac{1}{2}$, $\frac{1}{2} \times (\frac{1}{3} - 0) + \frac{1}{3} \times (1 - \frac{1}{3}) = \frac{7}{18}$, and $\frac{7}{18}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, 1, $\frac{7}{9}$ and $\frac{7}{9}$.

Recall 3 (level 1) $(E1)$ is 3; $(E2)$ is resp. $\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{3} \times (\frac{1}{3} - 0) = \frac{1}{9}$, and $\frac{1}{9}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, 1, $\frac{1}{3}$ and $\frac{1}{3}$.

There is a way to combine the two sets of precisions that we do not describe here but give an example instead. If a user wants to see more than two third of the ideal elements for list **B**, then for 75 % of the users this means a precision of $\frac{2}{3}$ and for 25 % of them this means 1: Hence the precision of $.\dot{7}5 \times \frac{5}{6} + .25 \times 1 = .875$. For the same list, if a user wants to see between $\frac{1}{2}$ (excluded) and $\frac{2}{3}$ (included) of the ideal elements, then for 75 % of the users that means seeing 2 ideal elements with a precision $\frac{2}{3}$, and for 25 % of the users that means seeing 2 ideal elements with a precision 1. Hence, a precision of $.75 \times \frac{2}{3} + .25 \times 1 = .75$.

The evaluations order the runs in an order which is appropriate: **A** has the maximum score and **C** is worse than **D** (**D** and **C** have their two first list item swapped, and **D** has a fully ideal element as its top ranked element). **B**, containing only ideal elements, is overall better than **C** and **D**.

4.2 Fetch & Browse

We use the following lists:

A List D2[h,k] D1[b]: this is the ideal list

Table 2. Precision-recall for the Focussed, VVCAS, and SVCAS tasks. The precision for the four lists and four recall intervals are shown. The line "correspondence" show what are the number of ideal elements the user wants to see if (1) she considers that only elements with an exhaustivity 2 are ideal (2) she considers that elements with an exhaustivity at least 1 are ideal.

recall level	$]0,\frac{1}{3}]$	$]\frac{1}{3},\frac{1}{2}]$	$]\frac{1}{2},\frac{2}{3}]$	$]\frac{2}{3},1]$
correspondence	1,1	1,2	2,2	2,3
A	1	1	1	1
B	0.63	0.63	.75	0.88
C	0.67	0.69	0.44	0.33
D	1	0.94	0.44	0.33

B List D1[b] D2[h,k]: the ideal list in reverse order

C List D2[i,k] D1[b]: the first document returned contains a near miss (i); as i is fully specific (contained in an ideal element), the probability that the user consults the next element (k) is 1.

D List D2[g,k] D1[b]: in this list, the first element of the first returned document is an element that overlaps partially with an ideal element; hence, the user will *consider* the element k of D2 with a probability inferior to 1. Said otherwise, not all the user will continue to consult the highlighted elements within D2. The actual probability that the user consults the element k is $0.8+0.2\times\frac{1}{2}=0.9$ as one half of g overlaps with h: At the first rank the user sees h with a probability .5, and k with a probability .9. Another thing to note, is that if h and k satisfy the user, then she sees at first rank at least one ideal element with a probability $\frac{1}{2}\times.9+\frac{1}{2}\times.1+\frac{1}{2}\times.9=\frac{1.9}{2}$, the three terms of the sum being the case where (1) the user sees h and k, (2) the user sees k but not h, and (3) the user sees k but not h.

Note also that the only real difference between lists C and D is that the first element is not fully specific (because the same proportion of users will browse to h from element g than from element i).

Like in the previous Section, we distinguish two cases:

1. If the user is satisfied with an element of exhaustivity at least 2 (75 % of the users – there is a justification that we don't present here):
   Recall 1 (level 1/2) $(E1)$ is 1; $(E2)$ is resp. 1, 1, $1\times(\frac{1}{2}-0)+\frac{1}{2}\times(1-\frac{1}{2})=\frac{3}{4}$, and $\frac{3}{4}$ for lists **A**, **B**, **C** and **D**. Precision is 1, 1, $\frac{3}{4}$, and $\frac{3}{4}$.
   Recall 2 (level 1) $(E1)$ is 2; $(E2)$ is resp. $\frac{1}{2}$, $\frac{1}{2}$, $\frac{1}{2}\times(\frac{1}{2}-0)=\frac{1}{4}$, and $\frac{1}{4}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, 1, $\frac{1}{2}$ and $\frac{1}{2}$.
2. If the user is satisfied with an element of exhaustivity at least 1 (25 % of the users):
   Recall 1 (level 1/3) $(E1)$ is 1; $(E2)$ is resp. 1, 1, 1, and $1\times(\frac{1.9}{2}-0)+\frac{1}{2}\times(1-\frac{1.9}{2})=\frac{3.9}{4}$ for lists **A**, **B**, **C** and **D**. Precision is 1, 1, 1, and $\frac{3.9}{4}$.
   Recall 2 (level 2/3) $(E1)$ is 1; $(E2)$ is resp. 1, $\frac{1}{2}$, $1\times(\frac{1}{2}-0)+\frac{1}{2}\times(1-\frac{1}{2})=\frac{3}{4}$, and $1\times(\frac{.9}{2}-0)+\frac{1}{2}\times(\frac{1.9}{2}-\frac{.9}{2})=\frac{2.8}{4}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, $\frac{1}{2}$, $\frac{3}{4}$ and $\frac{2.8}{4}$.

Table 3. Evolution of the different probabilities, with respect to the different lists (A, B, C, D) for the Fetch and Browse tasks. The three columns below probability $\mathrm{P}(S_{i,x})$ correspond respectively to the probability that element a, b, or c is seen by the user after rank *i*. The probability P_2 (resp. P_1) is the probability that the user found at least ... ideal elements after rank *i*, given that she only is satisfied with elements at least highly (resp. fairly) exhaustive.

	List D2[h,k] D1[b]: A								List D1[b] D2[h,k]: B							
	$\mathrm{P}(S_{i,x})$			$\mathrm{P}_2(F_i \geq)$		$\mathrm{P}_1(F_i \geq)$			$\mathrm{P}(S_{i,x})$			$\mathrm{P}_2(F_i \geq)$		$\mathrm{P}_2(F_i \geq)$		
rank	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**
1	0	1	1	1	0	1	1	0	1	0	0	1	0	1	0	0
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	List D2[i,k] D1[b]: C								List D2[g,k] D1[f]: D							
	$\mathrm{P}(S_{i,x})$			$\mathrm{P}_2(F_i \geq)$		$\mathrm{P}_1(F_i \geq)$			$\mathrm{P}(S_{i,x})$			$\mathrm{P}_2(F_i \geq)$		$\mathrm{P}_1(F_i \geq)$		
rank	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**
1	0	$\frac{1}{2}$	1	$\frac{1}{2}$	0	1	$\frac{1}{2}$	0	0	$\frac{1}{2}$	.9	$\frac{1}{2}$	0	$\frac{1.9}{2}$	$\frac{.9}{2}$	0
2	1	$\frac{1}{2}$	1	1	$\frac{1}{2}$	1	1	$\frac{1}{2}$	1	$\frac{1}{2}$	.9	1	$\frac{1}{2}$	1	$\frac{1.9}{2}$	$\frac{.9}{2}$

Recall 3 (level 1) $(E1)$ is 2; $(E2)$ is resp. $\frac{1}{2}$, $\frac{1}{2}$, $\frac{1}{2} \times (\frac{1}{2} - 0) = \frac{1}{4}$, and $\frac{1}{2} \times (\frac{.9}{2} - 0) = \frac{.9}{4}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, 1, $\frac{1}{2}$ and $\frac{.9}{2}$.

Using the same technique that in the previous section, we know combine the precisions for these two sets of users.

Table 4. Precision-recall for the list A-D (Fetch&Browse). The precision for the four lists and four recall intervals are shown. The line "correspondence" show what are the number of ideal elements the user wants to see if (1) she considers that only elements with an exhaustivity 2 are ideal (2) she considers that elements with an exhaustivity at least 1 are ideal.

recall level	$]0, \frac{1}{3}]$	$]\frac{1}{3}, \frac{1}{2}]$	$]\frac{1}{2}, \frac{2}{3}]$	$]\frac{2}{3}, 1]$
correspondence	1,1	1,2	2,2	2,3
A	1	1	1	1
B	1	.88	.88	1
C	.81	.75	.56	.50
D	.81	.74	.55	.49

The evaluations are as expected; list **A** has the maximum score, followed by list **B** (where the two documents were swapped). Then list **C** is superior to **D**, although very close: the difference lies in that the user did not continue to explore the document as the first element, for list **D**, was not fully specific.

5 Discussion

In this article, we presented the EPRUM metric and how it was used in INEX 2005 to evaluate participant submissions. EPRUM is a generalisation of precision-recall. For instance, it reduces to standard recall-precision if browsing between elements is not allowed and element relevance is binary.

Most metrics used to compare the performance of semi-structured document search engines rely – sometimes implicitly – on a simplistic user behaviour: The user is supposed to consult exclusively the elements of the list returned by the engine. This user model is no more adapted to recent IR tasks like XML. In particular it does not allow considering user ability to navigate between elements, using the list as entry points to the information she seeks.

EvalJ

EPRUM is implemented in the EvalJ software, along with all the other metrics of INEX. It can be downloaded from this URL:
http://evalj.sourceforge.net `http://evalj.sourceforge.net`

References

[1] G. Kazai and M. Lalmas. Notes on what to measure in inex. In A. Trotman, M. Lalmas, and N. Fuhr, editors, *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*. University of Otago, Univeristy of Glasgow, Information Retrieval Festival, 2005.

[2] G. Kazai, M. Lalmas, and A. P. Vries. The overlap problem in content-oriented XML retrieval evaluation. In *Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 72–79, Sheffield, UK, July 2004. ACM Press.

[3] B. Piwowarski and P. Gallinari. Expected ratio of relevant units: A measure for structured information retrieval. In N. Fuhr, M. Lalmas, and S. Malik, editors, *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop*, Dagstuhl, France, Dec. 2003.

[4] B. Piwowarski, P. Gallinari, and G. Dupret. An extension of precision-recall with user modelling (PRUM): Application to XML retrieval. submitted for publication, 2005.

HiXEval: Highlighting XML Retrieval Evaluation

Jovan Pehcevski and James A. Thom

School of CS and IT, RMIT University, Melbourne, Australia
{jovanp, jat}@cs.rmit.edu.au

Abstract. This paper describes our proposal for an alternative XML retrieval evaluation that is solely based on the highlighted relevant text.

1 Introduction

How to properly evaluate the XML retrieval effectiveness is still an open research problem. INEX, as in previous years, is used as arena to investigate the behaviour of a variety of metrics. However, unlike in previous years, a new set of official metrics is adopted in INEX 2005, which belong to the eXtended Cumulated Gain (XCG) family of metrics [5, 6].

The following three metrics are the official INEX 2005 metrics used to measure the retrieval effectiveness of submitted runs:

1. `nxCG` – for a given rank `r`, `nxCG[r]` measures the relative gain a user has accumulated up to that rank, compared to the gain they could have accumulated if the system had produced the optimal ranking.
2. `ep/gr` (effort-precision/gain-recall) – measures the amount of relative effort (as the number of visited ranks) a user is required to spend compared to the effort they could have spent when inspecting an optimal ranking for a cumulated gain level.
3. `Q` and `R` – modified normalised cumulated gain measures which employ bonus gain functions that directly incorporate the rank position of the cumulated gain level.

The three official INEX 2005 metrics are explained in detail by Kazai and Lalmas [4].

In an effort to simplify the XML retrieval evaluation, we propose to solely incorporate the knowledge of the *highlighted* information for a given INEX topic. To obtain this knowledge, we use the statistics stored in the INEX 2005 relevance judgements gathered during the highlighting assessment task at INEX 2005.

The highlighting task is as follows: first, for a returned article the assessor is asked to highlight all its relevant content. Second, after the assessment tool automatically identifies the elements that enclose the highlighted content, the assessor is asked to judge the level of *exhaustivity* of these elements, and of all their ancestors and descendants. Last, based on the highlighted text, the level of

```
<file collection="ieee" name="co/2000/r7108">
 <element path="/article[1]" E="1" size="13556" rsize="5494"/>
 <element path="/article[1]/bdy[1]" E="1" size="9797" rsize="4594"/>
 <element path="/article[1]/bdy[1]/sec[1]" E="1" size="1301" rsize="409"/>
 <element path="/article[1]/bdy[1]/sec[2]" E="1" size="2064" rsize="2064"/>
 <element path="/article[1]/bdy[1]/sec[2]/st[1]" E="?" size="30" rsize="30"/>
 <element path="/article[1]/bdy[1]/sec[4]/p[1]" E="1" size="731" rsize="731"/>
 <element path="/article[1]/bm[1]/app[1]" E="1" size="2085" rsize="900"/>
 <element path="/article[1]/bdy[1]/sec[6]/ip1[1]" E="1" size="706" rsize="177"/>
</file>
```

Fig. 1. A sample from the INEX 2005 CO topic 203 relevance judgements for the relevant file `co/2000/r7108`. For each judged element, `E` shows the value for exhaustivity (with possible values `?`, 1 and 2), `size` denotes the element size (measured as total number of contained words), while `rsize` shows the actual number of words highlighted as relevant by the assessor.

specificity is computed automatically as a ratio of highlighted to fully contained text.

Table 1 shows a sample of relevance judgements obtained for the INEX 2005 CO topic 203. For each judged element, `E` shows the exhaustivity value of the element (with possible values `?`, 1 and 2), `size` denotes the total number of words contained by the element, while `rsize` shows the actual number of highlighted words by the assessor.

One approach of measuring the *relevance* of an element is to combine values obtained from the two INEX relevance dimensions. For example, if the observed value for `E` is 1 and both values for `size` and `rsize` are the same, the element is deemed as *highly specific* but only *partially exhaustive*. A quantisation function is then used to combine these two values into a number that is subsequently used to reflect the relevance of the element. However, the official INEX 2005 metrics treat each element with an `E` value of `?` as non-relevant, which means that these 'too small' elements do not bring any gain for the retrieval evaluation. We argue that a system-oriented evaluation metric should also take these elements into account, particularly because they *do* contain highlighted (and thus relevant) information.

We contend that the purpose of the XML retrieval task is to find elements that contain *as much relevant information as possible*, without also containing a significant amount of non-relevant information. Therefore, to measure the extent to which an XML retrieval system returns relevant information, we follow an approach that only takes into account the amount of highlighted text in a retrieved element, without considering the `E` value of that element. We propose `HiXEval` (pronounced hi–ex–eval) – an alternative evaluation metric for XML retrieval that is based on the traditional definitions of precision and recall which have been extended to include the knowledge obtained from the INEX 2005 highlighting assessment task.

In INEX 2004, the *set-based overlap* was used as an indicator of the level of overlap between the returned elements [2]. In this paper we describe four overlap indicators, of which three are derived from the set-based overlap. We believe that by having more than one indicator of overlap, the nature of the overlap problem can be understood better.

The remainder of this paper is organised as follows. In Section 2 we provide a brief description of our hybrid XML retrieval approach that is used to generate the INEX 2005 runs. Samples of these runs are used as preliminary examples in this paper. In Section 3 we provide a formal definition of our alternative INEX 2005 evaluation metric, and provide some preliminary observations of what is measured. In Section 4 we describe the four overlap indicators, which aid in better understanding of the nature of the overlap problem. In Section 5, by using the alternative INEX 2005 metric, we evaluate the retrieval effectiveness of our official INEX 2005 runs for each retrieval strategy in the INEX 2005 `CO+S` sub-task. We conclude in Section 6 by outlining possible avenues for future work.

2 Description of the hybrid XML retrieval approach

In this section we provide a very brief description of our hybrid XML retrieval approach used to generate the INEX 2005 runs. For further details see our INEX 2005 ad hoc paper [8].

The system we use in INEX 2005 follows a *hybrid* XML retrieval approach, combining information retrieval features from Zettair[1] (a full-text search engine) with XML-specific retrieval features from eXist[2] (a native XML database). The hybrid approach can be seen as a “fetch and browse” [1] XML retrieval approach, since full articles estimated as likely to be relevant to a query are first retrieved by Zettair (the *fetch* phase), and then the most specific elements within these articles are extracted by eXist (the *browse* phase) [10].

Three similarity measures are currently implemented in Zettair, each based on one of the following three information retrieval models: the vector-space model, the probabilistic model, and the language model. For the *fetch* phase of our hybrid system, we investigate which of the three information retrieval models (`PCosine`, `Okapi`, or `Dirichlet`) yields best effectiveness for full article retrieval.

To identify and rank the appropriate granularity of elements to return as answers, we use a retrieval module that utilises the structural information in the eXist list of extracted elements. Our retrieval module presents what we call *Coherent Retrieval Elements* (CREs) as final answers. For the *browse* phase of our hybrid system, we investigate which combining choice – among the two ways for identifying CREs (`nCRE` and `oCRE`) and the two XML-specific heuristics

[1] `http://www.seg.rmit.edu.au/zettair/`

[2] `http://exist-db.org/`

Article	nCRE answer element	T-matches	P-length	F-frequency
co/2000/r7108	/article[1]/bdy[1]/sec[2]	3	3	9
co/2000/r7108	/article[1]/bdy[1]	3	2	31
co/2000/r7108	/article[1]	3	1	39
co/2000/r7108	/article[1]/bdy[1]/sec[6]/ip1[1]	2	4	2
co/2000/r7108	/article[1]/bm[1]/app[1]	2	3	8
co/2000/r7108	/article[1]/bdy[1]/sec[1]	2	3	5
co/2000/r7108	/article[1]/bdy[1]/sec[4]/p[1]	1	4	2

Table 1. Ranked list of nCRE elements using the TPF heuristic combination for article co/2000/r7108. The query used is "code signing verification", which represents the title part of the INEX 2005 topic 203.

for ranking the CREs ((TPF and PTF2)) – yields best effectiveness for element retrieval.

To identify the appropriate element granularity, two types of answer elements are returned by our retrieval module: oCRE and nCRE. The oCRE answer elements typically represent less specific elements, with the expectation that they would provide better context for the contained text than that provided by more specific elements. The nCRE answer elements, on the other hand, additionally include most specific (leaf) elements as answers, with the expectation that these newly included elements would also allow for more focused retrieval.

With the TPF ranking heuristic, first the CREs are sorted in a descending order according to the number of distinct query terms a CRE contains (the more distinct query terms it contains, the higher its rank). Next, if two CREs contain the same number of distinct query terms, the one with the longer length of its absolute path is ranked higher (which ensures that more specific elements are preferred over less specific ones). Last, if the lengths of the two absolute paths are also the same, the CRE with more frequent query term appearances is ranked higher than the CRE where query terms appear less frequently. The final answer list when the TPF ranking heuristic is used is shown in Table 1.

With PTF2, the CREs are first sorted in a descending order according to the length of the absolute path of a CRE (where the longer CRE path results in a higher rank, although the CREs containing exactly one query term are moved at the end of the ranked list). Next, if the absolute path lengths of two CREs are the same, the one that contains more distinct query terms is ranked higher. Last, if it also happens that both numbers of distinct query terms are the same, the CRE with more frequent query term appearances is ranked higher. The final answer list when the PTF2 ranking heuristic is used is shown in Table 2.

3 HiXEval – an alternative metric for XML retrieval evaluation

The HiXEval metric credits systems for retrieving elements that contain as much highlighted (relevant) textual information as possible, without also containing a

Article	nCRE answer element	P-length	T-matches	F-frequency
co/2000/r7108	/article[1]/bdy[1]/sec[6]/ip1[1]	4	2	2
co/2000/r7108	/article[1]/bdy[1]/sec[2]	3	3	9
co/2000/r7108	/article[1]/bm[1]/app[1]	3	2	8
co/2000/r7108	/article[1]/bdy[1]/sec[1]	3	2	5
co/2000/r7108	/article[1]/bdy[1]	2	3	31
co/2000/r7108	/article[1]	1	3	39
co/2000/r7108	/article[1]/bdy[1]/sec[4]/p[1]	4	1	2

Table 2. Ranked list of `nCRE` elements using the `PTF2` heuristic combination for article `co/2000/r7108`. The query used is "code signing verification", which represents the `title` part of the INEX 2005 topic 203.

significant amount of non-relevant information. Therefore, to measure the extent to which an XML retrieval system returns relevant information, we only take into account the amount of highlighted text in a retrieved element, without considering the exhaustivity value of that element. We propose to extend the traditional definitions of precision and recall as follows.

$$Precision = \frac{amount\ of\ relevant\ information\ retrieved}{total\ amount\ of\ information\ retrieved}$$

$$Recall = \frac{amount\ of\ relevant\ information\ retrieved}{total\ amount\ of\ relevant\ information}$$

For each element **e** that belongs to a ranked list of elements returned by the system for an INEX topic, we denote $rsize(\mathbf{e})$ as the number of highlighted (relevant) words in **e**, and $size(\mathbf{e})$ as the total number of words contained by **e**. Three distinct scenarios are possible for each element **e** that belongs to this ranked list:

1. **e** is a not-yet-seen element (*NS*);
2. **e** has been fully seen previously (*FS*), and
3. **e** is an element-part (*EP*), which has been in part seen previously.

To measure the amount of *retrieved relevant information* from an element **e** at a given rank **r**, a relevance value function $\mathbf{pre_r}(\mathbf{e})$ is defined as follows.

$$\mathbf{pre_r}(\mathbf{e}) = \begin{cases} \frac{rsize(\mathbf{e})}{size(\mathbf{e})} & if\ \mathbf{e}\ is\ NS \\ (1-\alpha)\cdot\frac{rsize(\mathbf{e})}{size(\mathbf{e})} & if\ \mathbf{e}\ is\ FS \\ \alpha\cdot\frac{rsize(\mathbf{e})-\sum_{e'} rsize(\mathbf{e}')}{size(\mathbf{e})} + (1-\alpha)\cdot\frac{rsize(\mathbf{e})}{size(\mathbf{e})} & if\ \mathbf{e}\ is\ EP \end{cases}$$

where $\mathbf{e}'$ represents an already retrieved element, descendant of $\mathbf{e}$, which appears before $\mathbf{r}$ in the ranked list (if any).

The function $\mathbf{pre_r}(\mathbf{e})$ is used to ensure that, to achieve a precision gain at rank $\mathbf{r}$, the retrieved element $\mathbf{e}$ needs to contain *as little non-relevant information as possible.* The parameter α is a weighting factor, and it represents the importance of retrieving non-overlapping elements in the ranked list. Setting α to 1 ensures that the system will only be credited for retrieving relevant information that has not been previously retrieved by other overlapping elements. On the other hand, setting α to 0 ensures that the system is always credited for retrieving relevant information, regardless of whether the same information has already been retrieved.

To measure the amount of *relevant information retrieved* from an element $\mathbf{e}$ at a given rank $\mathbf{r}$, a relevance value function $\mathbf{rec_r}(\mathbf{e})$ is defined as follows.

$$\mathbf{rec_r}(\mathbf{e}) = \begin{cases} rsize(\mathbf{e}) & \textit{if } \mathbf{e} \textit{ is NS} \\ (1-\alpha)\cdot rsize(\mathbf{e}) & \textit{if } \mathbf{e} \textit{ is FS} \\ \alpha \cdot \left(rsize(\mathbf{e}) - \sum_{e'} rsize(\mathbf{e}') \right) + (1-\alpha)\cdot rsize(\mathbf{e}) & \textit{if } \mathbf{e} \textit{ is EP} \end{cases}$$

The function $\mathbf{rec_r}(\mathbf{e})$ is used to ensure that, to achieve a recall gain at rank $\mathbf{r}$, the retrieved element $\mathbf{e}$ needs to contain *as much relevant information as possible.* The parameter α is used for the same purpose of indicating the importance of retrieving non-overlapping elements as with the $\mathbf{pre_r}(\mathbf{e})$ function.

Let **Trel** be the total amount of relevant information for an INEX topic (if $\alpha = 1$, then this is the number of highlighted words across all documents; if $\alpha = 0$, then this is the number of highlighted words across all elements). Given the above definitions of the two relevance value functions, we define precision and recall at a given rank $\mathbf{r}$ as follows:

$$\mathbf{Precision@r} = \frac{\sum_{\mathbf{i=1}}^{\mathbf{r}} \mathbf{pre_i}(\mathbf{e})}{\mathbf{r}}$$

$$\mathbf{Recall@r} = \frac{\sum_{\mathbf{i=1}}^{\mathbf{r}} \mathbf{rec_i}(\mathbf{e})}{\mathbf{Trel}}$$

To combine both values for precision and recall into a single value for a given rank $\mathbf{r}$, we define the **F**-measure (the *harmonic mean*) as follows.

$$\mathbf{F@r} = \frac{2}{\frac{1}{\mathbf{Precision@r}} + \frac{1}{\mathbf{Recall@r}}}$$

	Precision@r				**Recall@r**				**F@r**					
Run	1	3	5	7	1	3	5	7	1	3	5	7	**iAP**	**nAP**
nCRE-TPF	**1.00**	0.44	0.26	0.19	**0.38**	**1.00**	**1.00**	**1.00**	**0.55**	**0.61**	0.42	0.32	**0.73**	**0.69**
nCRE-PTF2	0.25	**0.56**	**0.44**	**0.31**	0.03	0.57	1.00	1.00	0.06	0.57	**0.61**	**0.48**	0.54	0.47

Table 3. Evaluation results with using HiXEval for two INEX 2005 CO run samples. Value 1 was used for α, which gives a value of 5494 for **Trel**. The best results for each measure are shown in bold.

By comparing the **F@r** values obtained from different systems, it would be possible to see which system yields the best trade-off between precision and recall for a given rank. That is, in line with our previous argument, it would be possible to see which system is more capable of retrieving *as much relevant information as possible*, without also retrieving a significant amount of non-relevant – or even redundant – information.

3.1 Preliminary experiments

To test the proposed metric, we evaluate results obtained from two samples taken from two of our INEX 2005 runs: nCRE-TPF and nCRE-PTF2. The first sample uses the TPF ranking heuristic to present the resulting elements (shown in Table 1), while the second uses the PTF2 ranking heuristic (shown in Table 2). We use the relevance judgements shown in Fig. 1 as a recall-base in this example.

Table 3 shows evaluation results with using HiXEval for the two run samples. The value for parameter α is set to 1, which means that overlap between retrieved elements is considered. The value for **Trel** in this case is 5494, reflecting the total number of highlighted words that need to be retrieved to find all the relevant information. In addition to the measures explained previously, we also report values for **iAP** and **nAP**, which represent the interpolated average precision (calculated at 11 recall points) and non-interpolated average precision (calculated at each natural recall point), respectively.

We observe that the run sample that retrieves more general elements early in the ranking (TPF) produces better recall for all ranks, and it is also better on average than the run sample that retrieves less general elements first (PTF2). However, its precision quickly drops after a few elements are retrieved, since in this case all the relevant information *has already been retrieved previously*, so the system is not credited for retrieving the same information again. The system using the PTF2 ranking heuristic, on the other hand, retrieves new relevant information at each rank, resulting in increased precision *and* recall. As shown in the table, the trade-off between precision (retrieving as little non-relevant information as possible) and recall (retrieving as much relevant information as possible) is correctly captured by the **F**-measure, which produces higher values for the PTF2 run sample after five elements are returned.

Table 4 shows evaluation results when the value for parameter α is set to 0, which means that overlap between retrieved elements is not considered. The

	Precision@r				Recall@r				F@r					
Run	1	3	5	7	1	3	5	7	1	3	5	7	**iAP**	**nAP**
nCRE-TPF	**1.00**	**0.62**	**0.51**	**0.55**	**0.14**	**0.84**	**0.92**	**0.99**	**0.25**	**0.72**	**0.66**	**0.71**	**0.66**	**0.63**
nCRE-PTF2	0.25	0.56	0.49	0.55	0.01	0.22	0.57	0.99	0.02	0.31	0.53	0.71	0.52	0.49

Table 4. Evaluation results with using HiXEval for two INEX 2005 CO run samples. Value 0 was used for α, which gives a value of 14399 for **Trel**. The best results for each measure are shown in bold.

value for **Trel** in this case is 14399, reflecting the total number of (overlapping) highlighted words contained within all the relevant elements for this topic. We observe that the TPF run sample consistently produces better scores, irrespective of which measure is used. Ignoring overlap between retrieved elements, therefore, does not seem to properly reflect the actual performance of the two run samples.

4 Measuring Overlap

When measuring the retrieval performance of each submitted run, INEX also reports the observed level of overlap between the returned elements. In INEX 2004, the *set-based overlap* was used as an indicator of the level of overlap between the returned elements [2]. As currently defined, for a *set* of retrieved elements it measures the percentage of elements that either *contain* or are *contained by* at least one other element in the set. However, Hiemstra and Mihajlovic [3] argue that the set-based overlap appears to be a somewhat unstable measure, and that a probabilistic overlap measure could be a better indicator of the observed level of overlap. To support their argument for a probabilistic overlap measure, they refer the following example: Consider the set of 1500 retrieved elements, of which 1499 are non-overlapping, and there is only one element that fully contains each of the 1499 elements (we call this set as *Set 1500*). According to its current definition, the set-based overlap would be 100%, which does not correctly capture the nesting relationships among these elements.

We describe four different ways of measuring overlap [9], of which three are derived from the set-based overlap. They are defined as follows.

1. *O*verall overlap (O-overlap), which is identical to the set-based overlap as defined previously;
2. *A*scendants overlap (A-overlap), which for a set of retrieved elements measures the percentage of elements that *contain* at least one other element in the set;
3. *D*escendents overlap (D-overlap), which for a set of retrieved elements measures the percentage of elements that are *contained by* at least one other element in the set; and
4. *P*robabilistic overlap (P-overlap), which for a set of retrieved elements measures the probability that two randomly chosen elements from the set overlap with each other.

Set	**Overlap measure (%)**			
	`O-overlap`	`A-overlap`	`D-overlap`	`P-overlap`
	`(@5)`	`(@5)`	`(@5)`	`(@5)`
Set A	100.00	40.00	80.00	60.00
Set B	80.00	20.00	60.00	30.00
Set 1500	100.00	0.07	99.93	0.13

Table 5. Overlap values at five elements retrieved for three element sets, when four different overlap measures apply.

Consider the two sets of retrieved elements shown in Tables 1 and 2, respectively. The two sets, which we name *Set A* and *Set B*, are drawn from two INEX 2005 CO runs that use different ranking heuristics (`TPF` and `PTF2`). We apply the four overlap indicators, as defined above, to measure the level of overlap in the following three cases: Two cases when each of the two sets, *Set A* and *Set B*, is considered individually, and a case when only *Set 1500* is considered.

Table 5 shows the overlap values at five elements retrieved (`@5`) for each of these three cases, when four different overlap measures apply. We observe that the `O-overlap` measure constantly produces high overlap values, irrespective of which set is used. On the other hand, both `A-overlap` and `D-overlap` can be seen as useful, informative complements to `O-overlap`. Indeed, `D-overlap` provides information about the proportion of elements that are *contained by* at least one other element in the set, whereas `A-overlap` indicates how these contained elements are distributed among the *containing* ancestors (which corresponds to the number of nesting layers in the hierarchy). For example, the lower `A-overlap` value for *Set B* (20% compared to 40% for *Set A*) indicates that the distribution of overlapping elements for *Set B* is likely to be less hierarchical than that for *Set A* (which is actually the case). This is particularly evident in the case of *Set 1500*, where the observed `A-overlap` value is 0.07% – a very low value that confirms the flat inner distribution of the 1499 elements in the only one containing element.

Table 5 also shows that, although the `P-overlap` measure exhibits rather different behaviour than any of the other three measures, it still appears to correctly capture the nature of overlap when both sets A and B are considered individually: Indeed, the probability of randomly choosing two overlapping elements is lower for *Set B* than for *Set A* (30% compared to 60%). However, an inherent property of `P-overlap` is that when a set of elements that belong to different files is considered (which is usually the case with submitted runs), `P-overlap` typically reports a low overlap value, since for such sets there are many possible combinations of randomly choosing two elements, but few of them actually consider elements that belong to the same file (such that there is a possibility for them to overlap). Nevertheless, the `P-overlap` measure still seems to be a reliable overlap indicator in cases where the overlapping elements belong to the same file. In the case of *Set 1500*, for example, the observed `P-overlap` value is 0.13% – a value far lower than the one observed when using the O-overlap measure.

The above examples clearly show that more than one overlap indicator needs to be used if the nature of overlap is to be understood better. The three overlap indicators, O-overlap, A-overlap, and D-overlap, have also been chosen to be used as official INEX 2005 overlap indicators.

5 Experiments and results

In this section, we present evaluation results of our submitted INEX 2005 runs obtained from HiXEval for each retrieval strategy in the CO+S sub-task. We set the value of α to 1 and report a range of **F**-measure values, along with the values for iMAP and nMAP, which represent the interpolated mean average precision (calculated at 11 recall points) and non-interpolated mean average precision (calculated at each natural recall point), respectively.

Three retrieval strategies are explored in the CO+S sub-task at INEX 2005: Thorough, Focused, and FetchBrowse [7]. We use different variations of HiXEval for each of these three strategies. For example, for the Focused strategy we use a HiXEval variation which does not tolerate retrieving overlapping elements, regardless of whether the retrieved element has been seen fully or in part. On the other hand, for the FetchBrowse strategy we use a HiXEval variation which does not tolerate retrieving elements that are not properly grouped by an article. In both cases, such retrieved elements are regarded as non-relevant elements, which is subsequently reflected in the evaluation scores. In the following we present the performance results of our runs for each of the three strategies.

5.1 Thorough and Focused retrieval strategies

The evaluation results of our INEX 2005 CO+S runs for the Thorough strategy are shown in the upper part of Table 6. At 500 or less elements returned, better system performance is achieved with returning oCRE answer elements than with returning the nCRE elements. Similar behaviour is observed for the Focused retrieval strategy (shown in the lower part of Table 6). In fact, we observe 16% relative iMAP performance improvement when oCRE elements are returned for both retrieval strategies.

For the Thorough strategy at ten or less returned elements, the system performance is higher when using the TPF ranking heuristic than when using PTF2, although when retrieving more than ten elements we observe better performance with the PTF2 ranking heuristic. The two mean average precision values also confirm that better overall performance is achieved with the PTF2 ranking heuristic.

The above findings show that, to gain best retrieval value for the Thorough retrieval strategy under HiXEval, an XML retrieval system first needs to identify all the *contextual* elements that may represent relevant answers, and then retrieve the *most specific* among them.

	F@r										
Run	5	10	15	25	50	100	500	1000	1500	iMAP	nMAP
Thorough											
nCRE-PTF2	0.043	0.061	0.075	0.093	0.108	0.107	0.079	**0.061**	**0.050**	0.112	0.122
nCRE-S-PTF2	0.101	**0.127**	0.129	0.112	0.109	0.094	0.065	0.049	0.039	0.134	0.161
nCRE-TPF	0.066	0.071	0.072	0.068	0.068	0.066	0.038	0.027	0.021	0.087	0.082
nCRE-S-TPF	**0.116**	**0.127**	0.126	0.103	0.091	0.075	0.042	0.027	0.020	0.122	0.137
oCRE-PTF2	0.052	0.075	0.094	0.121	**0.130**	**0.127**	**0.083**	0.060	0.045	0.131	0.135
oCRE-S-PTF2	0.106	**0.127**	**0.132**	**0.125**	0.118	0.101	0.068	0.048	0.037	**0.140**	**0.165**
Focused											
nCRE-PTF2-NO	0.043	0.065	0.090	0.108	0.131	0.143	**0.102**	**0.079**	**0.057**	0.132	0.179
nCRE-S-PTF2-NO	0.105	0.124	0.132	0.121	0.116	0.111	0.079	0.059	0.043	0.145	0.199
nCRE-TPF-NO	0.083	0.099	0.118	0.131	0.153	**0.155**	0.094	0.053	0.036	0.149	0.185
nCRE-S-TPF-NO	**0.130**	**0.148**	**0.147**	0.132	0.130	0.120	0.073	0.042	0.029	**0.156**	**0.200**
oCRE-PTF2-NO	0.064	0.095	0.128	**0.154**	**0.164**	0.150	0.099	0.064	0.045	0.152	0.198
oCRE-S-PTF2-NO	0.110	0.130	0.144	0.140	0.131	0.112	0.077	0.050	0.035	0.147	0.199

Table 6. Evaluation results of our INEX 2005 CO+S runs for the Thorough (upper part) and Focused (lower part) retrieval strategies, when using the HiXEval metric. For each retrieval strategy, the best results under each measure are shown in bold.

For the Focused retrieval strategy at 100 or less returned elements, we observe that the TPF ranking heuristic performs better than PTF2. In this case better overall performance is also achieved with TPF. Thus, to gain best retrieval value for the Focused retrieval strategy under HiXEval, the system needs to identify and retrieve non-overlapping, contextual, and *less specific* elements. This is in contrast with our finding for the Focused retrieval strategy when the nxCG evaluation metric is used, where the best system performance is achieved by using nCRE answer elements with the PTF2 ranking heuristic [8].

We investigate the usefulness of using structural hints in the +S topics by comparing the performances of the CO+S runs that ignore and strictly interpret these hints, respectively. As shown in Table 6, each of the three +S runs performs consistently better than its corresponding CO run at 15 or less elements returned, irrespective of the retrieval strategy used. Interestingly, we also observe that all the three +S runs produce higher mean average precision values than their corresponding CO runs.

A useful feature of the HiXEval metric is that it allows a seamless comparison between the performance of runs used in different retrieval strategies. For example, by comparing each Focused run with its corresponding Thorough run, it is possible to determine which retrieval strategy brings better gain in retrieving relevant information. As shown in Table 6, each CO and +S run used in the Focused retrieval strategy performs consistently better than its corresponding run used in the Thorough strategy. The latter finding shows that the HiXEval metric is capable of penalising systems that retrieve *redundant relevant informa-*

	F@r										
Run	5	10	15	25	50	100	500	1000	1500	iMAP	nMAP
FetchBrowse-D											
nCRE-Okapi-PTF2	**0.153**	0.141	0.137	0.127	0.104	0.069	0.020	**0.010**	**0.007**	0.128	**0.101**
nCRE-PCosine-PTF2	0.145	**0.151**	**0.153**	**0.133**	**0.105**	**0.072**	**0.021**	**0.010**	**0.007**	**0.130**	0.097
nCRE-Dirichlet-PTF2	0.136	0.137	0.134	0.127	0.100	0.068	0.020	**0.010**	**0.007**	0.117	0.099
FetchBrowse											
nCRE-Okapi-PTF2	0.043	0.061	0.075	0.093	0.108	0.107	0.079	**0.061**	**0.050**	0.112	0.122
nCRE-S-Okapi-PTF2	**0.101**	**0.127**	**0.129**	0.112	**0.109**	0.094	0.065	0.049	0.039	**0.134**	**0.161**
nCRE-PCosine-PTF2	0.026	0.052	0.067	0.087	0.091	0.100	0.076	0.060	0.049	0.096	0.105
nCRE-S-PCosine-PTF2	0.094	0.110	0.109	0.110	0.098	0.098	0.062	0.048	0.040	0.124	0.137
nCRE-Dirichlet-PTF2	0.037	0.059	0.077	0.087	0.108	**0.109**	**0.080**	**0.061**	**0.050**	0.108	0.125
nCRE-S-Dirichlet-PTF2	0.077	0.117	0.125	**0.116**	0.107	0.093	0.062	0.048	0.040	0.130	0.154

Table 7. Evaluation results of our INEX 2005 CO+S runs for the FetchBrowse-D (upper part) and FetchBrowse (lower part) retrieval strategies, when using the HiXEval metric. For each retrieval strategy, the best results under each measure are shown in bold.

tion – a feature that has also been identified as useful by users in interactive XML retrieval experiments [11]. With their current setup, none of the official INEX 2005 metrics are capable of comparing runs across different retrieval strategies.

5.2 FetchBrowse retrieval strategy

Table 7 shows evaluation results of our INEX 2005 CO+S runs for the FetchBrowse retrieval strategy.

The upper part of Table 7 shows results when only full articles represent units of retrieval. We observe that at five articles returned Okapi produces the best performance, whereas PCosine is dominant when returning ten or more articles. At 1000 and more returned articles, the system performance is identical regardless of which of the three measures is used. Overall, PCosine seems to perform best for full article retrieval, followed by Okapi and Dirichlet. This is in contrast with our reported finding when inex_eval is used with strict quantisation function, where best overall system performance is achieved with the Okapi similarity measure [8].

The lower part of Table 7 shows results for the FetchBrowse retrieval strategy when elements are units of retrieval, where we investigate the extent to which each of the three similarity measures influences the system performance. We observe that Okapi yields best element retrieval at 50 or less elements returned, while Dirichlet is dominant when more than 50 elements are returned. Overall, Okapi seems to perform best for element retrieval, followed by Dirichlet and PCosine.

When structural constraints in the +S topics are strictly followed, we observe a constant increase in precision at 50 or less elements returned, irrespective of the similarity measure used. As in the previous two retrieval strategies, all the three `+S` runs produce higher mean average precision values than their corresponding `CO` runs. This finding shows that using structural hints in the INEX `+S` topics is also a useful feature for the `FetchBrowse` retrieval strategy as it is for the other two strategies.

6 Discussion

The simple idea behind `HiXEval` could also be applied to the `XCG` family of metrics. Indeed, for the `Focused` retrieval task the `XCG` metric first needs to build the so-called 'ideal recall-base', which is subsequently used with the full recall-base to evaluate the XML retrieval effectiveness [5]. Instead the currently used methodology to select the ideal nodes, we propose an alternative methodology where each relevant element may be assigned a relevance score that represents the **F**-measure value obtained *solely* for that element. As an example, the score of each element that belongs to the recall-base sample shown in Fig. 1 can be determined by first separately calculating the values for precision and recall for that element in isolation, and then by combining the two values into an **F**-measure value. For instance, for the element `/article[1]/bdy[1]` we calculate the following:

Precision[**bdy**] $= 4594/9797 = 0.47$ (the fraction of *retrieved relevant information* obtained solely from that element), **Recall**[**bdy**] $= 4594/5494 = 0.84$ (the fraction of *relevant information retrieved* obtained from the element), and finally **F**[**bdy**] $= 2 * 0.47 * 0.84/(0.47 + 0.84) = 0.60$ (the *harmonic mean*). The relevant elements could then be sorted in a descending order according to their assigned harmonic mean values. Finally, to identify the ideal elements, a top-down filtering approach could be applied to remove all the overlapping elements.

As a result of applying the above methodology, we identify the two elements, `/article[1]/bdy[1]` and `/article[1]/bm[1]/app[1]`, as our ideal element nodes for our recall-base sample shown in Fig. 1.

In this paper we have only reported experiments with `HiXEval` using our submitted INEX 2005 runs for the `CO+S` sub-task. We plan to extend this work by generating performance numbers for our submitted `CAS` runs. Naturally, we also aim to evaluate and report the performance of all the official INEX 2005 runs submissions.

Our preliminary results of using `HiXEval` with $\alpha = 0$ suggest that the overlap could be seen as a controlled variable when measuring the retrieval performance; that is, it would be interesting to see the extent to which a change in the observed level of overlap influences the observed system performance. We also plan to pursue this investigation further as part of our future work.

References

1. Y. Chiaramella, P. Mulhem, and F. Fourel. A model for multimedia information retrieval. Technical report, FERMI ESPRIT BRA 8134, University of Glasgow, April 1996.

2. A. de Vries, G. Kazai, and M. Lalmas. Evaluation metrics 2004. In *INEX 2004 Workshop Pre-Proceedings, Dagstuhl Castle, Germany, December 6-8, 2004*, pages 249–250, 2004.
3. D. Hiemstra and V. Mihajlovic. The simplest evaluation measures for XML information retrieval that could possibly work. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 6–13, Glasgow, UK, 2005.
4. G. Kazai and M. Lalmas. INEX 2005 evaluation metrics. 2005.
Available at URL: `http://inex.is.informatik.uni-duisburg.de/2005/`.
5. G. Kazai and M. Lalmas. Notes on what to measure in INEX. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 22–38, Glasgow, UK, 2005.
6. G. Kazai, M. Lalmas, and A. P. de Vries. The overlap problem in content-oriented XML retrieval evaluation. In *Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 72–79, Sheffield, UK, 2004.
7. M. Lalmas. INEX 2005 retrieval task and result submission specification. 2005.
Available at URL: `http://inex.is.informatik.uni-duisburg.de/2005/`.
8. J. Pehcevski, J. A. Thom, and S. M. M. Tahaghoghi. RMIT University at INEX 2005. In *Pre-Proceedings of the Fourth INEX Workshop, Dagstuhl, Germany, November 28–30, 2005*, 2005.
9. J. Pehcevski, J. A. Thom, S. M. M. Tahaghoghi, and A.-M. Vercoustre. Relevance for XML retrieval: The user perspective. (submitted for publication).
10. J. Pehcevski, J. A. Thom, and A.-M. Vercoustre. Hybrid XML retrieval: Combining information retrieval and a native XML database. *Information Retrieval*, 8(4):571–600, 2005.
11. J. Pehcevski, J. A. Thom, and A.-M. Vercoustre. Users and assessors in the context of INEX: Are relevance dimensions relevant? In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 47–62, Glasgow, UK, 30 July 2005.

XCG Overlap at INEX 2004

Alan Woodley Shlomo Geva

School of Data Communications and Software Engineering
Queensland University of Technology
PO Box 2434, Brisbane 4001, Queensland
{ap.woodley@student.qut.edu.au,s.geva@qut.edu.au}

Abstract. Since 2002, the INitiative for the Evaluation of XML Retrieval (INEX) has set the benchmark for rigorous evaluation of XML information retrieval (XML-IR) systems and approaches. INEX has based much of its evaluation methodology on that of earlier text retrieval IR workshops, in particular TREC. However, INEX evaluation was modified for the specific requirements of XML-IR since unlike traditional IR, which is concerned with document retrieval, XML-IR is concerned with the retrieval of the most suitable document component in response to a query. It is much closer to passage retrieval, but is oriented towards XML documents with rich structural semantics. Under the current INEX metric, systems are evaluated against a complete recall base, which includes many overlapping elements. The treatment of overlapping elements has been the most contentious issue at INEX, with the accepted conjecture been that the system rank of participants would dramatically change if the metric treated overlap differently. In this paper, we show that contrary to popular belief the treatment of overlap does not dramatically affect system ranks. This has significant implications for the interpretation of INEX performance comparisons.

1 Introduction

In XML-IR the unit of retrieval is not a document but an XML element within the document tree. The objective of XML-IR is to identify the optimal unit of retrieval in response to a user query. Therefore, it is necessary to evaluate all Candidate Units of Retrieval (CUR) within a document. Overlap between CURs exists when an element is deemed relevant, as well as one or more of its ancestors/descendents. For instance, a section may be deemed relevant as well as several paragraphs within the section. Usually only a few of these units (or even only one unit) may be deemed to be *optimal* units of retrieval, but any of these units may be deemed to be a *suitable* unit of retrieval. The evaluation of a particular retrieval run (a list of ranked result elements in response to a query) necessitates the creation of a recall base that contains all *suitable* elements and consequently many overlapping elements.

A problem with evaluation arises when a system returns relevant overlapping elements. The traditional evaluation procedures reward the return of all relevant results because overlap does not usually arise in document based collections (except in the case of finding several copies of the same document). However, overlap exists in XML-IR due to the inherent hierarchal nature of XML documents.

Since 2002, the INitiaive for the Evaluation of XML Retrieval (INEX) has been the benchmark for evaluating the performance of XML-IR systems. The existing INEX metric, INEX 2002, was based on the traditional evaluation models, and consequently systems that return overlapping elements are rewarded for each returned element. From the end-user perspective, overlap is not a desirable property since the same information may be returned multiple times. Therefore, a number of systems that participate in INEX attempt to pick only the optimal units of retrieval within a document. This creates two classes of systems: some that are concerned with focused retrieval, and others that are concerned with thorough retrieval. When evaluated with the INEX 2002 metric, which does not consider overlap, the focused systems are disadvantaged. Since 2002 there have been 5 different metrics proposed, each with 8 different quantisations. The objective of these new metrics was to devise a scheme that will not reward systems that return overlapping results, or conversely, will not penalize systems that return overlap-free results. Previous work has indicated that there is little correlation between the rank of systems using the new metrics and the existing INEX 2002 metric. However, it is unknown to the extent (if any) that the new metrics penalise systems with overlapping results and reward systems that return overlap-free results.

We have taken a different approach to investigating overlap. Instead of deriving a new metric to handle overlap, we explicitly remove overlap from all INEX 2004 submissions and then re-evaluate them using the XCG metric, thereby allowing us to investigate the extent that the XCG metric penalises high-overlap systems. We also re-evaluate the original INEX 2004 submissions with the XCG metric, and compare the two systems ranks. These two system ranks are analogues to the focused and thorough tasks at INEX 2005. This comparison allows us to determine if the better scoring systems are retrieving more relevant results and therefore have a superior retrieval strategy, or if they are being unfairly rewarded via overlap. While others have previously compared the effect of explicitly removing overlap from a single submission [14], this is the first time that such a comprehensive analysis of INEX submissions has been conducted. It is also the first time that non-overlapping systems will be compared with other non-overlapping systems. Our results indicate that the XCG metric does penalise against system with overlapping systems. Furthermore, while that removing overlap from systems may change their rank, the difference is not as dramatic as first thought.

The rest of the paper is organised as follows: Section 2 provides an overview of XML-IR and INEX. Section 3 details the nature of overlap within XML-IR including metrics used to overcome the overlap ‘problem’. Section 4 describes our own solution for dealing with overlap. Finally, Section 5 presents the results of our experiments.

2 INitiative for the Evaluation of XML Retrieval (INEX)

2.1 Task and Overview

The main task for traditional information retrieval systems is to present a list of ranked documents in response to a user query. However, the separation of content and structure in XML documents allows XML-IR users to receive results more precise than the document level. Therefore, the task of XML-IR systems is two-fold: first the ability to locate elements that satisfy the user's information need, and second the ability to choose the most appropriately sized elements. In practice, this means that users of XML-IR systems might retrieve a relevant section, sub-section or paragraph from a relevant document, rather than the document itself. The user may explicitly request the unit of retrieval (Content and Structure queries) or it may be completely up to the retrieval engine to determine the appropriate unit (Content Only queries).

The INitiative for the Evaluation of XML Retrieval (INEX) provides the infrastructure for the evaluation of XML-IR systems via a large XML test collection and appropriate scoring mechanisms. INEX, like TREC, uses an evaluation process based upon the Cranfield methodology [2]. Each year the following process is followed:

1. Participants contribute topics (end user queries) and a subset of topics is selected for evaluation. INEX uses two types of topics to evaluate systems, Content Only and Content and Structure.
2. The topics are distributed to participants who execute them on their search engines and for each topic produce a ranked list of results. The top 1500 ranked results for each topic are combined into a single submission file. Participates are allowed to send between 1 and 3 submissions per task to INEX.
3. The top *n* (currently set to 100) results from each submission are pooled together, disassociated from their originating submissions and duplicates are eliminated. This is called the system pool.
4. The results in the system pool are individually judged by the original topic contributors, who act as end users manually assessing the relevance of the results. When judges find a document with a relevant result they must search the document for other relevant results, thereby increasing the size of the pool.
5. Using the assessment set and a standard evaluation module, the participating search engines are scored in terms of performance (recall/precision) using seven variations of the INEX 2002 metric. The scores are aggregated to produce an official Aggregate Mean Average Precision (AMAP), which is used to officially rank systems.
6. Results are returned to participants who then write and present papers at the workshop.

Historically, the method of system pooling and relevance assessments has not been without criticism. The two main criticisms have been how the results are chosen for assessments, and how to handle the subjective nature of relevance. In INEX, the assessed results are chosen by a modified version of system pooling. Unfortunately, pooling inherently misses some relevant results because all results ranked below the pool depth are automatically regarded as irrelevant. However, Research by Zobel [25] concluded that while a system pool will only find about 70% of the relevant results in a collection the impact of system ranking was not significant. The second criticism is that a result's relevance is subjective to whoever is judging the result. Evidence has shown inconsistent judgements made between several people and even by the same person. However, research by Voorhees [24] concluded that although judges may disagree, it does not significantly affect system ranking.

However, the hierarchical nature of XML documents raises several issues in regards to XML-IR evaluation that are not found in the evaluation of traditional document retrieval systems. In particular, both the notion of relevance and the metrics used to evaluate the performance of systems are different. We will explore these differences in the next section.

2.2 Relevance in XML-IR

Central to the evaluation of IR systems is the (fuzzy) concept of relevance. There is no single and precise definition of relevance accepted by the IR community [5], even though there has been several attempts to define one [4,16,20]. Such a definition is beyond the scope of this paper. Therefore, we will use the same definition of relevance as is used during the INEX evaluation phase, where a result is judged relevant if it fulfils the information need of the user.

Key to this definition of relevance is end user relevance assessments. Like TREC, INEX uses an evaluation process similar to the one used during the Cranefield experiments. However, several changes were needed in order to adapt to the specific needs of XML-IR system evaluation. Theses changes relate to how judges assess the relevancy of individual results. Here we outline those changes and compare them with traditional document retrieval.

1. **First, relevance is judged over two dimensions: exhaustiveness and specificity**. Exhaustiveness measures how much a particular result fulfils the user's information need, that is, how thoroughly a particular result discusses the subject matter of the query. In comparison specificity measures the focus of a particular result, that is, how much relevant information in comparison with irrelevant information is contained in the element. In document retrieval, relevance tends to be judged on a single dimension.
2. **Second, relevancy is not independent between ancestors and descendants**. Imagine we have a result with the logical XPath *article/body/section/paragraph*. Logically each ancestor node contains all the information contained in each of its descendant nodes. So if a judge assessed the left element (paragraph*)* as relevant, then its parent node (section*)* contains at least the same information, then it must also be relevant. This logic traverses up the XML tree to the root node (article). INEX handles this constraint by enforcing that every relevant child node must have a relevant parent node, and consequently, that every relevant parent must have at

least one relevant child. In contrast, in traditional IR relevancy of results (documents) is said to be independent.

3. **Third, element relevance is non-binary**. As one propagates up an XML tree, the values for the two dimensions are bound to change. In general, ancestor nodes tend to be more exhaustive than descendants since they contain a larger amount of information. Conversely, descendant nodes tend to be more specific than ancestors since they contain less irrelevant information. Hence, in structured retrieval relevance needs to be evaluated on a graded rather than binary scale. In INEX, each dimension is judged as one of four values from zero to three where zero is judged as irrelevant. Also, an element cannot have a zero score in one dimension and a non-zero score in another. This produces nine possible levels of relevancy, plus a single non-relevant level. In contrast, most document-level evaluation methods tend to classify documents as either relevant or irrelevant.

2.3 The INEX Metrics

Once the results in the system pool are judged by the assessors. The assigned scores, along with a set of metrics, are used to evaluate the performance of retrieval systems. Most of the metrics used in XML-IR are based upon those already used in traditional IR, albeit slightly modified to handle the nature of XM-IR. Since the inception of INEX, several metrics have been either used or proposed, however, in this section we will only describe the INEX 2002 metric (also called the inex_eval metric), since it was the official metric used in INEX 2004. Discussion of other metrics will be saved until later in the paper.

2.3.1 INEX 2002 Metric. The INEX 2002 metric has been used since the inception of INEX. It is based upon Raghavan et al.'s [19] precall measure and calculates the probability that a result viewed by a user is relevant ($P(rel|retr)$):

(1) $$P(rel|retr)(x) = \frac{x \cdot n}{x \cdot n + esl_{x \cdot n}}$$

where *esl* denotes the expected search length [2],that is, the expected number of irrelevant results retrieved until an arbitrary recall point of *x* is achieved, and *n* is the total number of relevant results for a particular topic. A more detailed account of the metric can be found in the proceedings of first INEX workshop [9].

To apply the INEX 2002 metric the two dimensions must be mapped to a single relevance value by applying a quantisation function $\mathbf{f}_{quant}(e,s) : ES \longrightarrow [0,1]$. INEX currently employs a number of quantisation functions each representing a different user preference. Some of the quantisations score results on a purely binary scale, such as the *strict* metric, while others score results based on their degree of relevance such as the *generalised* and *specificity-orientated generalised* metrics. A detailed description of all the quantisation can be found in [13] and [7].

The main criticism of the INEX 2002 metric is that it ignores overlap. The next section describes overlap in more detail and outlines the reasons why some researchers perceive it as a problem.

3 The Problem with Overlap

3.1 Problem Description

Imagine the following results (all from the same document) are retrieved in response to a user query:

1. /article[1]/sec[1]
2. /article[1]/sec[1]/p[1]
3. /article[1]/sec[2]
4. /article[1]/sec[1]/p[2]
5. /article[1]
6. /article[1]/sec[2]/p[1]
7. /article[1]/sec[2]/p[2]

This list is a good example of overlap, where a result has a descendant or ancestor in the same list. Pehcecski *et al.* [17] identified two ways to measure overlap:

- *set-based overlap*, which for a set of results measures the percentage of results that have an ancestor in the set; and
- *list-based overlap*, which for a set of results measures the percentage of results for which there exists a higher ranked ancestor in the set.

Since the inception of INEX, overlap has been a controversial issue. Overlap has been criticised both from a user-orientated standpoint since it lacks a faithful end-user model, and from theoretical standpoint as it produces an over-populated recall base. Here, we summarise the previous research on this topic, outline some of the criticisms of overlap and present our responses.

3.1.1 Lack of Faithful End User Model. Due to the hierarchal nature of XML documents, relevance between nodes in a single document tree (XPath) is not independent. So, if a leaf is found relevant then so are all its ancestors, up to and including the root node. However, the INEX 2002 metric assumes result independence; thereby, rewarding each result solely by its quantised relevance value, regardless of if the results descendant or ancestor has already been rewarded. Hence, it is possible for the information contained in single leaf node to be rewarded multiple times.

There is no argument that overlap boosted the retrieval scores of systems, since most of the better performing systems at INEX have milked a lot of their results (for instance: in 2004 9 out of the top 10 systems in the CO task had overlap of over 70%). However, overlap alone does not guarantee that a retrieval system will achieve a high score, since it will only benefit systems that retrieve relevant results in the first place. So, just as a system will be rewarded multiple times for a relevant leaf node it will be penalised multiple times for retrieving irrelevant leaf nodes.

However, Trotman [22] argues that overlapping results are of no intrinsic value to users since it may not provide the users any new relevant information. This is a view supported by the experiments of INEX interactive track [15,21]. It is important to consider the needs of end-users when rewarding systems since the entire INEX methodology is based on the needs of end users. For example, INEX topics are based upon the type of queries commonly produced by end users. Secondly, during assessment judges are instructed to act as end users when evaluating the relevancy of results. To combat this problem some have proposed applying a user-based model into the evaluation of XML-IR systems.

However, a central problem with user-based XML-IR evaluation is that a proper understanding of the requirements of XML-IR users has not yet been formulated. Part of this problem has been a notable lack of research in XML-IR (and IR in general) devoted to user-based rather than laboratory-based (or batch) analysis. We already know that in document retrieval the results of user-based evaluation can differ greatly to the results of system-based evaluation [11, 23]. Therefore, one can argue that further research on the precise needs of XML-IR users should be conducted before a user-based model can fully be incorporated to the evaluation of XML-IR systems.

3.1.2 Overpopulated Recall Base. Recall-precision curves (RP curves) are a standard measure of IR system performance. RP curves record the precision (number if relevant results / number of retrieved results) of a retrieval system at various points of recall (for example: 10%, 20%,…,100%). For each recall point, a retrieval system's precision value is set to a normalised score between 0 and 1. In essence, the performance of retrieval systems are plotted against the performance of a perfect retrieval system (called the ideal recall base). In general, the ideal recall base should have a precision value of 1 for all recall points, however, this is not a situation that occurs within the INEX 2002 metric.

As identified by Kazai *et al.* [12] there are two reasons for this anomaly. The first is due to graded rather than binary relevance; however, the second reason is that the ideal recall base is 'over populated', meaning that a 'perfect' submission would have to return all relevant results, including those with overlap. Kazai *et al.* argue that this behaviour is contrary to the task of XML retrieval and has led to skewed INEX RP curves, with participants precision plotted at much lower values than the participants at other workshops (for example: TREC, CLEF).

We have already addressed the first of Kazai *et al.*'s criticisms, and while their second criticism is correct, it also warrants discussion. The discussion centres on the purpose of RP curves for comparing the performance of systems. It is important to understand that the scores presented in RP curves should be interpreted as relative rather than absolute values. We know that the values in RP curves can change depending on the topics, collection and judges used at the various stages of evaluation [24]. However, as long as the ranking of participants does not significantly change, then the

RP curves and the metrics that produced them are valid for system comparison. Likewise, while INEX RP curves may appear to be plotted at lower values than those using other datasets, as long as it does not affect the relative performance of systems then it is also a valid method for comparing system performance.

3.2 Previous Solutions

The present solution to deal with overlap in XML-IR has been the derivation of new metrics. These metrics, unlike the INEX 2002 metric, are designed to handle overlap in various ways, so that systems with low overlap are not disadvantaged. Here we describe these metrics, emphasising how each one handles overlapping results.

3.2.1 INEX 2003 Metric. The INEX 2003 metric (also called inex_ng) was the first metric designed to handle overlap by including both component size and overlap in the definitions of recall and precision [8]. The metric uses the total size of results as its basic parameter, as opposed to other metrics that measure the recall or precision after a certain number of results. Overlap is measured by considering the increment in text size of previously seen results. The metric has been criticised, first, because it does not address the problem of an overpopulated recall base, second, because it assumes that relevant information is distributed uniformly within a component and third, because it treats the two relevance dimensions in isolation, even though the task definition states that both are needed to properly identify the most appropriate unit of retrieval [13].

3.2.2 XCG Metric. The cumulated gain for XML (XCG) metrics [8] are an extension of the cumulative gain (CG) metrics proposed by Järvelin and Kekäläinen [10]. The CG metrics were designed to evaluate traditional information retrieval systems using multi-graded, rather than binary, relevance values. For each rank in a results list, the CG metric calculates the sum of relevance up to and including that rank of a results list. It also produces and ideal gain vector derived from the full non-overlapping recall base and sorted decreasingly by relevance. The normalised cumulated gain (nCG) measure is calculated by dividing the CG vector with the ideal gain vector. The XCG metrics extend the CG metrics by applying an INEX quantisation function [7] to the results lists and full recall base to derive the ideal gain vector. While the XCG managed to separate the user-behaviour model from the metric, the proper relevance value function is still open to debate and as are the interpretation of the curves after the actual and ideal CG curves meet.

3.2.3 PRUM Metric. The Precision-Recall with User Modelling[18] extends Raghavan's [19] probabilistic precision-recall metrics to include users' browsing behaviour. PRUM eliminates independence between components on the same XPath, by allowing users to consult the context (ancestors, descendants, siblings) of returned results. PRUM defines users' behaviour stochastically, by deriving the probability that a user has seen a particular element. For example: if it is known that a user has seen a parent result with probability 1, then the probability that the user has seen the result's first child is 0.95. The PRUM metric assumes an ideal results set and is defined as the probability that a user sees a newly relevant element when consulting the context of the retrieved element, while knowing that the user wants to see a given amount of relevant units.

3.2.4 T_2I Metric. The idea of the Tolerance to Irrelevance (T_2I) metric [6] is to provide the user with a set of entry points into a document that is close to relevant information. Starting at the entry point, the user reads (the portion of) the document until the predefined tolerance to irrelevance (number of words or sentences) has been reached, at which point the user moves onto the next result. T_2I rewards systems that return more specific elements since their entry point is more likely to be closer to relevant information than large elements. In fact, the entry point for some large element could be so far away from the relevant information that the tolerance to irrelevance is exhausted before the relevant information is reached. The problem of the overpopulated recall base is eliminated by extending the definition of relevance to include previously seen relevant results as irrelevant. T_2I variants of other existing evaluation metrics can be found in [6].

4 Our Solution

The central criticism of the INEX 2002 metric is that it ignores overlap by assuming independence between ancestor and descendant nodes. It has been conjectured that some systems exploit this feature by returning several nodes along an XPath, and hence being (unfairly?) rewarded multiple times for the same piece of information. The fact that the majority of the high scoring systems in INEX 2004 also had high overlap, while the poorer scoring system also had low overlap has been provided as evidence to support this conjecture. However, the following question remains unsolved: did these high overlap system score highly because of overlapped results, or because they had a better underlying retrieval algorithm that found more relevant results than the low scoring systems?

In some ways, INEX's use of the INEX 2002 metric has made answering this question difficult since systems it compares systems with high overlap to those with little or no overlap. The INEX 2002 metric is *overlap positive*, in that it inherently rewards systems that return relevant overlapping elements, thereby penalising non-overlap systems. Recently, researchers have tried to derive metrics that *handle* rather than *ignore* overlap. These new metrics are in effect, implicitly removing overlap from the submissions. However, it is not fully known if these metrics are *overlap neutral,* thereby not penalising nor rewarding overlap, or if they are *overlap negative*, in that they penalise overlap. Our approach is different. We explicitly remove overlap from the submissions before the evaluation phase. By explicitly removing overlap our approach is overlap neutral. We used two techniques to remove overlap:

- **Highest Rank (HR)**: where a result is removed from the list if it has a higher ranked *ancestor or descendant*; and
- **Leaves Only (LO):** where a result is removed from the list if it has a higher or lower ranked *descendant*.

So assuming that the following results list (introduced earlier and from the same document) was submitted:

1. /article[1]/sec[1]
2. /article[1]/sec[1]/p[1]
3. /article[1]/sec[2]
4. /article[1]/sec[1]/p[2]
5. /article[1]
6. /article[1]/sec[2]/p[1]
7. /article[1]/sec[2]/p[2]

Applying the Highest Rank technique would produce:

1. /article[1]/sec[1]
2. /article[1]/sec[2]

While applying the Leaves Only technique would produce:

1. /article[1]/sec[1]/p[1]
2. /article[1]/sec[1]/p[2]
3. /article[1]/sec[2]/p[1]
4. /article[1]/sec[2]/p[2]

We performed these two techniques on all the INEX 2004 submissions to produce two new sets of submissions. While previous work has been performed on removing overlap from single submissions [14], this is the first time that such an extensive investigation has been performed on a large amount of submissions. We evaluated the modified submissions using the recall-XCG metric using the standard parameters for focused retrieval (that is overlap = on). Then we ranked each of the modified submission sets by decreasing order of Mean Average Precision (MAP) to produce modified submission ranks. We also evaluated the original submissions using the recall-XCG metric using the standard parameters for thorough retrieval (that is overlap = off). While previous research has investigated the effect of evaluating INEX submissions using variations of the XCG metric[12], those submissions contained a mixture of high-overlap and low-overlapping system. This is the first time that submissions with zero overlap are compared with each other.

Our experiments allow us to explore two unanswered questions. First, what to what extent does the XCG metric penalise systems with overlapping results, thereby determining if the XCG metric is overlap negative or overlap neutral. And second, is there a strong correlation between the original and overlap removed submissions. If the two system ranks did not correlate then the high ranks achieved by the high scoring/overlap systems were likely due to unfairly rewarding overlapped results. Alternatively, if the systems ranks correlated then the high ranks achieved by the high scoring/overlap systems were likely due to them retrieving more relevant results. The next section presents the results of our experiments.

5 Results

5.1 Single Submission

Here we present the result of removing overlap from a single submission. First, we produced a CO and CAS submission from our current version of GPX that contained overlapping elements (i.e. a 2005 thorough submission) using the 2004 topic set. Then we applied both the overlap removal techniques introduced above to produce highest rank and leaves only versions of the submission. These algorithms removed overlapping elements from the submissions, thereby, decreasing their run length. Then we evaluated the submission using the recallXCG metric and sog quantisation, and recorded the MAP. Figures 1 and 2 present the ep/gr plots for our three submissions, while tables 1 presents their MAP.

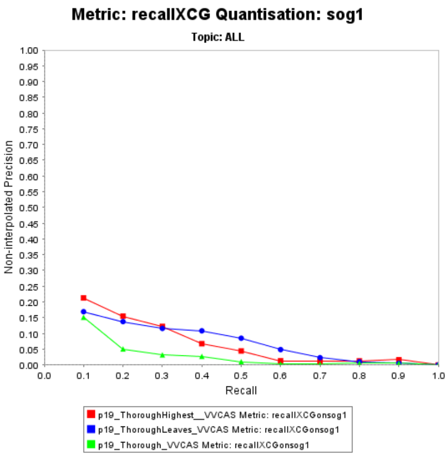


Fig. 1. CO SOG ER/GP Plot

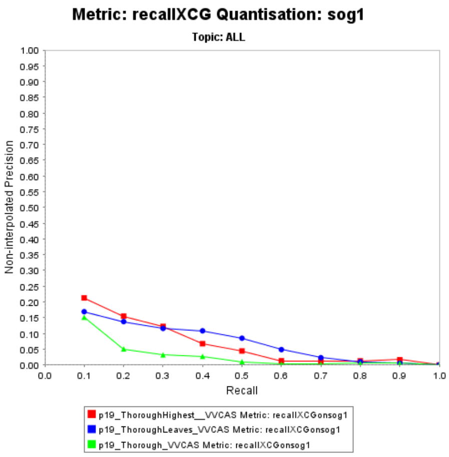


Fig. 2. VVCAS SOG ER/GP Plot

Table 1. MAP SOG for Thorough and Overlap Reduced Submissions

	CO	VCAS
Thorough	0.02393	0.02865
HighestRank	0.05358	0.06560
LeavesOnly	0.04319	0.07026

As the figures and table show, under the recall-XCG metric the thorough submission performs worse than the modified overlap removed submissions. This verifies that the XCG metrics are overlap negative, and justifies our use of the overlap removal algorithm as a precursor to systems rank comparison.

5.2 All Submissions

Here, we present the results of executing the overlap removal algorithms on all the INEX 2004 submissions. As before, we removed overlap from the submissions and then evaluated them with recall-XCG metric and sog quantisation. We also evaluated the original submissions with the overlap parameter set to off. We noticed however, that since more results would be removed from the systems with high original overlap then this technique could be biased towards the original low overlap systems. As a countermeasure, we executed the experiment multiple times using submissions with smaller result lengths. Figures 3 – 10 present the rank comparison between the overlap removed and original submissions at run lengths 100 and 1500. Tables 2 and 3 presents the rank correlations (Spearman-Rho and Kendall-Tau) between the overlap removed and original systems at various run lengths.

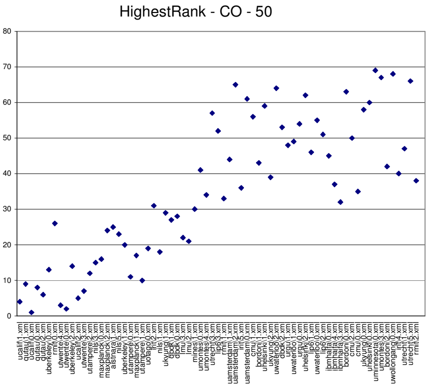


Fig. 3. HighestRank - CO - 50

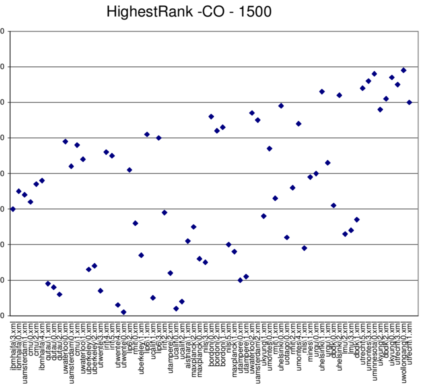


Fig. 4. HighestRank - CO - 1500

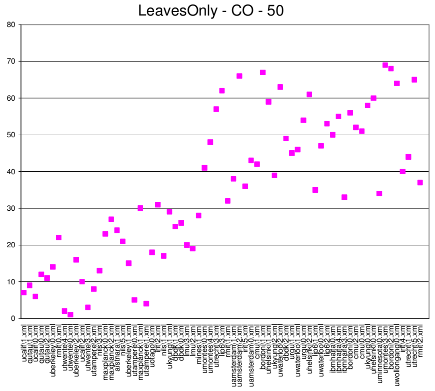


Fig. 5. LeavesOnly - CO - 50

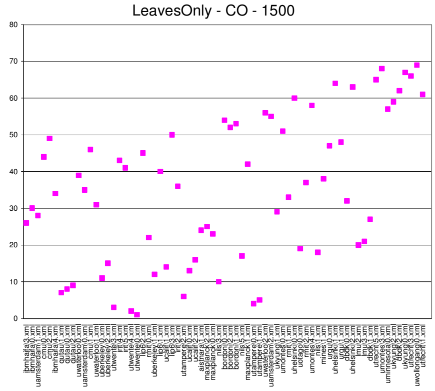


Fig. 6. LeavesOnly - CO - 1500

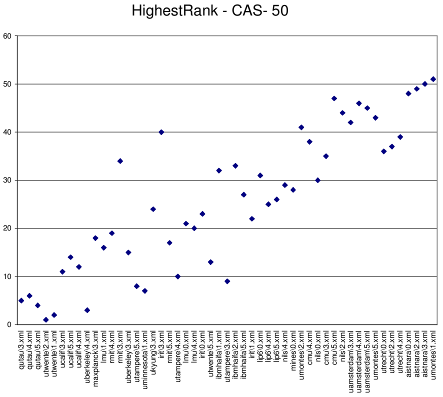


Fig. 7. HighestRank - CAS - 50

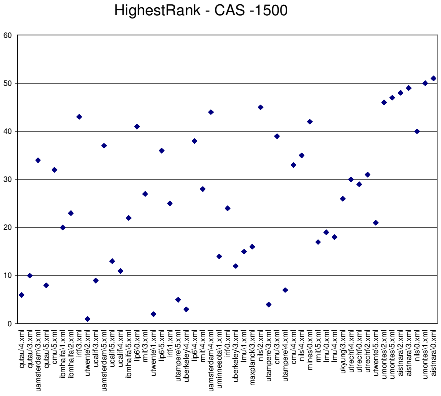


Fig. 8. HighestRank - CO - 1500

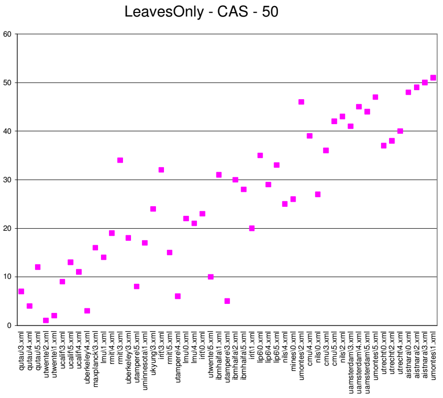


Fig. 9. LeavesOnly - CAS - 50

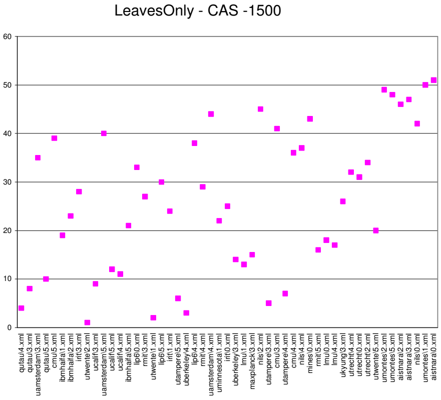


Fig. 10. LeavesOnly - CAS - 1500

Table 2. Correlation Scores HighestRank/Original

	50	**100**	**150**	**300**	**500**	**1000**	**1500**
CO (Kendall)	0.6863	0.6300	0.5499	0.7502	0.7485	0.7076	0.5012
CO (Spearman)	0.8210	0.8852	0.5843	0.8789	0.8638	0.8087	0.4761
CAS(Kendall)	0.7506	0.9624	0.6690	0.6580	0.7600	0.7020	0.5153
CAS(Spearman)	0.8766	1.0000	0.7302	0.7460	0.8545	0.7945	0.4686

Table 3. Correlation Scores LeavesOnly/Original

	50	**100**	**150**	**300**	**500**	**1000**	**1500**
CO (Kendall)	0.6658	0.7017	0.5302	0.7221	0.7221	0.7101	0.5277
CO (Spearman)	0.7762	0.9416	0.5265	0.8366	0.8270	0.7891	0.5085
CAS(Kendall)	0.7490	0.8133	0.6737	0.6675	0.7678	0.7145	0.5373
CAS(Spearman)	0.8690	0.9231	0.7271	0.7738	0.8657	0.8129	0.5085

As the figures and tables show there is little correlation between the overlap removed and original submission when run lengths of 1500 are considered. However, the curvature of the plots indicate a strong correlation between the MAP of the original and modified submissions at a lower result length. The correlation is not perfect, and some systems were disadvantaged by metrics that ignore overlap, however not as many systems were disadvantaged as originally thought. Furthermore, the correlation is strong enough to suggest that systems score well because they find more relevant results than low-scoring systems regardless of if the metric considers overlap.

6 Conclusion

The aim of this paper was to investigate the role of overlap in XML information retrieval. We dispute the existing conjecture that the high overlap/scoring systems in INEX 2004 performed strongly under metrics that reward well only because they were been reward multiple times for the same information. Rather, we show that the high overlap/scoring systems perform strong regardless of how overlap is handled.

References

1. Clarke, C. L. A., Tilker, P., L. (2005) : MultiText Experiments for INEX 2004, in Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl, Germany, December 6–8, 2004, Revised Selected Papers.
2. Cleverdon, C. and Keen, E. (1966): Aslib Cranfield Research Project: Factors determining the performance of indexing systems (Vol. 1: Design, Vol. 2: Results), Cranfield, England.
3. Cooper, W. S. (1968): Expected search length; a single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. American Documentation **19**(1):30-41.
4. Cooper, W. S. (1971): A definition of relevance for information retrieval. Information Storage and Retrieval **7**:19-37.
5. Crestani, F., Lalmas, M., Van Rijsbergen, R. J., and Campbell, T. (1998): Is this document relevant? Probably: A survey of probabilistic models in information Retrieval. ACM Computing Surveys, **30**(4).

6. de Vries, A. P., Kazai, G., Lalmas, M. (2004) : Tolerance to irrelevance : A user-effort orientated evaluation of retrieval systems without predefined retrieval unit. In Recherche d'Informations Assistee par Ordinateur (RIAO 2004), Avignon, France.
7. de Vries, A. P., Kazai, G., Lalmas, M. (2005) : Evaluation Metrics 2004, in Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl, Germany, December 6–8, 2004, Revised Selected Papers.
8. Gövert, N., Kazai, G., Fuhr, N., Lalmas, M. (2003): Evaluating the effectiveness of content-orientated XML Retrieval. Technical Report Computer Science 6, Technischer bericht, University of Drotmund.
9. Gövert N. and Kazai G (2003): Overview of the initiative for the evaluation of XML retrieval (INEX) 2002. In Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX), Dagstuhl, Germany, 1-15.
10. Järvelin, K., and Kekäläinen, J (2002): Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information System, **20**(4):551-556.
11. Hersh, W., Turpin, A., Price S., Chan, B., Kramer, D., Sacherek, L.Olson, D. (2000): Do batch and user evaluation give the same results? In Proceedings of The 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, Greece, **23**:17-23, ACM.
12. Kazai, G., Lalmas, M., de Vries, A. (2004): The overlap problem in content-orientated XML retrieval evaluation. In Proceedings of The 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, England, **27**:72-79, ACM.
13. Kazai, G. (2004): Report of the INEX 2003 metrics working group. In Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML Retrieval (INEX), Dagstuhl, Germany, 184-190, ERCIM Publications.
14. Kekäläinen, J., Junkkari, M., Arvola, P., and Alto, T. (2005): Trix 2004 – strugling with overlap. In Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl, Germany, December 6–8, 2004, Revised Selected Papers, 127-139, Berlin: Springer (LNCS ; 3493).
15. Kim, H. and Son, H. (2004): Interactive searching behavior with structured XML documents. In Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl, Germany, December 6–8, 2004, Revised Selected Papers, 424-436, Berlin: Springer (LNCS ; 3493).
16. Mizzaro, S. (1996): Relevance: The whole (hi)story,. Techical Report UDMI/12/96/RR (Dec.), Dipartimento di Matematica e Informatica, Universita' di Udine, Italy.
17. Pehcevski, J., Thom, J. Vercoustre,A-M (2005): User and Assessors in the Context of INEX: Are Relevance Dimensions Relevant? In Proceedings of INEX 2005 Workshop on Element Retrieval Methodology, Glasgow, Scotland, 42-57.
18. Piwowarski, B., and Gallinari, P. (2005): Precision recall with user modeling: application to XML retrieval. Submitted for publication.
19. Raghavan, V., Bollman, B. Jung G. (1989): A critical investigation of recall and precision. ACM Transaction on Information Systems, **7**(3):205-229.
20. Seracevic, T. (1970): The concept of "relevance" in information science: A historical review. In Introduction to Information Science. Chapter 14. SERACEVIC T. (ed), RR Bower Company, New York.
21. Tombros, A., Larsen, B., Malik, S. (2004): The interactive track at INEX 2004. In Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl, Germany, December 6–8, 2004, Revised Selected Papers, 410-423, Berlin: Springer (LNCS ; 3493).
22. Trotman, A. (2005): Wanted: Element Retrieval Users. In Proceedings of INEX 2005 Workshop on Element Retrieval Methodology, Glasgow, Scotland, 58-64.
23. Turpin, A. and Hersh, W. (2001): Why batch and user evaluation give the same results? In Proceedings of The 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, Greece, **24**:225-231, ACM.
24. Voorhees, E. M. (1998): Variations in relevance judgments and the measurement of retrieval effectiveness. Proceedings of The 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, **21**:315-323, ACM Press.
25. Zobel, J. (1998): How reliable are the results of large scale information retrieval experiments, Proceedings of The 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, ACM Press.

The Interpretation of CAS

Andrew Trotman[1] and Mounia Lalmas[2]

[1] Department of Computer Science, University of Otago, Dunedin, New Zealand
andrew@cs.otago.ac.nz,

[2] Department of Computer Science Queen Mary University of London, London, UK
mounia@dcs.qmul.ac.uk,

Abstract. There has been much debate over how to interpret the structure in queries that contain structural hints. At INEX 2003 and 2004, there were two interpretations: SCAS in which the user specified target element was interpreted strictly, and VCAS in which it was interpreted vaguely. But how many ways are there that the query could be interpreted? In the investigation at INEX 2005 (discussed herein) four different interpretations were proposed, and compared on the same queries. Those interpretations (SSCAS, SVCAS, VSCAS, and VVCAS) are the four interpretations possible by interpreting the target elements, and the support elements, either strictly or vaguely. An analysis of the submitted runs shows that those that share an interpretation of the target element correlate - that is, the previous decision to divide CAS into the SCAS and VCAS (as done at INEX 2003 and 2004) was sound. The analysis is supported by the fact that the best performing VSCAS run was submitted to the VVCAS task and the best performing SVCAS run was submitted to the SSCAS task.

1 Introduction

Does including a structural hint in a query make a precision difference and if so how should we interpret it? At INEX 2005 the *ad hoc* track has been investigating this question. Two experiments were conducted, the CO+S experiment, and the CAS experiment.

In the CO+S experiment the participants were asked to submit topics with content only (CO) queries containing just search terms, and optionally an additional structured (+S) query specified in the NEXI [10] query language. Given these two different interpretations of the same information need it is possible to compare the precision of queries containing structural hints to those that do not *for the same information need.* The details of the CO+S experiment are discussed elsewhere.

In a separate experiment participants were asked to submit topics containing queries that contain content and structure (CAS) constraints specified in NEXI [10]. These topics were used to determine how the structural hints, necessarily in a CAS topic, should be interpreted by a search engine. The two extreme views are the database view that all structural constraints must be upheld, and the information retrieval view that satisfying the information need is more important than following the query.

This contribution discusses the mechanics of the CAS experiment from the topic submission process, the document collection, through to the evaluation methods. The different tasks are compared using Pearson's product moment correlation coefficient showing that there were essentially only two tasks, those that in previous years have gone by the name VCAS and SCAS. Further analysis shows that of the tasks SSCAS is the easiest and VVCAS the hardest.

2 CAS Queries

Laboratory experiments in information retrieval following the Cranfield methodology (described by Voorhees [12]) require a document collection, a series of queries (known as topics), and a series of judgments (decisions as to which documents are relevant to which topics). In element retrieval this same process is followed - except with respect to a document element rather than a whole document.

Content and structure queries differ from content only queries in so far as they contain structural hints. Two types of structural hints are present, those that specify where to look (support elements) and those that specify what to return to the user (target elements). In INEX topic 258

```
//article[about(.,intellectual property)]//sec[about(., copyright law)]
```

the search engine is being asked to identify documents about intellectual property and from those extract sections about copyright law. The target element is //article//sec (extract //article//sec elements), and the support elements are //article for one clause (with support from //article about intellectual property) and //article//sec for the other (and support from //article//sec about copyright law). Full details of the syntax of CAS queries is given by Trotman and Sigurbjörnsson [10]. The applicability of this language to XML evaluation in the context of INEX is also discussed by Trotman and Sigurbjörnsson [11].

2.1 Query Complexity

The simplest CAS queries contain only a single structural constraint. Topic 270,

```
//article//sec[about( ., introduction information retrieval)]
```

asks for //article//sec elements about "introduction information retrieval". A more complex query can be decomposed into a series of single constraint queries (or child queries). Topic 258,

```
//article[about(.,intellectual property)]//sec[about(., copyright law)]
```

could be written as a series of single constraint queries, each of which must be satisfied. In this case it is decomposed into topic 259,

```
//article[about(.,intellectual property)]
```

and topic 281,

```
//article//sec[about(., copyright law)]
```

if both hold true of a document then the (parent) query is true of that document - and the target element constraints can be considered. The same decomposition property holds true for all multiple constraint CAS topics (so long as the target element is preserved) - it is inherent in the distributive nature of the query language.

Having separate parent and children topics it is possible to look at different interpretations of the same topic. As a topic is judged according to the narrative the judgments are by definition vague. Strict conformance of these judgments to the target element can be generated using a simple filter. This is the approach taken at INEX 2003 and 2004 for the so-called SCAS and VCAS tasks. But what about the sub-clauses of these topics? Should they be interpreted strictly or vaguely? With the judgments for the child topics, vague and strict conformance to these can also be determined. With the combination of child and parent judgments it is possible to look at many different interpretations of the same topic.

2.2 Topic format

INEX captures not only the query, but also the information need of the user. These are stored together in XML. Methods not dissimilar to this have been used at TREC [2] and INEX [1] for many years. As an example, INEX topic 258

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="258"  query_type="CAS" ct_no="72">
<InitialTopicStatement>
I have to give a computer science lesson on intellectual property
and I'm looking for information or examples on copyright law to
illustrate it. As I'm looking for something which is specific, I
don't think I can find a whole article about it. I'm consequently
looking for section elements.
</InitialTopicStatement>
<castitle>
//article[about(.,intellectual property)]//sec[about(., copyright law)]
</castitle>
<description>
Return sections about copyright law (information or examples) in an
article about intellectual property.
</description>
<narrative>
I have to give a computer science lesson on intellectual property,
and I'm looking for information or examples on copyright law to
```

```
illustrate it. More precisely, I'd like to have information about
authors rights and how to protect your creation. As I'm looking for
something which is specific, I don't think I can find a whole
article about it. I'm consequently looking for section elements.
Information or examples can concern copyright on software,
multimedia or operating systems. Copyright on literary work can help
but only for examples. Information concerning domain names and
trademarks is not relevant.
</narrative>
</inex_topic>
```

contains several parts all discussing the same information need:

- <**InitialTopicStatement**> a description of why the user has chosen to use a search engine, and what it is that the user hopes to achieve.
- <**castitle**> the CAS query specified in the NEXI language [10].
- <**description**> a natural language expression of the information need using the same terms as are found in the <**castitle**>. This element is used by the natural language track at INEX [13].
- <**narrative**> a description of the information need and what makes a result relevant. When judgments are made they are made against this description so it is important that it precisely describes the difference between relevant and irrelevant results. For experiments that additionally take into account the context of a query (such as the interactive track [8]), the purpose for which the information is needed (the work-task) is also given in the narrative.

Both the parent query and the child queries are stored in this way - but an additional element, the <**parent**> element, is present in child topics. This element stores the castitle of the child's parent. This method of linking children to parents was chosen over using identifiers as it was considered less likely to be prone to human input error.

2.3 Query Interpretation

A contentious point about CAS queries is how to interpret them. The database (strict) view is that the structural hints are constraints which must be followed exactly in order for a returned element to satisfy the query. The information retrieval (vague) view is that the structural hints are hints and can be ignored so long as a returned element is relevant in the mind of the user (it satisfies the information need).

A single clause query might be interpreted strictly, or vaguely - that is the constraint might be followed or can be ignored. If, for example, a user asks for an article abstract about "information retrieval", then perhaps an article introduction might just as well satisfy the need - or perhaps not.

With multiple clause queries, there are many possible interpretations. In the CAS experiment at INEX 2005, the strict and vague interpretations are applied to both the target element, and the support elements. This gives four interpretations written XYCAS where X is the target element and Y is the support element, and either X or Y can be S for strict or V for vague. Those interpretations are:

- **VVCAS:** The target element constraint is vague and the support element constraints are vague. This is the information retrieval view of the topic.
- **SVCAS:** The target element constraint is strict, but the support element constraints are vague.
- **VSCAS:** The target element constraint is vague, but the support element constraints are followed strictly.
- **SSCAS:** Both the target element constraint and the support element constraint are followed strictly. This is the database view.

3 Document Collection

The document collection used in the experiments was the INEX IEEE document collection version 1.8. This collection contains 16,819 documents taken from IEEE transactions and magazines published between 1995 and 2004. The total size of the source XML is 705MB. This is the latest version of the INEX collection at publication date.

4 Data Acquisition

This section discusses the acquisition of the queries from the participants and the verification that they are representative of previous years. It also discusses the acquisition of the judgments and the construction of the different judgment sets.

4.1 Query Acquisition

The document collection was distributed to the participating organizations. They were then each asked to submit one CAS topic along with any associated single clause child-topics (should they exist). These topics then went through a selection process in which queries were parsed for syntactic correctness, semantic correctness, consistency, and validated against their child topics. A total of 17 queries passed this selection process.

The breakdown of CAS topic complexity (excluding child-topics) for each of INEX 2003, 2004, and 2005 is given in Table 1. From visual inspection it can be seen that the breakdown in 2005 is representative of previous years, most queries contain two clauses with approximately the same number of three and one clause topics. In 2005 there were no topics with more than 3 clauses.

Table 1. A Breakdown of the complexity of INEX 2005 CAS topics shows that they are representative of previous years

Clauses	1	2	3	4+
2003	7	12	6	5
2004	4	22	4	4
2005	3	12	2	0

Table 2. The 17 topics and the topic numbers of their children

Parent	Children	Parent	Children	Parent	Children
244	245, 246	258	259, 281	270	
247	248, 249, 276	260		275	274, 273
250	251, 252	261	262, 263	280	277, 278, 279
253	254, 255	264	282, 283	284	266, 285
256	272, 271	265	267, 268	288	242, 243
257		269	286, 287		

4.2 Child Topics

Each topic and child topic was given a unique identifier (stored in the topic_id attribute of the inex_topic tag). Table 2 shows which topics are parent topics and which topics are their children. Topic 258, for example, has topics 259 and 281 as children whereas topic 260 is a single clause query and has no children.

It may appear at the onset that these child topics can be used as part of the evaluation giving a total of 47 topics. This, however, is not the case. The guidelines for topic development [7] identifies that for evaluation purposes queries must be diverse, independent, and representative. Using both the parent and the children topics for computing performance violates the independence requirement - and weights evaluation in favor of longer topics (which have more children).

Using just the child topics, and discarding the parents, violates the requirement that topics are representative. In Table 1, the breakdown of topics from previous years is shown. Most topics have two clauses, whereas child topics (by definition) have only one. The children, without their parents, are not representative.

4.3 Judgment Acquisition

The topics and child-topics were distributed to the participants. Each participating group was invited to submit up to two runs for each CAS task. At least one was required for VVCAS. A run consisted of at most 1,500 ranked results for each parent and child topic. There were no restrictions on which part of the topic was used to generate the query - participants were permitted to use the narrative, or description, or castitle if they so chose.

These results were then pooled in a similar manner to that used at TREC (and shown to be robust there by Zobel [15]). The details of the INEX pooling method are give by Piwowarski and Lalmas [6] and a discussion of the robustness is provided by Woodley and Geva [14].

The pool identifies which documents and elements the search engines considered relevant to the query. Using a graphical interface (the 2005 version of X-Rai [5, 6]) to the document collection, the original author of the query (where possible) was asked to identify which elements of which documents in the judgment pool were, in fact, relevant to the information need. Assessors first highlighted relevant passages from the text, and then they assigned relevance values to all elements in this region on a three points scale: highly exhaustive, partly exhaustive, or too small. This assessment was performed for the parent topics in isolation of the child topics - and not necessarily by the same assessor.

As a topic may contain many different interpretations of the information need (for example the description and the castitle) all judgments were made with reference to the description contained in the topic narrative.

4.4 CAS Relevance Assessments

Table 3. Topics assessed by more than one assessor, and which pool was assigned to which assessment set

Topic	**Pool**	
	Set-a (official)	**Set-b (other)**
261	350	362
244	354	358
250	356	369
258	289	360

In a separate experiment the consistency of the judgments is being measured across multiple assessors. This is done by asking two or more judges to assess the same topic, without knowledge of the other's decisions. Of the CAS topics, those listed in Table 3 were multiple-judged.

The consequence of this multiple assessment process is that there is no single set of relevance assessments. Inline with INEX 2004, the assessments are divided into two groups: set-a, and set-b (see Pehcevski *et al.* [4] and Trotman [9] for a discussion of the 2004 results of this experiment). The INEX 2005 assignment was made based on proportion of completion at the date the first relevance assessments were released. Those judgments that, from visual inspection, appeared most complete were assigned to set-a, while the other was assigned to set-b. In this way set-a, the set used to generate the official results, was most complete and therefore most reliable.

Internal to X-Rai (the online assessment tool), each assessment of each topic by each judge is given an internal identifier - the pool id. Table 3 also shows which pool ids were assigned to which judgment set.

4.5 CAS Relevance Sets

From set-a, four sets of judgments were generated, one for each of the four CAS interpretations - each derived from the same initial set of judgments.

- **VVCAS:** The assessments as done by the assessors (against the narrative).
- **SVCAS:** Those VVCAS judgments that strictly satisfy the target element constraint. This set of judgments was computed by taking the VVCAS judgments and removing all judgments that did not satisfy the target element constraint. This was done by a simple matching process in all except topic 260 in which the target element is specified as //bdy//*. In this case all descendants of //bdy (excluding //bdy) are target elements.
- **VSCAS:** A relevant element is not required to satisfy the target constraint, however the document must satisfy all other constraints specified in the query. In all except two cases, this constraint is that for a judgment of the parent topic to be relevant, it must come from a document that also has SVCAS judgments for all its children. In one exception (topic 247), this conjunction is replaced with a disjunction. In the other exception (topic 250) there are (presently) no judgments as the assessment task has not been completed.
- **SSCAS:** Those VSCAS judgments that satisfy the target element constraint. These are computed from the VSCAS judgments in the same way that SVCAS judgments are computed from VVCAS judgments - strict conformance to the target element.

The guidelines for topic development [7] identify groups of tags that are equivalent. For example, for historic paper publishing reasons the sec, ss1, ss2 and ss3 tags are all used to identify sections of documents in the collection. The strict conformance to a given structural constraint occurs with reference to the equivalence list - //article//bdy//ss1 strictly conforms to //article//sec.

5 Measurement

The official metric used to report the performance of a system at INEX 2005 is MAep, the mean average nxCG rank at 1500 elements. This measure is described by Kazai and Lalmas [3]. The results (produced using xcgeval) for the INEX 2005 CAS task are available from INEX. There were 99 runs submitted to the CAS tasks, of which 25 were SSCAS, 23 SVCAS, 23 VSCAS, and 28 VVCAS[1].

Of the 17 topics used for evaluation (the parent topics of Table 2) judgments currently exist for only 10 topics - at the time of writing the assessment task

[1] Submissons version 1 and judgments version 7 are used throughout

had not been completed for the other 7 topics. Of those 10 topics, only 7 have any elements that strictly conform to their child topic structural constraint. The comparison of systems herein is based only on these topics.

Table 4. Number of relevant elements for each topic using generalised quantization

Topic	SSCAS	SVCAS	VSCAS	VVCAS
253	0	23	0	156
256	492	724	1431	2101
257	96	96	711	711
260	5159	5159	5264	5264
261	0	59	0	4437
264	6	40	155	1272
265	0	40	0	211
270	35	35	850	850
275	111	183	12870	16965
284	2	111	326	14265

Table 5. Number of relevant elements for each topic using strict quantization

Topic	SSCAS	SVCAS	VSCAS	VVCAS
253	0	0	0	11
256	139	162	198	228
257	0	0	0	0
260	66	66	66	66
261	0	0	0	2
264	0	0	12	44
265	0	0	0	1
270	1	1	2	2
275	18	22	330	424
284	0	5	4	196

In Table 4 and Table 5 the number of relevant element for each topic of each task is shown. The judgments for strict quantization are highly sparse - for the SSCAS task, there are only 4 topics with highly specific and highly exhaustive judgments. It does not seem reasonable to draw any conclusions from only 4 topics so the remainder of this analysis applies to only the generalized quantization of results.

By correlating the results of one task with those of another (say, VVCAS with SSCAS), it is possible to see how well a system designed to target one interpretation performs when evaluated using a different interpretation. This is the case when a search engine is designed to answer in one way, but the user expects results in another. Taking all the CAS runs (including the “unofficial” runs) the IBM Haifa Research Lab run VVCAS_no_phrase_no_tags submitted to the VVCAS task performs best using the VVCAS judgments (with a MAep score of 0.1314), but if the user need included a strict interpretation of the topic (it was evaluated using the SSCAS judgments) then it is at position 50 with a score of 0.0681.

By comparing the performance of runs submitted to each task it is possible to determine if one task is inherently easier, or harder, than the others. With a harder task there is more room for improvement - further investigation into this task might result in improvements all-round.

5.1 Do the Judgment Sets Correlate?

Table 6. Pearson’s product moment correlation coefficient between each CAS task

	SSCAS	**SVCAS**	**VSCAS**	**VVCAS**
SSCAS	1.0000	0.8934	0.4033	0.3803
SVCAS	0.8934	1.0000	0.3409	0.3768
VSCAS	0.4033	0.3409	1.0000	0.9611
VVCAS	0.3803	0.3768	0.9611	1.0000

Table 6 shows the Pearson’s product moment correlation coefficient computed for all runs when scored at each task. Scores close to 1 show a positive correlation, those close to -1 a negative correlation and those at 0 show no correlation.

It is clear from the table that VVCAS and VSCAS are strongly correlated. A search strategy that performs well at one task performs well at the other. SSCAS and SVCAS, both with a strict interpretation of the target element are less strongly correlated. There is little correlation between a strict interpretation of the target element and a vague interpretation of the target element (SVCAS and VSCAS, for example).

Figure 1 shows this correlation for the vague target element tasks. There is a cluster of best-scoring runs at the top-right of the graph. They are runs that have performed well at both VVCAS and VSCAS. These four runs are those from IBM Haifa Research Lab. Although different runs perform best on the VVCAS and VSCAS task, both “best” runs were submitted to the VVCAS task - providing further evidence of the correlation of the two tasks.

Figure 2 shows the same for the strict target element tasks. The cluster is not seen. The best performing run measuring on the SVCAS task was submitted to the SSCAS task (again IBM Haifa Research Lab). These same runs were only bettered by the four from the University of Tampere when measured for the SSCAS task. Although Tampere produced runs that performed well at the SSCAS task and not at the SVCAS task, IBM Haifa Research Lab produced runs that performed well at both tasks. Again further evidence of the correlation of the two tasks.

Figure 3 shows the performance of SSCAS against VVCAS. It is clear from this figure that those runs that perform well at one task do not perform well at the other. It appears, from visual inspection, that they are average performers at each other's tasks.

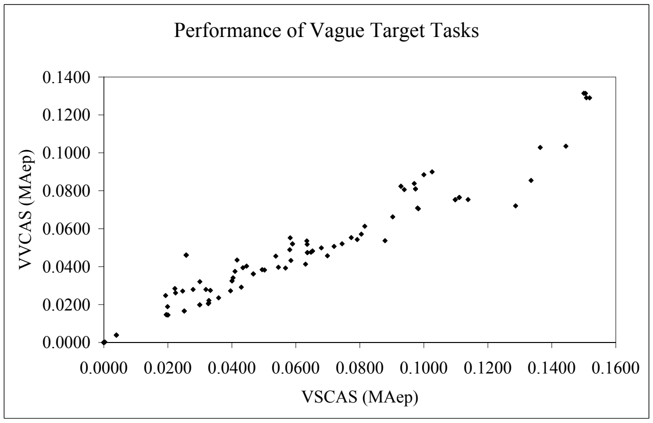


Fig. 1. Plot of performance of all submitted runs using VVCAS and VSCAS shows a strong correlation of one to the other

5.2 Tasks Complexity

For each task the mean performance (MAep) of the best 21 runs submitted to that task was additionally computed. This number was chosen because different numbers of runs were submitted to each task, and for all tasks there were at least 21 runs with a non-zero score. The mean and best performances are shown in Table 7 where it can be seen that as far as the runs are concerned, SSCAS is easier than SVCAS, which is easier then VVCAS, and the hardest is VSCAS. The SSCAS task may be easiest because the required structural constraints are specified explicitly in the query and the search engine can use this as a filter to remove known non-relevant elements from the result list.

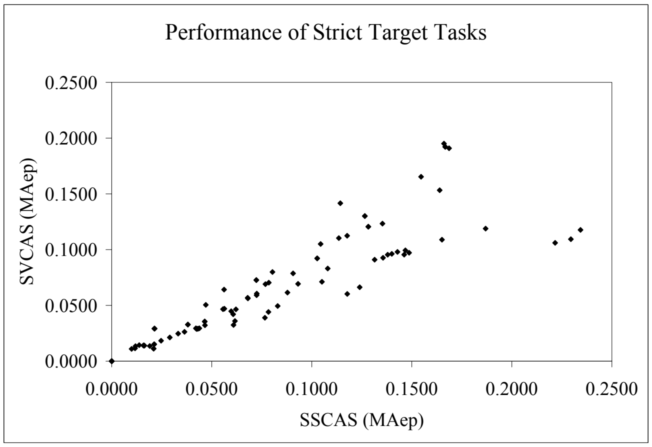


Fig. 2. Plot of performance of all submitted runs using SVCAS and SSCAS shows a strong correlation of one to the other

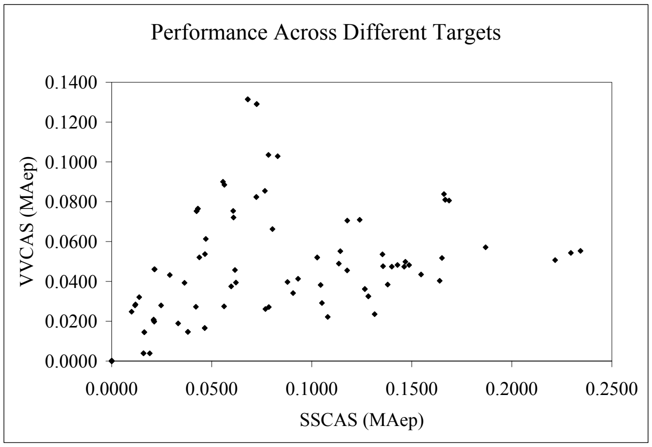


Fig. 3. Plot of performance of all submitted runs using VVCAS and SSCAS shows little correlation of one to the other

Table 7. Mean performance of top 21 runs from each task

	SSCAS	SVCAS	VSCAS	VVCAS
Mean	0.1285	0.0832	0.0628	0.0690
Std Dev	0.0510	0.0484	0.0439	0.0310
Best	0.2343	0.1922	0.1508	0.1314
Worst	0.0381	0.0292	0.0039	0.0208

6 Conclusions

The Pearson's correlation shows that there are only two different interpretations of the query, those with a strict interpretation of the target element and those with a vague interpretation of the target element (the database and the information retrieval views). It is possible to ignore the interpretation of the child elements and concentrate on only the target elements. In previous years, INEX has made a distinction between strict and vague conformance to the target element, but has disregarded conformance to child constraints (the so-called SCAS and VCAS tasks). This finding suggests the experiments of previous years did, indeed, make the correct distinction. Checking child constraints does not appear worthwhile from an evaluation perspective.

The vague task has proven more difficult than the strict task. Strict conformance to the target element can be computed as a filter of a vague run - from those vague elements, remove all that do not conform to the target element constraint. The vague interpretation of CAS is a better place to concentrate research effort.

If the CAS task continues in future years, a single set of topics, without the child topics is all that is necessary for evaluation and participants should concentrate on the vague interpretation of topics.

References

1. Fuhr, N., Gövert, N., Kazai, G., and Lalmas, M. (2002). INEX: Initiative for the evaluation of XML retrieval. In *Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*.
2. Harman, D. (1993). Overview of the first TREC conference. In *Proceedings of the 16th ACM SIGIR Conference on Information Retrieval*, (pp. 36-47).
3. Kazai, G., and Lalmas, M. (2005). INEX 2005 evaluation metrics. In *Proceedings of INEX 2006*.
4. Pehcevski, J., Thom, J. A., and Vercoustre, A.-M. (2005). Users and assessors in the context of INEX: Are relevance dimensions relevant? In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, Second Edition*, (pp. 47-62).
5. Piwowarski, B., and Lalmas, M. (2004). Interface pour l'évaluation de systèmes de recherche sur des documents XML. In *Proceedings of the Premiere COnference en Recherche d'Information et Applications (CORIA'04)*.

6. Piwowarski, B., and Lalmas, M. (2004). Providing consistent and exhaustive relevance assessments for XML retrieval evaluation. In *Proceedings of the 13th ACM conference on Information and knowledge management*, (pp. 361-370).
7. Sigurbjörnsson, B., Trotman, A., Geva, S., Lalmas, M., Larsen, B., and Malik, S. (2005). INEX 2005 guidelines for topic development. In *Proceedings of INEX 2005.*
8. Tombros, A., Larsen, B., and Malik, S. (2004). The interactive track at INEX 2004. In *Proceedings of INEX 2004*, (pp. 410-423).
9. Trotman, A. (2005). Wanted: Element retrieval users. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, Second Edition*, (pp. 63-69).
10. Trotman, A., and Sigurbjörnsson, B. (2004). Narrowed Extended XPath I (NEXI). In *Proceedings of INEX 2004*, (pp. 16-40).
11. Trotman, A., and Sigurbjörnsson, B. (2004). NEXI, now and next. In *Proceedings of INEX 2004*, (pp. 41-53).
12. Voorhees, E. M. (2001). The philosophy of information retrieval evaluation. In *Proceedings of the The Second Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems*, (pp. 355-370).
13. Woodley, A., and Geva, S. (2004). NLPX at INEX 2004. In *Proceedings of INEX 2004*, (pp. 382-394).
14. Woodley, A., and Geva, S. (2005). Fine tuning INEX. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, Second Edition*, (pp. 70-79).
15. Zobel, J. (1998). How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st ACM SIGIR Conference on Information Retrieval*, (pp. 307-314).

TIJAH Scratches INEX 2005 Vague Element Selection, Overlap, Image Search, Relevance Feedback, and Users

Vojkan Mihajlović[1], Georgina Ramírez[2], Thijs Westerveld[2], Djoerd Hiemstra[1],
Henk Ernst Blok[1], and Arjen P. de Vries[2]

[1] University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands
{v.mihajlovic, d.hiemstra, h.e.blok}@utwente.nl
[2] Centre for Mathematics and Computer Science,
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
{georgina, thijs, arjen}@cwi.nl

Abstract. Retrieving information from heterogeneous data sources in a flexible manner and within single (database) framework is still a challenge for many data retrieval systems. In this paper we present the extension of our prototype database system TIJAH developed for structured retrieval. The extension is aimed at modeling vague selection of XML elements and image retrieval. All three levels (conceptual, logical, and physical) of the TIJAH system are enhanced to support these new concepts. Additionally, we analyze different ways of removing overlap, explain how structural information can be used for relevance feedback, and investigate what real users want from structured documents.

1 Introduction

In this paper we discuss our participation at INEX 2005 with TIJAH, a three-level database system for structured information retrieval. The TIJAH [12–14] is developed as a transparent XML-IR database system consisting of conceptual, logical, and physical levels. TIJAH can handle queries with the strict selection of XML elements, specified in the NEXI query language [20] and to reason about textual information. This year we extended it in two directions: towards handling vague specification of XML elements in the query (similar to [5]), and towards supporting retrieval from heterogeneous domains (images and videos), following the guidelines from multimedia retrieval database systems [3]. Moreover, we analyze the performance of different approaches to remove overlap, continue with the relevance feedback experiments [17] using the TIJAH system, and study real users' information needs when searching structured documents.

The first point that we want to address in this paper is handling imprecise specification of elements in the XML search. Similarly as user gives only a number of terms as hints for searching within a document, XML elements specified within the query need not be considered as a strict requirement but as a hint for structural search. Therefore, when formulating a query the user can state that the search (support) element or answer (target) element should be treated as a hint or as a constraint in the retrieval process. To support this vague search we introduced vague element search as a concept in our TIJAH system.

On the other hand, to cope with the heterogeneous data sources (images and videos) the TIJAH system is extended with new features on each level that can express image search. The image search is handled in the same framework as the text search at the conceptual level where additional syntactical specification is added to the NEXI query language, and at logical level where new operators are introduced in the Score Region Algebra (SRA) [13]. However, due to different nature of the domain data, images are stored and handled in a different manner than textual XML data at the physical level.

We also present our approaches for removing overlap and relevance feedback. To remove overlap, we define an *utility* function that intends to capture the amount of *useful* information each element contains. We use this function to decide which is the most appropriate retrieval unit for each of the parent-children relationships. Our relevance feedback approach uses the structural characteristics of the relevant elements to update the priors in a language modeling framework.

Finally, our participation in the interactive track aims at studying and classifying different types of user needs when searching structured documents.

1.1 Overview

The following section explains the extensions introduced in the TIJAH system to model vague XML element specification. Section 3 details our approach for image retrieval. The relevance feedback and overlap approaches are discussed in Section 4 and 5 respectively while the intended user studies are explained in Section 6. We wrap-up the paper with the results from the numerous experiments performed for each track and its sub-tasks in Section 7 and with conclusions and future directions in Section 8.

2 Vague Node Selection

This section details the motivation and the implementation of vague selection of nodes in our three-level database framework. We explain the extensions on each level aimed for vague search on elements.

2.1 Vague element node selection in NEXI

Instead of extending our conceptual parser for rewriting content-and-structure (CAS) queries into SVCAS, VSCAS, and VVCAS (SSCAS is equal to CAS in our case), where prefix 'S' or 'V' stands for vague specification of target and support elements, we decided to extend the NEXI grammar with one extra symbol '$\sim$'. The 'tilde' symbol is used in front of the element name in the query specification, denoting that the element name does not have to be strictly matched in the query evaluation. We support this decision by arguing that the user should be responsible for stating his confidence in the knowledge of the hierarchical organization of the data he is querying, or whether he is certain or not what is the element name in which he wants to search for information.

The vague element selection can be treated similarly as a query expansion on terms in traditional IR. For example, if a user searches for the term 'conclusion', he might also be satisfied with terms 'decision', 'determination', 'termination', or 'ending' in the answer. In structured documents, if a user asks for 'car' elements, he would probably not mind getting 'auto' or 'vehicle' elements as an answer. Furthermore, he might also agree with the answers: 'van', 'sports-car', or 'convertible'.

While the list of possible synonyms, hypernyms, and hyponyms for terms can be considered as relatively static over time (e.g., WordNet [15]) and the degree of similarity can be pre-specified, in the case of element name expansion the problem is more complex and dynamic. Besides the terms that have the same or similar meaning, like the ones given above, it can happen that element names follow different naming pattern. Thus, elements might have complex element names such as: 'sport_car', 'sport-car', 'vehicles_list', etc.. Abbreviations could also be used, such as for section elements in INEX IEEE collection: 'sec', 'ss1', 'ss2', 'ss3'. Additionally, if a user asks for elements denoting one concept it might not be wrong if the answer is an element from a similar concept. Plenty of such examples can be identified in INEX; e.g., if a user asks for sections, he might be satisfied with paragraphs (see the extensive list of equivalence classes in [11]), abstracts, or even short articles (summaries). Furthermore, the list of element names can be larger in semantically richer and heterogeneous XML collection and it can evolve over time with the introduction of new XML collections.

The problem of element name matching is studied in the research area of schema matching and numerous techniques exist that try to resolve this problem (see [4, 16] for survey). However, we decided to simplify the vague element name search task and use the results from INEX 2004 assessments to find the expanded element names. We define the list of expanded element names based on the list of element names assessed as highly exhaustive elements in INEX 2004 assessments process. The lists are given in Table 1 and we term these lists *element name expansion lists*[3].

2.2 Introducing complex selection operator for vague node selection

The vague node selection at the conceptual level (NEXI) is translated into complex vague node selection operator at the logical level. However, the vague node selection operator in score region algebra has more expressive power than the simple NEXI extension on the conceptual level. It allows much finer specification of search and answer elements than a simple vague '$\sim$' node name specification. The vague node selection operator in SRA is defined as a union of all XML

[3] Other elements in INEX IEEE collection are not included in Table 1 as they were not present as target elements in the 2004 topic set.

Table 1. Element name expansion list based on INEX 2004 assessments.

Element name	**Expanded element names**
`abs`	`abs, fm, kwd, vt, p, sec, article, bdy, ref`
`article`	`article, bdy, sec, abs, fm, bm, bib, bibl, bb, p, ref`
`atl`	`atl, st, fgc`
`bb`	`bb, bm, bibl, bib, atl, art`
`bdy`	`bdy, article, sec, abs, p, ref`
`bib`	`bib, bm, bb, atl, art`
`fig`	`fig, sec, st, p, fgc, st, atl`
`fm`	`fm, sec, abs, kwd, vt, p, article, bdy, ref`
`kwd`	`kwd, abs, fm, st, fgc, atl`
`p`	`p, vt, abs, sec, fm, article, bdy, st`
`sec`	`sec, abs, fm, vt, p, article, bdy, bm, app`
`st`	`st, atl, fgc`
`tig`	`tig, bb`
`vt`	`vt, p, sec, bm, fig`

element regions that match the names of the 'expanded name regions' within the element name expansion list. By default all 'expanded regions' are down-weighted by a predefined factor. The definition of the operator is as follows:

$$\sigma^{expansion(class)}_{n=name,t=node}(R_1) := \{r|r_1 \in R_1 \ \wedge \ (s,e,n,t) := (s_1,e_1,n_1,t_1) \ \wedge \ t = node \\ \wedge \ (n,p) \in expansion(class,name)\} \tag{1}$$

Here $expansion(class)$ is a set that contains all the expansions for all the region names in one expansion class, where expansion list for each region $name$ is denoted as $expansion(class, name)$ (with cardinality n):

$$expansion(class, name) := \{(ex\_n_1, ex\_w_1), (ex\_n_2, ex\_w_2), ..., (ex\_n_n, ex\_w_n)\}$$

Here ex_n_i is a expanded element name and ex_w_i is a real number in the range $[0,1]$ denoting the down-weight factor. The operator $\sigma^{expansion(class)}_{n=name,t=node}(R_1)$ assigns name (ex_n) and score (ex_w) values to the region name (n) and score (p) based on the name and score values in the expansion list $expansion(class, name)$.

The simple selection operator in basic score region algebra operator set (see [14]) can be considered as a complex selection operator where $expansion(class)$ set is empty. Note that the complex selection operator can also be expressed using the basic SRA selection operator and scaling operator as follows:

$$\sigma^{expansion(class)}_{n=name,t=node}(R) := \bigsqcup_{p \atop (ex\_n_i,\, ex\_w_i) \,\in\, expansion(class,\, name)} (\sigma_{n=ex\_n_i,t=node}(R) \circledast ex\_w_i) \tag{2}$$

We defined two expansion classes for our INEX 2005 participation: (1) based on the equivalence classes defined for INEX IEEE collection [11], termed eq_class, and (2) based on a fusion of equivalence classes and our INEX 2004 expansion element name lists given in Table 1 such that every expanded element name in this table that has the equivalent name in the *eq_class name* part is also expanded with the *eq_class* equivalent names for *name*, termed *uni_class*. Therefore, the *eq_class* selection on section elements can be expressed as $\sigma^{expansion(eq\_class)}_{n=`sec',t=node}(R)$, and vague node selection ∼`sec` can be transformed into the next SRA operation $\sigma^{expansion(uni\_class)}_{n=`sec',t=node}(R)$. In such a way we can transparently define the set of expanded nodes and their respective weights and use them for vague node selection in a vague element name selection retrieval scenarios.

2.3 The implementation of vague selection operator

On the physical level, since we are working with the known INEX IEEE data collection, and as we used static INEX equivalence element name list and expansion element name list based on INEX 2004 assessments, we decided to replicate the two lists and store them as tables at the physical level, i.e., in MonetDB [1]. Thus, we have two tables with *(entity_name, expansion_name, expansion_weight)* for *uni_class* and *(entity_name, equivalent_name)*[4] for *eq_class* sets. The complex selection operator is then implemented as an additional MIL (MonetDB Interpreter Language) function, based on the definition given in the previous section, that uses the information from these tables.

For example, the vague *name* selection operator on region table R and the 'expansion regions' table S for the *uni_class* element names, in relational algebra could be defined as:

$$\pi_{r.s,r.e,r.n,r.t,s.weight}(\sigma_{s.n=name}(S) \bowtie_{s.n=r.n} (\sigma_{r.t=node}(R)))$$

2.4 Retrieval models

We based the instantiation of retrieval models on the best models used for flat-file information retrieval, as well as XML retrieval: language models [7], Okapi (INQUERY) model [2, 18], and Garden Point XML (GPX) [6]. Based on experimental results in [13] we instantiated the following functions for assigning score values to regions in SRA. For the relevance score computation on regions we used: Equation 3 for language models, Equation 4 for Okapi, and Equation 5 for GPX.

$$f^{\mathrm{LM}}_{\sqsupset}(r_1, R_2) = p_1 \cdot (\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{size(r_1)} + (1-\lambda)\frac{|R_2|}{size(Root)}) \tag{3}$$

$$f^{\mathrm{Okapi}}_{\sqsupset}(r_1, R_2) = p_1 \cdot ln \frac{|\{r \in C | n = n_1\}| - |\{r \in C | n = n_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}| + 0.5}{|\{r \in C | n = n_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}| + 0.5}$$
$$\cdot \frac{(k_1+1) \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{k_1((1-b) + b\frac{size(r_1)}{avg\_size(n_1)}) + \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2} \tag{4}$$

[4] In the preliminary experiments we did not store weights for equivalent region names as we assumed that their default weight is 1.0.

$$f_{\sqsupset}^{\mathrm{GPX}}(r_1, R_2) = p_1 \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{|R_2|} \tag{5}$$

For upwards score propagation we used either weighted sum (Equation 6) or sum (Equation 7), while for downwards score propagation we employed Equation 8.

$$f_{\blacktriangleright}(r_1, R_2) = p_1 \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2 \cdot size(r_2)}{size(r_1)} \tag{6}$$

$$f_{\blacktriangleright}(r_1, R_2) = p_1 \cdot \sum_{r_2 \in R_2 | r_1 \prec r_2} p_2 \tag{7}$$

$$f_{\blacktriangleleft}(r_1, R_2) = p_1 \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2 \tag{8}$$

Abstract operators $\otimes$ and $\oplus$ are implemented both as simple sum, or as product and sum, except in the case of GPX model where the instantiation is given in Equation 9.

$$p_1 \otimes p_2 = p_1 \cdot p_2 = \begin{cases} p_1 + p_2 & \text{if } p_1 = 0 \vee p_2 = 0 \\ \\ A \cdot (p_1 + p_2) & \text{otherwise} \end{cases} \tag{9}$$

3 Image similarity search

To enable search on multimedia collection (provided by Lonely Planet) we also introduced extensions to the TIJAH system defined along three levels of our prototype DB.

3.1 Extending the NEXI syntax for image search

To include the image search in TIJAH we extended the TIJAH system with the *about image*. The NEXI syntax is extended with an extra token '`src:`' that defines the location of the source image with which the destination image should be matched. Therefore, in the multimedia query 1:

```
//destination[about(.//images//image, buddha src:/images/BN417_16.jpg)]
    //*[(about(., asia) or about(., asian))
         and (about(., buddha) or about(., buddhist))]
```

the first *about* contains a request for image similarity search. The destination image that need to be matched is located under the local `images` directory with the name of `BN417_16.jpg`. In the preprocessing step, the '`src:`' part of the *about* is transformed into *about_image* and its relative path given in the NEXI '`src:`' specification is resolved into the path to the location where the data for image matching is stored. The image about command is then forwarded to the logical level.

3.2 Image search in SRA

None of the SRA operators defined at the logical level could handle such *about_image* statement. Therefore, to express image search in SRA we extended the SRA operator set with the additional operators σ^i and $\sqsupset_p^i$. The σ^i operator has similar definition as basic score region algebra operator $\sigma_{n=name,t=type}$, except that the score p is now computed by a call to an external function (f^i). The function f^i uses information extracted from the reference figure and the figure that should be selected and it computes the score of an image region based on similarity between the reference image and the selected image:

$$\sigma^{i\approx sample}_{n=name,t=attr}(R_1) := \{r | r_1 \in R_1 \ \wedge \ \exists r_2 \in C \ \wedge \ r_2 \prec r_1 \ \wedge \ t_2 = attr\_val \ \wedge$$

$$(s,e,n,t) := (s_1,e_1,n_1,t_1) \ \wedge \ t = attr \ \wedge \ n = name \ \wedge \ p = f^i(n_2, sample)\} \tag{10}$$

Here *sample* is the location of the reference image data specified with the '`src:`' statement in the NEXI query, resolved in the preprocessing step, C is a set of all regions in the database, *attr* it the attribute node, and *attr_val* is value of the attribute *attr*.

The operator $\sqsupset_p^i$ is defined in the same way as $\sqsupset_p$ operator (for the exact definition see [13]), except that it allows computing score of a region containing images with the usage of different scoring formula than for terms given in Equations 3 to 5. Therefore, the *about_image* in the multimedia query 1 is transformed into the next SRA expression:

$$\sigma_{n='image',t=node}(C) \sqsupset_p^i \sigma^{i\approx\text{'BN417\_16.jpg'}}_{n=file\_name,t=attr}(C)$$

3.3 Implementation of image search

At indexing time, we estimated a generative probabilistic model of each of the images in the collection (see below); the model parameters are stored in separate tables in the database. In addition, we constructed a table that links the image identifiers to the corresponding nodes in the collection tree. The image selection operator is implemented as a new MIL function that computes the Gaussian mixture model similarity score between each collection image model and the example image.

3.4 Retrieval model

Similarity between example images and collection images is estimated using Gaussian mixture models (GMM). To this end, each of the images in the collections $(\omega(n_i))$ is modeled as mixtures of Gaussians with a fixed number of components K:

$$P(\boldsymbol{x}|\omega(n_i)) = \sum_{k=1}^{N_K} P(K_{i,k}) \ \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}_{i,k}, \boldsymbol{\Sigma}_{i,k}), \tag{11}$$

where N_K is the number of components in the mixture model, $K_{i,k}$ is component k of class model $\omega(n_i)$ and $\mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the Gaussian density with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$:

$$\mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}, \tag{12}$$

where d is the dimensionality of the feature space and $(\boldsymbol{x} - \boldsymbol{\mu})^T$ is the matrix transpose of $(\boldsymbol{x} - \boldsymbol{\mu})$.

These Gaussian mixture models are used to represent the images. The score of an image given an example image from a query, is determined by the likelihood that the corresponding model generates the feature vectors ($\mathcal{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$) representing the example image. Like in the LM case for text, we interpolate with a background model based on collection statistics:

$$f^i(n_i, sample) = \prod_{\boldsymbol{x} \in \mathcal{X}_{sample}} [\lambda \cdot P(\boldsymbol{x}|\omega(n_i)) + (1-\lambda) \cdot P(\boldsymbol{x}|\omega(n_i))] \tag{13}$$

The feature space of the vectors $\boldsymbol{x}$ is based on the DCT coefficients obtained from 8x8 pixel blocks. For details of the feature vectors and the GMMs, see [21, 22].

On the other hand the scores in the operation $R_1 \sqsupset_p^i R_2$ are computed as a multiplication of score values of regions from the left operand and score values of the contained regions from the right operand, since each `image` element in the Lonely Planet collection containd only one image.

4 Relevance feedback

The main idea of any relevance feedback strategy is to use the knowledge of relevant items to retrieve more relevant items. So far, research has concentrated on using content-related information from the known relevant elements. However, for XML retrieval the structural characteristics of the relevant elements might also play an important role. Following the lines of what we started last year [14], we investigate the potential of the structural information for this type of task and analyze if retrieving structurally similar elements improves retrieval effectiveness.

4.1 Structural information in relevant elements

We study two different aspects of the structure of documents that can help the retrieval system to discriminate between relevant and non relevant elements. Namely, the containing journal of an element and the element type. Table 2 shows the number of different journals and element types judged relevant per topic. If we compare these numbers to the total number of different journals (24) and different element types (187) contained in the new collection, we can see that the knowledge of which journals and element types are relevant for each

Table 2. Number of different journals and element types judged relevant per topic. Statistics taken from relevance assessments 2005 version 2. Average over 28 CO topics. All degrees of relevance are taken into account.

Type info.	**Avg.**	**Median**	**Max**	**Min**
Journals	7.9	8	16	2
Elements	34.4	34.5	73	9

of the topics is a very important piece of information that can help retrieval systems to perform a better search.

One way to use the knowledge of which structural characteristics are relevant for a certain topic is to increase the a priori belief in relevance of the elements that have the same structural characteristics. In this way, we use the information of which relevant journals and element types are found in the top 20, to calculate priors and increase the a priori belief in relevance of the elements that are contained in that journal or that are from that specific element type.

4.2 Updating priors in a language modeling framework

For our baseline experiments, we used statistical language models (see Section 2.4). Using Bayes' rule and assuming independence between query terms, the probability of an element E given a query Q can be estimated as the product of the probability of generating the query terms q_i from the element's language model and the prior probability of the element:

$$P(E|Q) \propto \prod_{q_i \in Q} P(q_i|E)P(E) \tag{14}$$

Typically, little prior knowledge about the probability of an element is available and either uniform priors are used, or $P(E)$ is taken to be related to the element's length (i.e.,long elements are assumed to be more likely to contain relevant information) (cf. [9]). However, once we have some information about relevant elements, for example from a user's relevance judgments, we can use this information to update the priors. From the judgments, we can discover the characteristics of relevant elements and update the priors in such a way that elements with similar characteristics are favored[5]. Note that this does not require updating of the content models, i.e., the elements' language models do not change.

Therefore, once we get information about the structural characteristics of the relevant elements for a given topic, we define the priors for the journals and element types and use them to retrieve structurally similar elements. However, since in the top 20 we may not have seen all relevant journals or element types, there is the risk of assigning a prior equal to zero to element types or journals that do actually contain relevant information. To avoid this effect of relying too

[5] Strictly speaking $P(E)$ can no longer be called a prior, since it depends on the topic at hand.

much on what is seen in the top 20, we interpolate $P(x(E)|rel)$ with the general probability of seeing elements from $x(E)$. Thus the prior becomes:

$$P_x(E) = \frac{\alpha P(x(E)|rel) + (1-\alpha)P(x(E))}{P(x(E))}, \tag{15}$$

where $x(E)$ identifies the journal (element type) to which E belongs, $P(x(E)|rel)$ is estimated as the fraction of relevant items belonging to the journal (element type) and $P(x(E))$ is the fraction of elements in the collection that belongs to that journal (element type).

5 On overlap

To identify the appropriate element to return is not an easy problem. A common approach to remove overlap from result lists is to select the highest scored element from each of the paths. In our opinion, this approach has two main drawbacks. On the one hand, it does not consider the length of the elements. If a high scored element is rather small, all the other elements from that path (maybe only slightly lower scored) will be removed without considering if they would be more appropriate (size wise) retrieval units. On the other hand, this approach does not consider relationships between elements in the tree. If, for example, all the paragraphs within a section are high scored but only some are the highest elements of their path, only these ones will be returned. We argue that, in this scenario, it might be more appropriate to return the section that contains all the paragraphs than only some of the paragraphs on their own.

We believe that the appropriate retrieval unit will be determined by the total amount of *useful* information that unit contains. If a very high scored element is very short, the amount of useful information that carries is also small. Whereas if a not so high scored element is longer, the amount of *useful* information that the user will read is larger. Thus, the decision of which elements to return will be related not only to their retrieval model score but also to their size. In the same way, whether to return several siblings or their parent will be decided according to the amount of *irrelevant* information the user will have to read if the parent is returned. If the reminding text of the parent element contains somehow relevant information (even if not highly relevant), the parent should be returned and not the children.

To implement this idea, we define an *utility* function, related to the elements size and score, and calculate its value for each of the nodes in the tree. The nodes with a higher *utility* value than the sum of their children's ones are returned. In the case that the children need to be returned, only the ones with the *utility* value higher than a threshold are returned. Details on the different *utility* functions used and the results will be reported in the final version of this paper.

6 Users and information needs

The overall motivation of the interactive track at INEX is twofold. First, to investigate the behavior of users when interacting with components of XML documents, and second to investigate and develop approaches for XML retrieval which are effective in user-based environments [19]. A very important aspect of the track is the collaborative effort done in order to gather as much data on users' behavior as possible. For that, common user experiments are carried out by all participants and evidence is collected. These experiments not only record users' behavior but also question the users about several aspects of the search process such as interface and system issues, tasks, granularity of the answers, and users' satisfaction.

One of the new issues introduced in this year experiments is that users are asked to describe and perform the search based on their own information need. Our main interest is to analyze and classify different types of users' information needs and to understand what are their expectations regarding XML retrieval systems. Since the collected data has not yet been made available to all the participants, we will report our analysis in the final version of the paper.

7 Experiments

Among numerous tracks and scenarios specified for INEX 2005, we participated in the following: all CO and CAS ad-hoc track sub-tasks, multimedia track, interactive track, and relevance feedback track. Below, after introducing the metrics reported in the paper, we will explain in detail our approaches for each of these (sub)tasks. The runs given in bold are the official ones, but run on the updated (correct) volume files in INEX document collection.

7.1 Metrics

The official INEX metrics for 2005 ad-hoc and relevance feedback track are based on extended Cumulative Gain (xCG) metrics [10]. The official metrics are: normalized xCG (nxCG), effort-precision/gain-recall (ep/gr), and extended Q and R[6]. The evaluation can be done either with the generalized or with the strict quantization. In this paper we report the evaluation results obtained with nxCG at various recall points: 10, 25, and 50 and mean average ep/gr. For multimedia track we report mean average precision (MAP) values.

Note that for any document cut-off value, say 10, it can be shown that, if strict quantization is used (or any other binary quantization), and overlap is not taken into account, and the total number of relevant elements is bigger than 10, then nXCG at 10 and precision at 10 give exactly the same results. However, if the number of relevant elements is smaller than 10 for some topics, then this might have a big impact on the measured performance.

[6] http://inex.is.informatik.uni-duisburg.de/2005/inex-2005-metricsv4.pdf

For instance, IBM Haifa's run "SSCAS_no_phrase_no_plus" and Max Planck Institute's (MPI) run "MPII_TopX_SSCAS" have the same average precision at 10 over 4 topics with relevant elements: 0.225 for both runs (over topic 256, 260, 270 and 275). That is, on average 22.5% of the elements inspected in the top 10 is highly exhaustive and specific. However, for one of these 4 SSCAS topics (topic 270), only 1 relevant document is known. Because of this, the nXCG at 10 over the 4 topics is twice as high for MPI (0.450), which found the document in its top 10, as it is for IBM (0.225), which did not find it in its top 10. Apparently, a 100% gain in nXCG does not have to say much about the actual percentage of relevant items seen by the user. Precision at x is less sensitive to the total number of known relevant elements than XCG at x, and therefore defining the ideal recall base as needed for XCG is not really an issue for precision [8].

7.2 Ad-hoc track: CO queries

Thorough The aim of the *Thorough* retrieval strategy is to find all highly exhaustive and specific elements. Thus, to find all relevant information regardless of overlapping results. This year we submitted only two runs with the aim of using them as baseline runs for the other tasks and sub-tasks. Description and results for these two runs are given in Table 3.

Table 3. Results for CO.Thorough experiments with strict (^{S}) and generalized (^{G}) quantization.

Run id	**Description**	**nXCG[10]**	**nXCG[25]**	**nXCG[50]**	**ep/gr**
LMs_04_lp^S	LMs, $\lambda = 0.4$, lp	0.0880	0.0897	0.0996	0.0024
CO_LMs_trm_085^S	**LMs,** $\lambda = 0.85$**, lp**	0.0923	0.0855	0.0859	0.0022
LMs_04_lp^G	LMs, $\lambda = 0.4$, lp	0.2388	0.2540	0.2303	0.0849
CO_LMs_trm_085^G	**LMs,** $\lambda = 0.85$**, lp**	0.2161	0.1856	0.1839	0.0610

Although under the strict quantization there are not big differences between the two runs, under the generalized, one of the runs ($\lambda = 0.4$) outperforms the other considerably. We used this run as baseline for the rest of the CO experiments.

Focussed The aim of the *Focussed* retrieval strategy is to find the most exhaustive and specific element in a path. Once the element is identified and returned, none of the remaining elements in the path should be returned. In other words, the result list should not contain any overlapping elements.

The goal of our experiments for this task is twofold. We investigate if there are big differences in effectiveness between different approaches to remove overlap, and we evaluate the effectiveness of our own approach for different *utility* functions (see Section 5). As mention before, we do not have yet the results for our approach but they will reported in the final version of this paper. To investigate differences in performance between approaches, we implemented two already known ways of removing overlap: namely, the *naive* and the *common* approach. The *naive* approach filters out from the result list everything except one specific type of element (assuming that there is not overlap between elements of the same type). The *common* approach is implemented as follows: first, we select the highest scored element from the result list and remove its ancestors and descendants, then we take the second highest scored element and remove its ancestors and descendants, and then we continue recursively until all elements from the result list have been either selected or removed. The results from these implementations are shown in Table 4.

Table 4. Results for CO.Focussed experiments with strict (^{S}) and generalized (^{G}) quantization.

Approach	**nXCG[10]**	**nXCG[25]**	**nXCG[50]**	**ep/gr**
Baseline, with overlap^S	0.0817	0.0713	0.0777	0.0510
Naive: select articles^S	0.0080	0.0080	0.0120	0.0043
Naive: select sections^S	0.0320	0.0434	0.0421	0.0142
Naive: select paragraphs^S	0.1257	0.1600	0.1703	0.0676
Common^S	0.1097	0.0971	0.1134	0.0531
Baseline, with overlap^G	0.1441	0.1340	0.1260	0.0467
Naive: select articles^G	0.1557	0.1217	0.1031	0.0452
Naive: select sections^G	0.1852	0.1801	0.1560	0.0664
Naive: select paragraphs^G	0.2372	0.2262	0.2172	0.0834
Common^G	0.2296	0.1940	0.1986	0.0796

The approach that performs better is the one that retrieves only paragraphs. In general, for the *naive* approach, and as expected for a *Focussed* retrieval task, the longer the element, the lower the performance. It is however more surprising that, for the strict case, it is more desirable to return overlapping elements (baseline) than to return e.g. only the sections. In our opinion, for a focussed retrieval task the overlapping elements are not desired and therefore, should be stronger penalized. The common approach performs well although, due to the drawbacks mentioned before, some relevant information is removed.

Fetch and Browse The aim of the *Fetch and Browse* retrieval strategy is to first identify relevant articles (fetching phase), and then to identify the most exhaustive and specific elements within the fetched articles (browsing phase).

To achieve this task, several decisions need to be taken: e.g., how to rank the articles, how to rank the elements within an article, how many from these elements should be shown to the user (do we want to show more articles and few elements within them or many elements inside fewer articles?), do we want to return overlapping elements?

For this year experiments, we decided to rank the articles by its own score (the one given by the retrieval model) and to remove overlap inside the articles. We experimented with the number of elements to return within an article and the way to remove overlap. We only submitted two official runs and as we can not evaluate additional ones, the results of these experiments will be reported in the final version of the paper.

7.3 Ad-hoc track: CAS queries

Since we decided to extend the NEXI syntax with the vague selection we had to manually rewrite the queries for each CAS scenario except the SSCAS. For example, the (SS)CAS query 225:

```
//article[about(.//fm//atl, "digital libraries")]
                    //sec[about(.,"information retrieval")]
```

is rewritten into three variants:

- SVCAS:
  ```
  //article[about(.//∼fm//∼atl, "digital libraries")]
          //sec[about(.,"information retrieval")]
  ```
- VSCAS:
  ```
  //article[about(.//fm//atl, "digital libraries")]
          //∼sec[about(.,"information retrieval")]
  ```
- VVCAS:
  ```
  //article[about(.//∼fm//∼atl, "digital libraries")]
          //∼sec[about(.,"information retrieval")]
  ```

We decided not to consider the ‘article’ element as a vague element in case it is not the target element or it is not the element in which the *about* search should be performed, as in these cases the ‘article’ element just serves as a focusing element for deeper search in the XML tree.

We also aimed at comparing vague node selection with two query rewriting techniques that we used previous years for INEX [12, 14]. These rewriting techniques treat structural constrains as strict but mix the terms in different about clauses. In the first rewriting approach (rw I), all terms that are in different *about* clauses in the same predicate expression, and are not at the top level (i.e., not in `about(., term)` expression), are added to an extra top-level *about* close in the same predicate expression. The second approach (rw II), is an extension of the first one, where not only the terms from non top-level *abouts* are added to the new *about*, but also all the terms from the other predicate[7], if there exists any, are added to the top-level *about* in each predicate.

The results for different scenarios and the comparison of the approaches are given in the Table 5. For the official submissions we used the uniform down-weighting factor (either $w = 0.55$ or $w = 0.65$) based on the outcome of our experiments on INEX 2004 test collection. We report only the results with generalized quantization as only four out of seven assessed topics have highly exhaustive and highly specific elements for SSCAS sub-task.

Based on the experimental results given in Table 5 we can conclude that the rewriting techniques in general improve relevance score at early nxCG (except

[7] Note that the NEXI syntax allows only two predicates with the *about* clauses to be specified in one query.

Table 5. Results for various CAS experiments with generalized quantization.

Sub-task	Description	nXCG[10]	nXCG[25]	nXCG[50]	ep/gr
SSCAS	**LMs,** $\lambda = 0.5$	0.218	0.3141	0.4139	0.1191
	Okapi, $k_1 = 1.5$**,** $b = 0.75$	0.1671	0.2132	0.3846	0.1001
	LMs, $\lambda = 0.5$, rw I	0.2194	0.3991	0.4516	0.1248
	LMs, $\lambda = 0.5$, rw II	0.3129	0.424	0.466	0.1428
SVCAS	LMs, $\lambda = 0.5$	0.2138	0.2665	0.3082	0.1167
	LMs, $\lambda = 0.5$**,** $w = 0.55$	0.2203	0.2622	0.2868	0.1173
	LMs, $\lambda = 0.5$**, rw I**	0.1956	0.2774	0.3248	0.1187
	LMs, $\lambda = 0.5$, $w = 0.55$, rw II	0.2951	0.2989	0.3241	0.1315
VSCAS	LMs, $\lambda = 0.5$	0.2697	0.2754	0.257	0.0595
	LMs, $\lambda = 0.5$**,** $w = 0.55$	0.2	0.2263	0.2304	0.0925
	LMs, $\lambda = 0.5$, $w = 0.55$, rw I	0.1995	0.25	0.2464	0.0814
	LMs, $\lambda = 0.5$**, rw II**	0.298	0. 2908	0.2559	0.0648
VVCAS	LMs, $\lambda = 0.5$	0.2677	0.2815	0.2659	0.0509
	LMs, $\lambda = 0.5$**,** $w = 0.55$	0.2364	0.254	0.2425	0.0741
	LMs, $\lambda = 0.5$**,** $w = 0.65$	0.2295	0.2644	0.2322	0.0737
	LMs, $\lambda = 0.5$, $w = 0.55$, rw I	0.2246	0.2679	0.2467	0.0699
	LMs, $\lambda = 0.5$, $w = 0.55$, rw II	0.2823	0.2931	0.2554	0.1001

for rewrite I for nxCG[10]) for all sub-tasks, no matter if they are used in combination with the vague selection or not. Furthermore, the rewrite II technique in combination with the vague element selection seams to give the best scores. However, for ep/gr the rw I gives lower scores, while the rw II gives higher scores only in combination with the vague element selection. The vague element selection itself tends to improve the ep/gr but not the nxCG scores. Due to a small set of assessed topics we take this outcomes as a hypothesis.

7.4 Ad-hoc track: COS queries

Since COS queries have the same form as CAS queries we applied the same manual rewriting to COS queries. Thus, we made four different scenarios for COS queries, that we denote SSCOS, VSCOS, SVCOS, and VVCOS, and due to the limited number of submissions we submitted only SSCOS, VSCOS, and VVCOS as official runs. In our experiments we planned to test the degree of improvements in the effectiveness using the strict or vague structural constrains. The outcome for COS.Thorough can be seen in Table 6. We can see that with more vagueness we the results are better, for both metrics. However, for the COS.Thorough, as opposed to CAS subtasks, rw I seams to be more adequate than rw II in combination with the vague element selection. The analysis of the *Fetch and Browse* and *Focussed* scenarios will be given in the final paper.

7.5 Multimedia track: image queries

An important goal of our multimedia extension was to showcase and test the flexibility and extendibility of the SRA approach. In addition, we tested if us-

Table 6. Results for COS.Thorough experiments with strict (^{S}) and generalized (^{G}) quantization.

Sub-task	**Description**	**nXCG[10]**	**nXCG[25]**	**nXCG[50]**	**ep/gr**
SSCOS^S	**LMs,** $\lambda = 0.4$	0.0529	0.0536	0.0571	0.0014
SVCOS^S	LMs, $\lambda = 0.4$, $w = 0.55$	0.0529	0.0489	0.0559	0.0014
VSCOS^S	**LMs,** $\lambda = 0.4$, $w = 0.55$	0.0882	0.0736	0.0724	0.0020
VVCOS^S	**LMs,** $\lambda = 0.4$, $w = 0.55$	0.0882	0.0736	0.0895	0.0021
VVCOS^S	LMs, $\lambda = 0.4$, $w = 0.55$, rw I	0.0882	0.0883	0.0918	0.0021
VVCOS^S	LMs, $\lambda = 0.4$, $w = 0.55$, rw II	0.0882	0.0859	0.0859	0.0020
SSCOS^G	**LMs,** $\lambda = 0.5$	0.2677	0.222	0.176	0.0304
SVCOS^G	LMs, $\lambda = 0.4$, $w = 0.55$	0.2734	0.2265	0.1963	0.0351
VSCOS^G	**LMs,** $\lambda = 0.5$, $w = 0.4$	0.2625	0.2426	0.2104	0.0443
VVCOS^G	**LMs,** $\lambda = 0.5$, $w = 0.4$	0.2659	0.2551	0.2367	0.0677
VVCOS^G	LMs, $\lambda = 0.5$, $w = 0.4$, rw I	0.2943	0.27	0.2454	0.0706
VVCOS^G	LMs, $\lambda = 0.5$, $w = 0.4$, rw II	0.2686	0.2441	0.2179	0.0664

ing visual similarity can contribute to better multimedia retrieval results. To this end, we compared the multimedia queries discussed in Section 3 to similar queries with all image similarity clauses (`src:`) removed. The results of these two approaches using three different models for text search is given in Table 7. Language models and GPX clearly perform better than Okapi, but we did not find any improvement using visual similarity, in fact the best run uses only textual language models and is significantly better than its multimedia counterpart.

Table 7. Results for MM track.

LM	**MAP**	**Okapi**	**MAP**	**GPX**	**MAP**
text only	0.2751	text only	0.2110	text only	0.2567
multimedia	0.2600	multimedia	0.2133	multimedia	0.2627

7.6 Relevance feedback track

To analyze the effects of using structural information in the relevance feedback process as described in Section 4, we designed two main experiments. The first one varies the values for α in Equation 15 to analyze the effects of assigning different importance to the structural information found in the top 20. The values used are: 0.75, 0.5 and 0.25. This experiment is done on top of one of our runs for the CO.Thorough task that uses language models and a linear length prior.

The second experiment aims to identify which of the two types of structural information provides better improvement to the overall effectiveness of the IR system. Therefore, we fix the value of α in Equation 15 to 0.5 and analyze the gain obtained when using journal priors, element priors, and both priors at the same time. This experiment is done on top of one of our runs for the COS.Thorough task that uses the VVCAS approach explained in Section 2.

There is a common run in both experiments that is intended to show the differences in gain when using the journal priors ($\alpha = 0.5$) on top of a CO baseline and when using them on a COS baseline. Our goal is to see if the fact that the second baseline already uses structural information diminishes the effect of the priors.

In the time of writing this paper, there are not official results available. Thus, we will report them in the final version of the paper.

8 Conclusions and Future Work

Throughout the paper we show that the TIJAH database system is flexible enough to incorporate new advanced search techniques, such as vague element selection and relevance feedback, and search on heterogeneous data sources, such as a combination of images and text.

We plan to continue the experimental evaluation of different scenarios for search in structured documents: (1) the approaches for handling overlap and for supporting user searching behavior (fetch and browse), (2) the vague element search with non-uniform down-weighting (based on assessment results) and its combination with rewriting techniques, (3) the usage of structural relevance feedback, and (4) image search for improving retrieval results.

9 Acknowledgments

We would like to thank Roberto Cornacchia at CWI, Amsterdam, for providing the visual similarity code and for pre-processing the Lonely Planet images.

Many thanks to the Netherlands Organisation for Scientific Research (NWO) for funding the research described in this paper (grant number 612.061.210).

References

1. P. Boncz. *Monet: a Next Generation Database Kernel for Query Intensive Applications.* PhD thesis, CWI, 2002.
2. J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY Retrieval System. In *Proceedings of the 3rd DEXA Conference*, 1992.
3. A.P. de Vries. *Content and Multimedia Database Management Systems.* PhD thesis, University of Twente, Twente, The Netherlands, 1999.
4. A. Doan and A.Y. Halevy. Semantic Integration Research in the Database Community. *AI Magazine*, 26:83–94, 2005.
5. N. Fuhr and K. Großjohann. XIRQL: An XML Query Language Based on Information Retrieval Concepts. *ACM Transactions on Information Systems*, 22(2):313–356, 2004.
6. S. Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. In N. Fuhr, M. Lalmas, and S. Malik, editors, *Proceedings of the Third Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, volume 3493 of *Lecture Notes in Computer Science*, pages 276–291, 2005.

7. D. Hiemstra. *Using Language Models for Information Retrieval.* PhD thesis, University of Twente, Twente, The Netherlands, 2001.
8. D. Hiemstra and V. Mihajlovic. The simplest evaluation measures for xml information retrieval that could possibly work. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, 2005.
9. Jaap Kamps, Maarten de Rijke, and Börkur Sigurbjörnsson. Length Normalization in XML Retrieval. In *SIGIR '04: Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval*, pages 80–87, 2004.
10. G. Kazai, M. Lalmas, and A.P. de Vries. The Overlap Problem in Content-oriented XML Retrieval Evaluation. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.
11. G. Kazai, M. Lalmas, and S. Malik. INEX'03 Guidelines for Topic Developments. In *Proceedings of the Second Initiative on the Evaluation of XML Retrieval (INEX 2003)*, ERCIM Workshop Proceedings, 2004.
12. J. List, V. Mihajlović, A. de Vries, G. Ramirez, and D. Hiemstra. The TIJAH XML-IR System at INEX 2003. In *Proceedings of the 2nd Initiative on the Evaluation of XML Retrieval (INEX 2003)*, ERCIM Workshop Proceedings, 2004.
13. V. Mihajlović, H.E. Blok, D. Hiemstra, and P.M.G. Apers. Score Region Algebra: Building a Transparend XML-IR Database. In *Proceedings of the ACM CIKM Conference*, 2005.
14. V. Mihajlović, G. Ramírez, A.P. de Vries, D. Hiemstra, and H.E. Blok. TIJAH at INEX 2004: Modeling Phrases and Relevance Feedback. In N. Fuhr, M. Lalmas, and S. Malik, editors, *Proceedings of the Third Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, volume 3493 of *Lecture Notes in Computer Science*, pages 276–291, 2005.
15. G.A. Miller, C. Fellbaum, R. Tengi, S. Wolff, P. Wakefield, H. Langone, and B. Haskell. WordNet: A Lexical Database for the English Language.
16. E. Rahm and P.A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *The VLDB Journal - The International Journal on Very Large Databases*, 10:334–350, 2001.
17. G. Ramírez, T. Westerveld, and A.P. de Vries. Structural Features in Content Oriented XML Retrieval. In *Proceedings of the ACM CIKM Conference*, 2005.
18. S. E. Robertson and S. Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th ACM SIGIR Conference*, 1994.
19. A. Tombros, B. Larsen, and S. Malik. The Interactive Track at INEX 2004. In N. Fuhr, M. Lalmas, and S. Malik, editors, *Proceedings of the Third Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, volume 3493 of *Lecture Notes in Computer Science*, pages 410–423, 2005.
20. A. Trotman and R. A. O'Keefe. The Simplest Query Language That Could Possibly Work. In N. Fuhr, M. Lalmas, and S. Malik, editors, *Proceedings of the Second Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, ERCIM Publications, 2004.
21. T. Westerveld. *Using generative probabilistic models for multimedia retrieval.* Ph.d. thesis, University of Twente, Enschede, The Netherlands, November 2004.
22. Thijs Westerveld and Arjen P. de Vries. Generative probabilistic models for multimedia retieval: query generation versus document generation. *IEE Proceedings - Vision, Image and Signal Processing*, 152(6):852–858, 2005.

XFIRM at INEX 2005: ad-hoc, heterogeneous and relevance feedback tracks. Preliminary work

Karen Sauvagnat, Lobna Hlaoua, and Mohand Boughanem

IRIT-SIG,
118 route de Narbonne, F-31 062 Toulouse Cedex 4, France

Abstract. This paper describes experimentations carried out with the XFIRM system in the INEX 2005 framework. The XFIRM system uses a relevance propagation method to answer CO and CAS queries. Runs were submitted to the ad-hoc, heterogeneous and relevance feedback tracks.

1 Introduction

The approach we used for our participation at INEX 2005 is based on the XFIRM system, and uses a relevance propagation method. The XFIRM system was adapted for submitting runs to the ad-hoc track (for CO, CO+S, and CAS tasks), the heterogeneous track and the relevance feedback track.

2 Experimental setup

2.1 XFIRM data model

The XFIRM system is based on a relevance propagation method. We use a generic data model that allows the implementation of many IR models and the processing of heterogeneous collection. We consider that a structured document sd_i is a tree, composed of leaf nodes ln_{ij} and attributes a_{ij} and simple nodes n_{ij} (all nodes that are not leaf nodes or attributes).
Structured document: $sd_i = (\{n_{ij}\}, \{ln_{ij}\}, \{a_{ij}\})$
In order to easily browse the document tree and to quickly find ancestors-descendants relationships, the model uses a representation of nodes and attributes based on the Xpath Accelerator approach [2].
All leaf nodes are indexed, because even the smallest leaf nodes can be relevant or can give information on the relevance of their ancestors. Intuitively, *title* or *subtitle* nodes are not informative, but if a query term occurs in those nodes, such information can be useful for evaluating the relevance of the ancestor node. Such an approach has other advantages: the index process can be done automatically, without any human intervention and the system will be so able to handle heterogeneous collections automatically; and secondly, even the most specific query concerning the document structure will be processed, since all the document structure is stored.

During query processing, relevance values are assigned to leaf nodes and relevance score of inner nodes are then computed dynamically, thanks to a propagation of leaf nodes score through the document tree. An ordered list of subtrees is then returned to the user.

2.2 Evaluation of leaf nodes scores

Whatever the considered type of queries, a first step in query processing is to evaluate the relevance value of leaf nodes ln according to the query. Let $q = t_1, \ldots, t_n$ be this query. Relevance values are computed thanks to a similarity function $RSV_m(q, ln)$, where m is an IR model.

$$RSV_m(q, ln) = \sum_{i=1}^{n} w_i^q * w_i^{ln} \tag{1}$$

Where w_i^q and w_i ln are respectively the weights of term i in query q and leaf node ln.
According to previous experiments [5], we choose to use the following term weighting scheme, which aims at reflecting the importance of terms in leaf nodes, but also in whole documents:

$$w_i^q = tf_i^q \qquad w_i^{ln} = tf_i^{ln} * idf_i * ief_i \tag{2}$$

Where tf_i^q and tf_i^{ln} are respectively the frequency of term i in query q and leaf node ln, $idf_i = log(|D|/(|di| + 1)) + 1$, with $|D|$ the total number of documents in the collection, and $|di|$ the number of documents containing i, and ief_i is the inverse element frequency of term i, i.e. $log(|N|/|nf_i| + 1) + 1$, where $|nf_i|$ is the number of leaf nodes containing i and $|N|$ is the total number of leaf nodes in the collection.
Inner nodes relevance values are evaluated thanks to one or more propagation functions, which depend on the searching task. Such propagation functions are described in the following sections.

3 CO task

3.1 Inner nodes relevance values evaluation

In our model, each node in the document tree is assigned a relevance value which is function of the relevance values of the leaf nodes it contains. Terms that occur close to the root of a given subtree seem to be more significant for the root element that ones on deeper levels of the subtrees. It seems therefore that the larger the distance of a node from its ancestor is, the less it contributes to the relevance of its ancestor. This affirmation is modelled in our propagation formula by the use of the $dist(n, ln_k)$ parameter. $dist(n, ln_k)$ is the distance between node n and leaf node ln_k in the document tree, i.e. the number of arcs that are necessary to join n and ln_k.
It is also intuitive that the more a node contains relevant leaf nodes, the more it is relevant. We then introduce in the propagation function the $|L_n^r|$ parameter,

which is the number of leaf nodes being descendant of n and having a non-zero relevance value (according to equation 1).

A relevance propagation function using these parameters has been tested in the INEX 2004 framework. In the 2005 evaluation campaign, we propose to add two parameters:

- We propose to increase small nodes importance during propagation. Indeed, we think that authors of documents use small nodes to highlight important informations. These nodes can therefore give precious indications on the relevance of their ancestors. In our propagation function, this intuition corresponds to the $\beta(ln_k)$ parameter.
- We introduce the ρ parameter, inspired from work presented in [3], which allows the introduction of document relevance in inner nodes relevance evaluation. The idea behind context is: an element in a relevant document should be better ranked than an identical element in a non-relevant document.

The relevance value r_n of a node n is therefore computed according the following formula:

$$\begin{aligned} r_n &= \rho * |L_n^r|. \sum_{ln_k \in L_n} \alpha^{dist(n,ln_k)-1} * \beta(ln_k) * RSV(q, ln_k) \\ &\quad +(1-\rho) * |L^r|. \sum_{ln_k \in L} \alpha^{dist(root,ln_k)-1} * \beta(ln_k) * RSV(q, ln_k) \\ &= \rho * |L_n^r|. \sum_{ln_k \in L_n} \alpha^{dist(n,ln_k)-1} * \beta(ln_k) * RSV(q, ln_k) \\ &\quad +(1-\rho) * r_{root} \end{aligned} \tag{3}$$

where ln_k are leaf nodes being descendant of n, L_n is the set of leaf nodes being descendant of n, and

$$\beta(ln_k) = \begin{cases} l_k/\Delta l \ if \ dist(n, ln_k) = 1 \ and \ l_k < \Delta l \\ log(\Delta l/l_k) \ if \ dist(n, ln_k) > 1 \ and \ l_k < \Delta l \\ 1 \ else \end{cases} \tag{4}$$

with l_k the length of node ln_k and Δl the average length of leaf nodes in the collection.

3.2 Runs

CO.Thorough strategy. For the CO.Thorough task, all nodes having a non-zero relevance value are returned by the XFIRM system. We experimented using various values of $\rho \in [0..1]$.

CO.Focussed strategy. In order to reduce/remove nodes overlap, we use two different algorithms:

1. For each relevant path, we keep the most relevant node in the path (around 20% of nodes overlap still remains)
2. For each relevant path, we keep the most relevant node in the path. The results set is then parsed again, to eliminate any possible overlap among ideal components.

CO.FetchAndBrowse strategy. In this task, elements are first ranked by the relevance of the document they belong to, and then by their own relevance. We use the following algorithm:

1. relevance values are computed for each document in the collection;
2. relevance values are computed for each node of the collection;
3. documents are ranked by decreasing order of relevance;
4. for each document, elements they contain are ranked by decreasing order of relevance and are returned to users.

Documents relevance are computed with the Mercure system [1].

3.3 Results

All results described in this paper use the inex1.8 version of the collection, which is the official 2005 collection. However, due to a misunderstanding, our official submissions were obtained with the inex1.6 version of the collection. For information, official submissions are mentioned in italic characters and are followed by the '*' symbol.

CO.THOROUGH strategy. Tables 1 and 2 show the results obtained with different values of ρ.

	nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
$\rho = 1$	0,1684	0,168	0,1772	0,0562	0,1100	0,2231
$\rho = 0.9$ *	*0,15* *	*0,156* *	*0,174* *	*0,043* *	*0,085* *	*0,188* *
$\rho = 0.9$	0,1712	0,1696	0,1786	0,0577	0,1089	0,2179
$\rho = 0.8$	0,1634	0,1845	0,1859	0,0569	0,1057	0,2138
$\rho = 0.7$	0,1727	0,2006	0,1883	0,0569	0,1044	0,2110
$\rho = 0.6$	0,1713	0,2058	0,1928	0,0565	0,1031	0,2078
$\rho = 0.5$	0,1762	0,2036	0,1894	0,0561	0,1019	0,2051
$\rho = 0.4$	0,1802	0,2075	0,1897	0,0557	0,1009	0,2040
$\rho = 0.3$	0,1931	0,2116	0,188	0,0555	0,1001	0,2020
$\rho = 0.2$	0,2049	0,2126	0,1857	0,0553	0,0996	0,2010
$\rho = 0.1$	0,2083	0,2144	0,1868	0,0548	0,0986	0,2011
$\rho = 0$	0,2384	0,2126	0,1862	0,0542	0,0976	0,1981

Table 1. CO.Thorough strategy. Quantisation: Generalised

Best results are obtained with small values of ρ, especially for the strict quantisation function. This seems to show that root relevance (i.e. document relevance) has a high impact on elements relevance.

CO.FOCUSSED strategy. Tables 3 and 4 show the results obtained with different values of ρ.

Algorithm 2 (results without any nodes overlap) allows to obtain better results than algorithm 1 for all metrics. As opposed to results obtained for the CO.Thorough strategy, document relevance seems to have no impact on element relevance (best results were obtained with $\rho = 1$).

	nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
$\rho = 1$	0,012	0,0299	0,0464	0,0009	0,0012	0,0208
$\rho = 0.9$ *	0,011 *	0,025 *	0,047 *	0,001 *	0,001 *	0,021 *
$\rho = 0.9$	0,012	0,0258	0,0462	0,0012	0,0015	0,0216
$\rho = 0.8$	0,008	0,0329	0,0475	0,0014	0,0016	0,0213
$\rho = 0.7$	0,008	0,0425	0,0514	0,0015	0,0017	0,0216
$\rho = 0.6$	0,008	0,0505	0,0522	0,0016	0,0018	0,0218
$\rho = 0.5$	0,012	0,0569	0,0546	0,0017	0,0019	0,0209
$\rho = 0.4$	0,016	0,0585	0,0555	0,0017	0,0019	0,0206
$\rho = 0.3$	0,024	0,0617	0,0555	0,0017	0,0020	0,0199
$\rho = 0.2$	0,036	0,0633	0,0555	0,0018	0,0020	0,0199
$\rho = 0.1$	0,044	0,0633	0,0563	0,0019	0,0021	0,0199
$\rho = 0$	0,0684	0,0636	0,0579	0,0019	0,0021	0,0194

Table 2. CO.Thorough strategy. Quantisation: Strict

		nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
Algorithm 1	$\rho = 1$	0,1202	0,1214	0,1279	0,0393	0,0798	0,1543
	$\rho = 0.9$	0,112	0,1073	0,0941	0,0260	0,0609	0,1249
	$\rho = 0.8$	0,1146	0,106	0,093	0,0256	0,06042	0,1208
	...	...	...	...	...	...	...
	$\rho = 0.1$	0,1042	0,094	0,0852	0,0231	0,0577	0,1177
	$\rho = 0$	0,0951	0,0901	0,078	0,0235	0,0607	0,1172
Algorithm 2	$\rho = 1$ *	0,119 *	0,122 *	0,119 *	0,030 *	0,060 *	0,132 *
	$\rho = 1$	0,1364	0,1445	0,1453	0,0396	0,0748	0,1579
	$\rho = 0.9$*	0.104 *	0.104 *	0.089 *	0.022 *	0.052 *	0.106 *
	$\rho = 0.9$	0,1299	0,1171	0,1021	0,0276	0,0624	0,1271
	$\rho = 0.8$	0,1235	0,1144	0,0988	0,0271	0,0622	0,1258
	...	...	...	...	...	...	...
	$\rho = 0.1$	0,1131	0,1033	0,092	0,0256	0,0613	0,1197
	$\rho = 0$	0,0951	0,0901	0,078	0,0235	0,0607	0,1172

Table 3. CO.Focussed strategy. Quantisation: Generalised

CO.FETCHBROWSE strategy. Results obtained with the CO.FetchBrowse strategy are described in table 5. Results are good, since we were ranked in the top 5 for both quantisation functions. More over results are significantly better for the MAP metric than those obtained for the CO.Thorough strategy (see tables 1and 2).

4 CAS task

4.1 Inner nodes relevance value evaluation

The evaluation of a CAS query is carried out as follows:

1. INEX (NEXI) queries are translated into XFIRM queries
2. XFIRM queries are decomposed into sub-queries SQ and elementary sub-queries ESQ, which are of the form: $ESQ = tg[q]$, where tg is a tag name, i.e. a structure constraint, and $q = t_1, ..., t_n$ is a content constraint composed of simple keywords terms.

		nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
Algorithm 1	$\rho = 1$	0,012	0,016	0,0336	0,0024	0,0034	0,0051
	$\rho = 0.9$	0,014	0,0156	0,0188	0,0030	0,0038	0,0015
	$\rho = 0.8$	0,014	0,0156	0,0196	0,0031	0,0039	0,0011
	...	...	...	...	...	...	...
	$\rho = 0.1$	0,004	0,0056	0,0128	0,0011	0,0016	0,0008
	$\rho = 0$	0	0,004	0,012	0,0009	0,0015	0
Algorithm 2	$\rho = 1$ *	*0,011 **	*0,006 **	*0,025 **	*0,002 **	*0,003 **	*0,004 **
	$\rho = 1$	0,016	0,0112	0,0296	0,0034	0,0046	0,0066
	$\rho = 0.9$*	*0.014 **	*0.020 **	*0.028 **	*0.004 **	*0.004 **	*0.002 **
	$\rho = 0.9$	0,014	0,0172	0,0204	0,0031	0,0039	0,0018
	$\rho = 0.8$	0,014	0,0172	0,0236	0,0031	0,0039	0,0011
	...	...	...	...	...	...	...
	$\rho = 0.1$	0,004	0,0072	0,0176	0,0013	0,0019	0,0011
	$\rho = 0$	0	0,004	0,012	0,0009	0,0015	0

Table 4. CO.Focussed strategy. Quantisation: Strict

	Generalised	Strict
$\rho = 1$	0,1167	0,0063
$\rho = 0.9$ *	*0,108 **	*0,006 **
$\rho = 0.9$	0,1229	0,0068
	...	...
$\rho = 0.1$	0,1183	0,0065
$\rho = 0$	0,0731	0,0042

Table 5. CO.FetchBrowse strategy. ep/gr - MAP-Element metric

3. Relevance values are then evaluated between leaf nodes and the content conditions of elementary sub-queries
4. Relevance values are propagated in the document tree to answer to the structure conditions of elementary sub-queries
5. Sub-queries are processed thanks to the results of elementary sub-queries
6. Original queries are evaluated thanks to upwards and downwards propagation of the relevance weights

Step 3 is processed thanks to formula 1. In step 4, the relevance value r_n of a node n to an elementary subquery $ESQ = tg[q]$ is computed according the following formula:

$$r_n = \begin{cases} \sum_{ln_k \in L_n} \alpha^{dist(n,ln_k)-1} * RSV(q, ln_k) \; if \; n \; \in construct(tg) \\ 0 \; else \end{cases} \tag{5}$$

where the $construct(tg)$ function allows the creation of set composed of nodes having tg as tag name, and $RSV(q, ln_k)$ is evaluated during step 2 with equation 1. The $construct(tg)$ function uses a *Dictionnary* Index, which provides for a given tag tg the tags that are considered as equivalent. For example, a *title* node can be considered as equivalent to a *sub-title* node. This index is built manually. More details about CAS queries processing are can be found in [5].

4.2 Runs

In order to answer the different searching tasks, we used different Dictionnary indexes:

- The DICT index is composed of equivalencies given in the INEX guidelines. For example, *ss1*, *ss2* and *ss3* nodes are considered as equivalent to *sec* nodes.
- The ExtendedDICT is composed of very extended equivalencies. For example, *sec*, *ss1*, *ss2* and *ss3* nodes are equivalent to both *p* and *bdy* nodes.

SSCAS strategy. We use the DICT index and results are filtered in order to answer strictly to constraints on the target element and support elements.

VVCAS strategy. We use the EXtendedDICT index, and no filter is applied on results.

SVCAS strategy. We use the DICT index. No filter is applied on results: they match the structure constraint on the target element in a strict way (since the DICT index is used), and their relevance score is eventually increased by the relevance score of results of subqueries on support elements.

VSCAS strategy. We use the DICT index on support elements and the ExtendedDICT on target elements.

4.3 Results

Results for all strategies are showed in tables 6, 7, 8 and 9. Results are good, since we are in the top 10 for almost all metrics.

		nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
Generalised	DICT *	0,329 *	0,397 *	0,374 *	0,105 *	0,157 *	0,258 *
	DICT	0.2861	0.2722	0.338	0.109	0.178	0.246
Strict	DICT *	0,325 *	0,31 *	0,32 *	0,016 *	0,02 *	0,121 *
	DICT	0.35	0.329	0.338	0.0166	0.021	0.1169

Table 6. SSCAS strategy

		nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
Generalised	ExtendedDICT *	0,29 *	0,258 *	0,246 *	0,061 *	0,101 *	0,206 *
	ExtendedDICT	0.3047	0.2727	0.2487	0.0687	0.115	0.219
Strict	ExtendedDICT *	0,067 *	0,076 *	0,136 *	0,005 *	0,006 *	0,044 *
	ExtendedDICT	0.0885	0.0756	0.1244	0.0054	0.0059	0.0468

Table 7. VVCAS strategy

		nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
Generalised	_DICT *_	_0,303 *_	_0,272 *_	_0,276 *_	_0,105 *_	_0,181 *_	_0,301 *_
	DICT	0.2645	0.2758	0.2916	0.1378	0.2318	0.330
Strict	_DICT *_	_0,42 *_	_0,408 *_	_0,416 *_	_0,017 *_	_0,02 *_	_0,1 *_
	DICT	0.44	0.4571	0.4662	0.017	0.022	0.11

Table 8. SVCAS strategy

		nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
Generalised	_DICT+ExtendedDICT *_	_0,194 *_	_0,21 *_	_0,207 *_	_0,07 *_	_0,119 *_	_0,218 *_
	DICT+ExtendedDICT	0.237	0.2346	0.2292	0.047	0.069	0.159
Strict	_DICT+ExtendedDICT *_	_0_	_0,007 *_	_0,023 *_	_0,007 *_	_0,008 *_	_0,07 *_
	DICT+ExtendedDICT	0.1667	0.15	0.15	0.006	0.007	0.05

Table 9. SVCAS strategy

5 CO+S task

5.1 Inner nodes relevance value evaluation

In the CO+S task, queries are processed as in the CAS task. Nodes relevance values are evaluated using equation 5.

5.2 Runs

+S.THOROUGH strategy and +S.FOCUSSED strategy. We either use the DICT or ExtendedDICT dictionnary index, since the aim of the task is to investigate the usefulness of the structural hints.

+S.FETCHBROWSE strategy We follow the same algorithm as the one used for the CO.FETCHBROWSE strategy.

5.3 Results and comparison to the CO task

	nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
DICT *	_0,172 *_	_0,147 *_	_0,123 *_	_0,016 *_	_0,03 *_	_0,089 *_
DICT	0,1759	0,162	0,1422	0,0192	0,0369	0,1022
ExtendedDICT*	_0,169 *_	_0,191 *_	_0,187 *_	_0,045 *_	_0,088 *_	_0,001 *_
ExtendedDICT	0,1787	0,2037	0,206	0,0569	0,1086	0,2166

Table 10. COS.Thorough strategy. Quantisation: Generalised

+S.THOROUGH strategy. Results are not as good as those obtained without structural hints (see table 1 and 2 for comparison).

+S.FOCUSSED strategy As opposed to results obtained for the +S.THOROUGH strategy, results here are better than those obtained without using structural hints (see tables 3 and 4 for comparison).

	nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
DICT *	0,024 *	0,037 *	0,047 *	0 *	0 *	0,008 *
DICT	0,0242	0,0321	0,0362	0,0003	0,0004	0,0062
ExtendedDICT*	0,027 *	0,042 *	0,056 *	0,001 *	0,001 *	0,019 *
ExtendedDICT	0,0308	0,0434	0,0532	0,0012	0,0014	0,0210

Table 11. COS.Thorough strategy. Quantisation: Strict

	nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
DICT	0,1567	0,1362	0,121	0,0458	0,1046	0,1783
ExtendedDICT*	0,144 *	0,128 *	0,127 *	0,031 *	0,069 *	0,143 *
ExtendedDICT	0.1586	0.1497	0.1428	0.0410	0.0855	0.1674

Table 12. COS.Focussed strategy. Quantisation: Generalised

	nxCG[10]	nxCG[25]	nxCG[50]	ep/gr - MAP	Q	R
DICT	0,0231	0,0308	0,0345	0,0096	0,01183	0,0166
ExtendedDICT*	0,009 *	0,009 *	0,025 *	0,002 *	0,004 *	0,004 *
ExtendedDICT	0.0154	0.0163	0.0571	0.0032	0.0046	0.0086

Table 13. COS.Focussed strategy. Quantisation: Strict

	Generalised	Strict
DICT *	0,111 *	0,015 *
DICT	0.0155	0,0008
ExtendedDICT *	0.0747 *	0,005 *
ExtendedDICT	0,072	0.0058

Table 14. +S.FetchBrowse strategy. ep/gr - MAP-Element metric

+S.FETCHBROWSE strategy. As for the +S.THOROUGH strategy, results obtained without using structural hints are better than those obtained for the +S.FETCHBRWOSE strategy (see table 5 for comparison).

6 Heterogeneous track

In progress.

7 Relevance feedback track

For the RF track, we used the three different algorithms described below.

7.1 Structure-oriented Relevance Feedback

Our goal in what we call structure-oriented RF is to enrich the initial CO query by adding structural constraints. Results of CO queries are components of different granularities. Even if a user does not ask for a particular component type, he's often interested in a few types of components (like for example *references* or *sections*).

Our approach consists in extracting from the set of judged elements the structure that could contain the information needed by the user. The idea in structure-oriented RF is therefore to find for each query the *appropriate generic structure*,

which is the generic structure shared by the greatest amount of relevant elements.
This structure will be added to the initial query in order to improve the information retrieval effectiveness.

Our algorithm consists in carrying out the intersection of each structure of the elements judged as relevant with the rest of relevant elements structures. As a result, we obtain a set of Common Structures, called SC.
Let E^r be the set of relevant elements and e_i be an element $\in E^r$. e_i is characterized by a path p_i and a relevance score w_i computed by the relevance evaluation process. A *simple path* sp_i composed of tag names can be derived from p_i. For example, the simple path corresponding to the path $/article[1]/bdy[1]/sec[3]$ is: $/article/bdy/sec$.
For each element $e_i \in E^r$, and for each $e_j \in E^r - \{e_i\}$, we apply the SCA function, which allows to retrieve (and to weight) the simple path of the smallest common ancestor of n_i and n_j. This simple path is then added to the set of common structures SC. The SCA fuction is computed for each pair elements of E^r. More precisely, we use the following algorithm for SCA:

```
$SCA(e_i, e_j)$
begin
If $sp_i.last = sp_j.last$, then $w_i \leftarrow w_i + w_j$
        if $\exists e_p(sp_p, w_p), with\ sp_p \in SC/sp_p.last = sp_i.last$  then $w_p \leftarrow w_p + w_i$
                                                                              else $SC \leftarrow sp_i$
If $sp_i.last \neq sp_j.last$, then $sp_j \leftarrow tail(sp_j)$
        $w_j \leftarrow w_j/2$
        $SCA(e_i, e_j)$
end
```

with *sp.last* is the last tag of the path *sp* and *tail*(*sp*) is a function allowing to contract the path *sp*, i.e. to remove the last tag of the path. For example, $tail(/article/bdy/section) = /article/bdy$.
In order to express the new (CAS) query, we then extract the top ranked structure in the SC set. This structure will be either used as it is in the new query (complex form) or simplified in a simple tag form. Original query terms are then added to the structural constraint.
For example, let $/article/bdy/sec$ be the top ranked structure of the SC set and "information retrieval"' be the original CO query. The new CAS query will either be *"/article/bdy/sec(about(.,information retrieval)"* (complex form) or *"sec(about(.,"information retrieval")"* (simple form).

7.2 Content-Oriented Relevance Feedback

Our Content-Oriented Relevance Feedback approach is based on the Rocchio' algorithm [4]. Our aim is to extract the most expressive terms from relevant elements. The content-oriented RF processes as follows :

– Let's consider the set of relevant elements $(E^r) : E^r = e_1^r, e_2^r, ..., e_k^r, ...e_m^r$,
– A relevant element e_k^r is composed of a set of leaf nodes$(ln_j) : e_k^r = ln_1^k, ..., ln_j^k, .ln_n^k$
– A leaf node ln_j^k is a sequence of terms: $ln_j = \{t_{ij}\}$.

For each term is assigned a score according to the following formula:

$$score(t_{ij}, ln_j^k) = \frac{tf_i^j}{size(ln_j)} \tag{6}$$

Where tf_i^j is the frequency of term t_i in leaf node ln_j^k and $size(ln_j)$ is the number of terms in ln_j.
We then compute the score of terms for each relevant element. For each term, we sum its scores in different leaf nodes.

$$score(t_i, e_k^r) = \sum_{ln_j \in e_k^r} score(t_i, ln_j) \tag{7}$$

As a result, we obtain a set of expressive words for each element judged as relevant. Best terms are selected according to the scores in the set of relevant elements E^r:

$$score(t_i) = \sum_{e_k^r \in E^r} score(t_i, e_k^r) \tag{8}$$

The new query is finally composed of terms ranked in the top k according to formula 8, that are added to the original query terms.

7.3 Content-and-Structure-Oriented Relevance Feedback

In this approach, we propose to combine the structure-oriented Relevance Feedback method and the content-oriented Relevance Feedback described above. The new query (CAS) is composed of the most appropriate generic stucture (complex or simple form) and of terms ranked in the top k according to formula 8, that are added to the original query terms.

7.4 Runs

CO.Thorough task. For this task, we used two approaches. The first uses the Structure-oriented Relevance Feedback method (described in section 7.1). The final CAS is composed of the most appropriate generic structure (specifying the most appropriate generic path: complex form) and of the original query terms. We considered relevant elements having an exhaustivity value E $\geq$ 1.
The second approach we used is the Content-and-Structure-Oriented Relevance Feedback. We added to the original query the terms ranked in the top 15 and the most appropriate generic structure (specifying the most appropriate generic tag: simple form).

COS.Thorough and VVCAS task. We applied in the same way two approaches for the COS.Thorough and VVCAS queries.
The first uses the Content-oriented Relevance Feedback method (described in the section 7.2). We added to the original query the terms ranked in the top 15. We considered relevant elements having an exhaustivity value E ≥ 1.
The second uses the Content-and-Structure-Oriented Relevance Feedback method. We added to the original CAS query (thanks to the boolean operator 'OR') the most appropriate generic structure (specifying the most appropriate tag: simple form) containing the 15 top ranked terms. We considered two cases for the relevant elements: E ≥ 1 and E =2.

References

1. M. Boughanem, T. Dkaki, J. Mothe, and C. Soule-Dupuy. Mercure at TREC-7. In *Proceedings of TREC-7*, 1998.
2. T. Grust. Accelerating XPath location steps. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA*. In M. J. Franklin, B. Moon, and A. Ailamaki, editors, ACM Press, 2002.
3. Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement for XML retrieval. In *Proceedings of INEX 2004, Springer*, pages 73–84, march 2005.
4. J. Rocchio. *Relevance feedback in information retrieval.* Prentice Hall Inc., Englewood Cliffs, NJ, 1971.
5. K. Sauvagnat, M. Boughanem, and C. Chrisment. Using relevance propagation for processing content-and-structure queries. *Information Systems, special issue on SPIRE 04*, 2006.

University of Amsterdam at INEX 2005: Adhoc Track

Börkur Sigurbjörnsson[1], Jaap Kamps[1,2], and Maarten de Rijke[1]

[1] Informatics Institute, Faculty of Science, University of Amsterdam
[2] Archives and Information Science, Faculty of Humanities, University of Amsterdam

Abstract. In this paper we describe the University of Amsterdam's participation in the INEX 2005 adhoc track. Our main research questions for this round of INEX were to investigate selective indexing strategies and different methods for using structural constraints in queries. As a side question, we did experiment with the automatic creation of structured queries.

1 Introduction

In this paper we describe the University of Amsterdam's participation in the INEX 2005 adhoc track. In previous years we have made our runs based on an index of all overlapping XML elements. Our main objective this year was to experiment with different methods of creating a more selective index. The aim is to create a more efficient retrieval system without sacrificing much of retrieval effectiveness. In our experiments with structured queries in the previous years we have found that structural constraints lead to improvements in initial precision. This year we wanted to explore whether different types of structural constraints contribute differently to this gain.

For the *CO.Focussed* task we experiment with different non-overlapping runs. First of all, we retrieve from our full overlapping element index and perform result list based overlap removal as a post processing step. Second, we retrieve from a non-overlapping element index. In our case, we used an index of all `<sec>` elements. Last, and perhaps least, we retrieve from an article index. For the *CO.Thorough* task we experiment with two pruning methods. One based on previous relevance assessments and the other based on the length of the XML elements. We compare the selective indexes to our full overlapping element index. For the *CO.FetchBrowse* task we use our CO.Focussed runs as a basis and simply group results for each article.

For the different *CO+S* tasks we experiment with different extents to use the structural constraints. First of all, we only use the target constraint. Second we use only the field constraints. Third we use both target and field constraints. For the CO queries which do not have a structural version, we introduce structured constraints using pseudo relevance feedback. For efficiency reasons, our system is restricted to handle a limited set of structural constraints. The appropriate set of structural constraints is chosen by studying content-and-structure queries of previous years..

This paper is further organized as follows. In Section 2 we introduce our indexing schemes. We describe our CO and CO+S runs in Sections 3 and 4 respectively. Section 5 gives a summary of our results. Finally we provide some discussion and conclusions in Section 6.

2 Indexing

For effective and efficient XML retrieval indexing plays an important role. Any element can, in theory, be retrieved. It has been shown, however, that not all elements are equally likely to be appreciated as satisfactory answers to an information need [2]. In particular, retrieval of the very many, very small elements is not likely to be rewarded by users. Furthermore, users (and hence metrics) may be willing to partially reward near misses. This prompts us to investigate whether we can reduce our indexing size, both in terms of retrievable units and storage size. We believe that this gives us more efficient retrieval without loosing any, or at least little, of retrieval effectiveness.

Element Indexes For retrieving elements we build four indexes.

- *Element index* We build the "traditional" overlapping element index in the same way as we've done in the previous years (see further [4, 5]).
- *Length based index*: It has been shown that very short elements are not likely to be regarded as relevant. We analyze the average length of elements bearing different tag-names. We then index only element types having an average length above a certain threshold. For INEX 2005 we set the threshold to be 25 terms. The term count was applied before stop-words were removed.
- *Qrel based index*: It has been shown that elements with certain tag-names are more likely than others to be regarded as relevant. We analyze the assessments and look at which elements are assessed more frequently than other. We index only elements that have appeared relatively frequently in previously assessment sets (i.e., they should constitute at least 2% of the total assessments). We index `article`, `bdy`, `sec`, `ss1`, `ss2`, `p`, `ip1`, and `fig`.
- *Section index*: Retrieval of non-overlapping elements is a hot topic in XML retrieval. We want to investigate how simple you can make your non-overlapping retrieval. We build an index based on non-overlapping passages, where the passage boundaries are determined by the structure. The simplest solution is to index only sections (`<sec>`). We believe that this simple strategy is effective, despite (due to) the fact that the sections do not provide a full coverage of the collection.

Article Indexes For retrieving articles we build two indexes.

- *Article index*: the "normal" article index
- *Query fields*: An article index containing both content and a selection of fields. The fields are chosen based on structure of previous structured queries. The fields chosen for INEX 2005 were: `abs`, `fm//au`, `fm//atl`, `kwd`, `st`,

Table 1. Properties of the the different indexes. *Unit* stands for the number of retrievable units. *Storage* stands for the size occupied in physical storage. *Query time* stands for the time needed to retrieve 1000 retrieval units from the index for each of the INEX 2005 topics. All retrieval times are relative to the maximum retrieval time. (This table will be completed in the proceedings version of this paper.)

Index	Units	Storage	Query time
Element index	10,629,617	1.9G	1.0
Length based	1,502,277	1.3G	t.b.a.
Qrel based	1,581,031	1.1G	t.b.a.
Sections	96,600	223M	t.b.a.
Articles	16,819	204M	t.b.a.
Query fields	16,819	275M	t.b.a.

`bb//au`, `bb//atl`, and `ip1`. The fields were chosen from a set of fields that were used in the INEX 2003 and INEX 2004 content-and-structure queries.

For all indexes, stop-words were removed, but no morphological normalization such as stemming was applied. Table 1 shows some statistics of the different indexes.

3 Content-Only Runs

For all our runs we use multinomial language model [1]. We use the same mixture model implementation as we used in INEX 2004 [5]. We assume query terms to be independent, and rank elements according to:

$$P(e|q) \propto P(e) \cdot \prod_{i=1}^{k} P(t_i|e), \tag{1}$$

where q is a query made out of the terms $t_1, \ldots, t_k$. We estimate the element language model by taking a linear interpolation of three language models:

$$P(t_i|e) = \lambda_e \cdot P_{mle}(t_i|e) + \lambda_d \cdot P_{mle}(t_i|d) + (1 - \lambda_e - \lambda_d) \cdot P_{mle}(t_i), \tag{2}$$

where $P_{mle}(\cdot|e)$ is a language model for element e; $P_{mle}(\cdot|d)$ is a language model for document d; and $P_{mle}(\cdot)$ is a language model of the collection. The parameters λ_e and λ_d are interpolation factors (smoothing parameters). Finally, we assign a prior probability to an element e relative to its length in the following manner:

$$P(e) = \frac{|e|}{\sum_e |e|}, \tag{3}$$

where $|e|$ is the size of an element e.

3.1 CO.Focused

In our focused task we experiment with two different ways of choosing focused elements to retrieve. First, based on the hierarchical segmentation of the collection. Second, based on a linear segmentation of the collection. We also wanted to compare these two approaches with a non-focused baseline, namely a document retrieval system. We submitted three runs:

- *Article run* (UAmsCOFocArticle) A baseline run created using our article index. We used a $\lambda = 0.15$ and a normal length prior.
- *Element run* (UAmsCOFocElements) A run created using a mixture model of the overlapping element index and the article index. We set $\lambda_e = 0.4$ and $\lambda_d = 0.4$. No length prior was used for this run. Overlap was removed in a list-based fashion, i.e. we traversed the list from the most relevant to the least relevant and threw out elements overlapping with an element appearing previously in the list.
- *Section run* (UAmsCOFocSections) A run created using a mixture model of the section index and the article index. We set $\lambda_e = 0.05$ and $\lambda_d = 0.1$. A normal length prior was used.

3.2 CO.Thorough

The main research question is to see if we can get away with indexing only a relatively small number of elements. In our runs we compare three element indexes. The "normal" element index, the qrel-based element selection and the length-based element selection. We submitted three runs:

- *Full element run* (UAmsCOTElementIndex) A run using a mixture model of the full element index and the article index. We set $\lambda_e = 0.05$, $\lambda_d = 0.1$, and used a normal length prior.
- *Qrel-based run* (UAmsCOTQrelbasedIndex) A run using a mixture model of the qrel-based element index and the article index. We set $\lambda_e = 0.05$, $\lambda_d = 0.1$, and used a normal length prior.
- *Length-based run* (UAmsCOTLengthbasedIndex) A run using a mixture model of the length-based element index and the article index. We set $\lambda_e = 0.05$, $\lambda_d = 0.1$, and used a normal length prior.

3.3 CO.FetchBrowse

For the fetch and browse we mirror the focused task submissions, but cluster the results so that elements within the same article appear together.

- *Article run* (UAmsCOFBArticle) This run is exactly the same as the article run we submitted for the focused task.
- *Element run* (UAmsCOFBElements) We took the focused element run and reordered the results in such a way that elements from the same document are clustered together. The document clusters are ordered by the highest scoring element within each document. We returned a maximum of 10 most relevant elements from each article.

- *Section run* (UAmsCOFBSections) We took the focused section run and reordered the result set in such a way that the elements from the same document are clustered together. The document clusters are ordered by the highest scoring section within each document.

4 Content-Only with Structure Runs

For the CO+S task we experiment with three ways of using structural constraints.

Target-only For queries that have a CAS title we only return elements which satisfy the target constraint of the CAS title. For queries that ask for sections, we accept the equivalent tags as listed in the topic development guidelines. NB! We use the terms in the title field of the queries because we want a direct comparison to CO runs. Retrieval is performed using a mixture model using the overlapping element index and the normal document index.

Fields-only Here we use the document index with query fields. We process the queries in three different ways, depending on their format. First, for the the `<castitle>` queries with field constraints that match our fielded article index, we rewrite the query such that it fits our index. For example, the query:

```
//article[about(.//abs, ipv6)]//sec[about(., ipv6 deployment) or
about(., ipv6 support)]
```

becomes

```
abs:ipv6 ipv6 deployment ipv6 support.
```

For the queries that only partly match our indexing scheme, we do additional processing, i.e.

```
//*[about(.//au, moldovan) and about(., semantic networks)]
```

becomes

```
fm//au:moldovan bb//au:moldovan semantic networks
```

since our index makes distinction between article authors and referenced authors. Second, for `<castitle>` queries that do not have fields that fit our index, we use the simply extract the query terms. I.e.

```
//article[about (.//bdy, synthesizers) and about (.//bdy, music)]
```

becomes

```
synthesizers music.
```

Third, for queries that do not have a `<castitle>`, we add structured query fields using pseudo relevance feedback on the fielded article index [3]. We look at the top 20 feedback terms and we add up to n fielded terms where n is the length of the original query. For example,

```
computer assisted composing music notes midi
```

becomes

```
bb//atl:music bb//atl:musical st:music ip1:musical ip1:music
fm//au:university computer assisted composing music notes midi
```

We use those queries to create an article run using the fielded article index. Now we do the following:

- We take an existing run and for each element in that run, we replace it's score with the score of it's article (using the fielded index and fielded queries).
- We do a combSUM of the original element run and the "article score" element run.

Target and Fields Constraints Here we process both the target and fields constraints in the same ways as discussed above.

4.1 Runs

The ways of processing structural constraints discussed above, are applied to each of the structured retrieval tasks.

+S.Focused

- *Strict on target* (UAmsCOpSFocStrictTarget) A run created using a mixture model of the overlapping element index and the article index. We set $\lambda_e = 0.4$ and $\lambda_d = 0.4$. No length prior was used for this run. Target restriction was implemented for queries that had one. Overlap was removed in a list-based fashion
- *Using constraints* (UAmsCOpSFocConstr) We apply the fields-only approach, described above, on the focused CO element run (UAmsCOFocElements).
- *Using constraints and strict on target* (UAmsCOpSFocConstrStrTarg) We apply the fields-only approach on the strict on target run (UAmsCOpSFoc-StrictTarget).

+S.Thorough

- *Strict on target* (UAmsCOpSTStrictTarget) A run created using a mixture model of the overlapping element index and the article index. We set $\lambda_e = 0.05$ and $\lambda_d = 0.1$. We apply a normal length prior. Target constraints are respected for queries that have one.

- *Using constraints* (UAmsCOpSTConstr) We apply the fields-only approach, described above, on the thorough CO element run (UAmsCOTElementIndex).
- *Using constraints and strict on target* (UAmsCOpSTConstrStrTarg) We apply the fields-only approach on the strict on target run (UAmsCOpSTStrictTarget).

+S.FetchBrowse

- *Strict on target* (UAmsCOpSFBStrictTarget) We reorder the focused strict on target run (UAmsCOpSFocStrictTarget) such that results from the same article are clustered together. Only the 10 most relevant elements are considered for each article.
- *Using constraints* (UAmsCOpSFBConstr) We reorder the focused run using constraints (UAmsCOpSFocConstr) such that results from the same article are clustered together. Only the 10 most relevant elements are considered for each article.
- *Using constraints* (UAmsCOpSFBConstrStrTarg) We reorder the focused run using constraints and strict targets (UAmsCOpSFocConstrStrTarg) such that results from the same article are clustered together. Only the 10 most relevant elements are considered for each article.

5 Results

In this section we will present and discuss our preliminary results. The results for the Focussed and Thorough tasks are based on results from the INEX (lip6) website as they appeared on November 6th, 2005. The results for the FetchBrowse were taken from the website in November 10th, 2005.

5.1 The Focussed Task

Table 2 shows the results for both the CO.Focussed and CO+S.Focussed runs.

CO.Focussed The aim of the CO.Focussed submission was to compare 3 non-overlapping retrieval strategies: element retrieval, section retrieval and article retrieval. As we see in Table 2 the element based retrieval outperforms the other two for almost all metrics. There are, however, some notable exceptions. Interestingly, the article run outperforms the other two when we look at extremely early precision of the generalized nxCG metric. For the generalized extended Q and R metric the section run gives the best performance, followed by the article run. This suggests that the element retrieval approach is a good approach when considering the full recall base. However, for early precision, section retrieval (and to some extent article retrieval) is a better alternative.

Table 2. Results for the CO.Focussed and COS.Focussed runs using various metrics

(a) nxCG (overlap=on, generalized)

Run	MAnxCG	@1	@2	@3	@4	@5	@10	@50	@100
Elements	.269	.195	.191	.221	.204	.209	.200	.165	.181
Sections	.204	.213	.209	.190	.194	.178	.176	.158	.153
Articles	.098	.248	.250	.205	.191	.184	.170	.102	.089
StrTarg	.225	.236	.230	.246	.236	.230	.224	.168	.168
Constr	.300	.161	.215	.224	.232	.215	.203	.170	.190
ConStrTar	.237	.289	.272	.257	.250	.231	.224	.182	.175

(b) EP/GR (overlap=on, generalized) and Extended Q and R (generalized)

	EP/GR		Ext. Q and R	
Run	iMAep	MAep	Q	R
Elements	.056	.071	.115	.159
Sections	.030	.064	.145	.215
Articles	.020	.048	.133	.195
StrTarg	.049	.072	.135	.202
Constr	.059	.074	.123	.166
ConStrTar	.057	.078	.144	.208

CO+S.Focussed The aim of the CO+S.Focussed submission was to compare different extents to which the structural constraints can be used. For the averaged metrics the run using only the fields-only approach (Constr) outperforms both the other runs using structured queries, as well as outperforming the runs using no structure at all. For the early precision metrics, the runs using strict-target interpretation (StrTarg and ConStrTarg) outperform the fields-only approach. The run using both field-constraints and target-constraints generally outperforms the run using target-constraints only. It seem thus that the field-constraints are generally useful for improving retrieval effectiveness, while the target constraints are particularly useful for achieving high early precision.

5.2 The Thorough Task

Table 3 shows the results for the CO.Thorough and CO+S.Thorough runs.

CO.Thorough The aim of the CO.Thorough submission was to experiment with two selective indexing approaches compared to a full element retrieval approach. Table 3 shows that for the averaged metrics, the selective indexing approaches to not have substantially worse performance than the full element retrieval approach. Furthermore, for the early precision metrics, the selective indexing approaches outperform the full element retrieval approach.

CO+S.Thorough The aims and the results for the CO+S.Thorough task are the same as for the CO+S.Focussed task. Field-constraints are useful over-all, but target constraints improve initial precision.

Table 3. Results for the CO.Thorough and CO+S.Thorough runs using various metrics

(a) nxCG (overlap=off, generalized)

Run	MAnxCG	@1	@2	@3	@4	@5	@10	@50	@100
Element	.309	.239	.191	.256	.265	.275	.265	.230	.218
Qrel	.301	.266	.227	.293	.287	.289	.275	.245	.231
Length	.302	.281	.247	.281	.272	.276	.264	.240	.218
StrTarg	.192	.322	.260	.263	.245	.246	.225	.186	.168
Constr	.334	.242	.244	.249	.250	.254	.261	.234	.246
ConStrTar	.206	.311	.264	.246	.242	.235	.216	.198	.180

(b) EP/GR (overlap=off, generalized) and Extended Q and R (generalized)

	EP/GR		Extended Q and R	
Run	iMAep	MAep	Q	R
Element	.072	.085	.162	.269
Qrel	.074	.089	.168	.275
Length	.070	.085	.166	.272
StrTarg	.046	.053	.098	.193
Constr	.078	.089	.166	.270
ConStrTar	.049	.056	.101	.193

Table 4. Results for the CO.FetchBrowse and CO+S.FetchBrowse runs using various metrics

(a) ERPRUM (ideal: GK-SOG, quant: Exh, behaviour: Hierarchic)

Run	Average	@1	@5	@10	@20	@100
Elements	.091	.089	.039	.025	.016	.005
Sections	.114	.123	.056	.031	.019	.005
Articles	.027	.072	.033	.021	.012	.003
StrTarg	.107	.098	.039	.026	.016	.005
Constr	.064	.106	.047	.029	.019	.006
ConStrTar	.070	.116	.044	.027	.017	.005

5.3 The Fetch-and-Browse Task

FetchBrowse Table 4 shows the results for our FetchBrowse submissions. At the time of writing, none of the FetchBrowse tasks have been evaluated with an official INEX metrics which handled overlap. Since we submitted only non-overlapping runs for this task, we report the EPRUM metric which takes overlap into consideration. Our section retrieval run outperforms the full element retrieval run both w.r.t. average precision and initial precision. It is interesting to note that for the Focussed task, the element run outperformed the section run for the averaged metric. Note, however, that there are several crucial differences between the results for the two tasks. First of all, the task is of course different. Second, the tasks are evaluated with different metrics. Third, for articles where more than 10 results were found, results 11 and above were removed from the FetchBrowse runs.

Table 5. Results for the CO.FetchBrowse-D and CO+S.FetchBrowse-D runs using various metrics

(a) inex_eval

		generalized				
Run	MAP	@1	@5	@10	@20	@100
Element	.186	.261	.207	.183	.154	.064
Sections	.264	.359	.289	.216	.163	.064
Article	.282	.424	.285	.260	.202	.069
StrTarg	.186	.293	.189	.171	.128	.057
Constr	.242	.315	.239	.216	.175	.068
ConStrTar	.250	.326	.265	.225	.178	.066

FetchBrowse-D Table 5 shows the document-run evaluation of our FetchBrowse submissions. The results seem to indicate that element retrieval is not an effective strategy for improving document retrieval. These results will be analyzed further in the proceedings version of this paper.

6 Conclusions

In INEX 2005 we set out to investigate several research questions.

- Does retrieval using the XML tree hierarchy improve significantly over using a simpler linear segmentation of the documents?
- How do different types of structural constraints contribute to improved retrieval effectiveness?
- Can we prune our overlapping element index to gain efficiency without loosing effectiveness?
- Can we create structured queries automatically using pseudo relevance feedback?

Our results show that retrieving from the full hierarchy of element outperforms retrieval from a linear segmentation. The segmentation based retrieval is however competitive when we look at initial precision. Article retrieval is interestingly effective at P@1 and P@2 for the CO.Focused task.

We showed that fielded constraints are helpful for improving average retrieval performance. Interpreting target constraints in a strict manner does hurt average performance. The target constraints do however improve retrieval when we look at early precision.

For the thorough task we experimented with two different pruning of the full overlapping element index. Neither of the pruning strategies lead to a considerably lower average performance. Both pruning strategies did however lead to improved initial precision. The length based pruning lead to greater improvement than the qrel based pruning.

For the FetchBrowse task the linear segmentation performed considerably better than the hierarchical segmentation. This result is different from the Focussed task. It is however not clear whether this difference lies in different task performed or in the different metric used to evaluate the two tasks.

For the document ranking part of the FetchBrowse task, ranking documents based on their own retrieval score outperformed the document retrieval based on the highest scoring element/section.

At this point we have not evaluated the effect of automatically creating structured queries through pseudo relevance feedback. This analysis remains as future work and will be carried out before publication of the proceedings version of this paper.

Future work includes looking at different granularities for the linear segmentation. Instead of looking at section level, we could look at using subsections, when available. Also we should include elements such as front-matter so that the linear segmentation has a full coverage of the collection.

Our processing of the CO+S queries is a bit ad-hoc. Future research should include a cleaner way of processing and retrieving using structural queries.

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 612-13-001, 612.000.106, 612.000.207, 612.066.302, 612.069.006, and 640.001.501.

References

1. D. Hiemstra. *Using Language Models for Information Retrieval.* PhD thesis, University of Twente, 2001.
2. J. Kamps, M. de Rijke, and B. Sigurbjörnsson. The importance of length normalization for XML retrieval. *Information Retrieval*, 8:631–654, 2005.
3. J. Ponte. Language models for relevance feedback. In W. Croft, editor, *Advances in Information Retrieval*, chapter 3, pages 73–96. Kluwer Academic Publishers, Boston, 2000.
4. B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-Based Approch to XML Retrieval. In *INEX 2003 Workshop Proceedings*, pages 19–26, 2004.
5. B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Mixture models, overlap, and structural hints in XML element retreival. In *Advances in XML Information Retrieval*, LNCS 3493, pages 196–210, 2005.

Searching XML Documents – Preliminary Work

Marcus Hassler and Abdelhamid Bouchachia

Dept. of Informatics, Alps-Adria University Klagenfurt, Austria
marcus.hassler@uni-klu.ac.at, hamid@isys.uni-klu.ac.at

Abstract. Structured document retrieval aims at exploiting the structure together with the content of documents to improve retrieval results. Several aspects of traditional information retrieval applied on flat documents have to be reconsidered. These include in particular, document representation, storage, indexing, retrieval, and ranking. This paper outlines the architecture of our system and the adaptation of the standard vector space model to achieve focussed retrieval.

1 Introduction and Motivation

Traditionally, content-based retrieval systems rely either on the Boolean model or the vector space model (VSM) [1–3] to represent the flat structure of documents as a bag of words. Extensions of these models have been proposed, e.g., the fuzzy Boolean model and knowledge-aware models. However, all of these indexing models do ignore the organization of text and the structure of documents until recently with the advent of "queriable digital libraries".

XML documents have a standard structure defined by a DTD or XML schema. While this structure provides documents with hierarchical levels of granularity, and hence more precision can be achieved by means of focussed retrieval, it does, however, imply more requirements on the representation and retrieval mechanisms. With the new generation of retrieval systems, the two aspects, the structure and the content, have to be taken into account. To minimally achieve that in presence of nested structure like chapter-section-subsection-paragraph, the traditional information retrieval techniques, e.g., the VSM, have to be adapted to fit the context of structure-aware retrieval. To design such systems, four basic aspects are of high importance:

(a) Representation: Textual content of the hierarchically structured documents is generally restricted to the leave nodes. Hence, representation mechanisms of the inner nodes content have to be defined.

(b) Retrieval granularity: A basic question is whether the indexing/retrieval unit must be known ahead of time or is completely dynamically decided by the user or eventually by the system itself.

(c) Ranking: Related to the first two aspects, a strategy for ranking the retrieved results has to be decided.

(d) Result presentation: The way results are presented is a key issue [4–6] and has to be considered early in the design of the system. Once ranked, the results are displayed showing their context of appearance. Further functionality enabling browsing is required.

Taking these aspects into account, we developed a retrieval system. It is fully implemented in Java and consists of three modular subsystems: indexing, retrieval and RMI (Remote Method Invocation) communication server as depicted in Fig. 1.

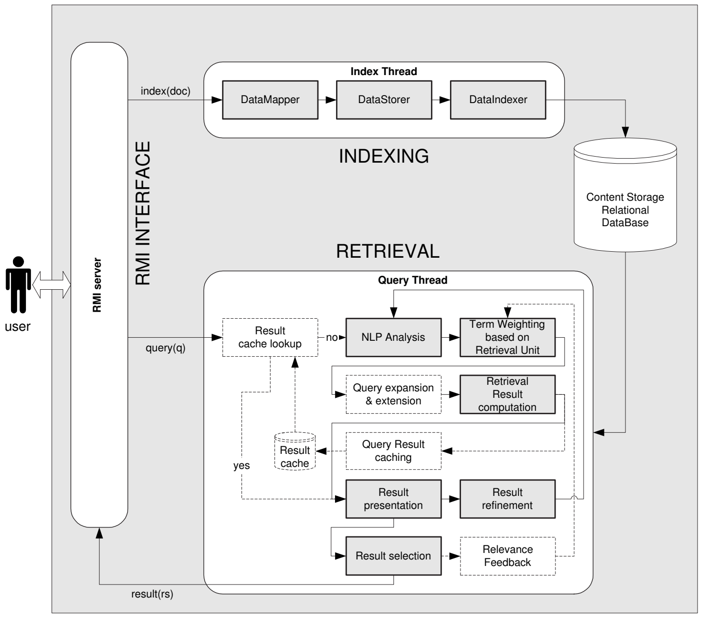


Fig. 1: Architecture of the system

The RMI server takes incoming requests for indexing and querying the system and initiates a new thread for each call. The basic motivation behind this is to achieve some degree of parallelism. The maximum number of parallel threads depends on the performance of the hardware. From the software architecture point of view, both index and query subsystem, use a pipelined pattern of processing units (Fig. 1). Dashed components describe planned extensions. For portability and tuning purposes, all subsystems are independently configurable via config files. During indexing, documents are transformed into our XML schema (DataMapper), stored in the database (DataStorer), and indexed for retrieval (DataIndexer). As soon as a query is sent to the system it is analyzed by a query thread. Documents in the database are matched against the query and relevant elements[1] are ranked in decreasing order.

[1] we use element and node interchangeable

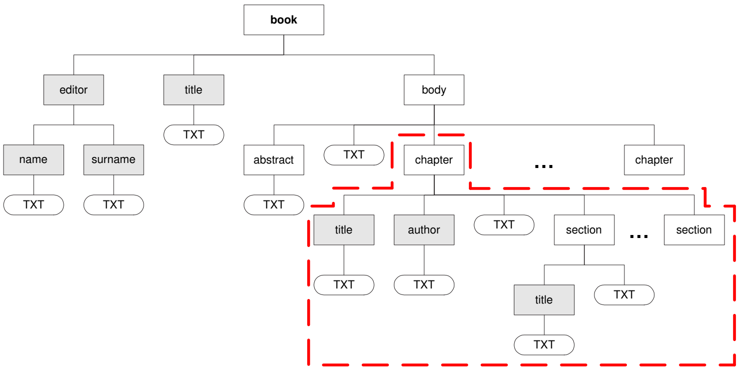


Fig. 2: Example XML document

In this paper, we will discuss the aspects (a)–(d), but with more focus is more on the representation and the indexing/retrieval problem. First, in Sec. 2, a generic schema for document representation is presented, onto which the XML documents are mapped. Section 3 describes the underlying database model used for storing the content and the corresponding representation. The most interesting issues namely indexing and retrieval are discussed in Sec. 4 and Sec. 5 respectively. Section 6 concludes the paper.

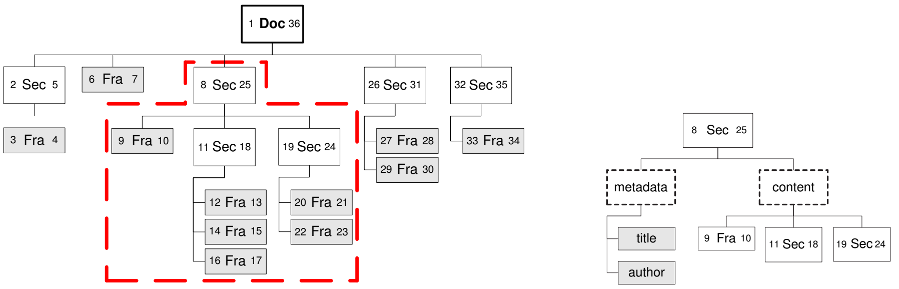
2 Document Structure

The hierarchical structure for the content of documents is usually described by means of a set of tags (e.g. chapter, section, subsection, etc.), as shown in Fig. 2[2]. In order to represent a collection of documents having different structure, we apply an XSLT transformation to derive a common document format (schema). This step eliminates structural ambiguities and resolves semantic relativism [7].

As illustrated in Fig. 3a, we introduce a general document format (defined through XML schema) that consists of only three main elements: `DOC` (document), `SEC` (section) and `FRA` (fragment). The `DOC` element defines the root of the document. `SEC` is the basic structural element of a document. By recursively defining `SEC` (e.g., section) as either containing raw content `FRA`s (e.g., paragraphs) and/or made up of other `SEC`s (e.g., subsections), the depth of nested structures is unlimited. To define smallest retrievable units for indexing and retrieval, we rely on fragments (`FRA`s). They stand for the leaf nodes in our document schema (see Fig. 3a).

A node in an XML document is viewed as a tuple (*metadata*,*content*), where *metadata* refers to descriptive information of the node itself, while *content* refers to the segments content, properly said (see Fig. 3b). Generally, the first type of nodes requires database-supported matching during retrieval, while the second type is subject to partial matching (VSM).

[2] This example will be used throughout this paper



(a) Transformed example

(b) Metadata and Content blocks

Fig. 3: XML document representation

2.1 Metadata

In addition to the content block, the metadata block of a node contains information describing that node. Examples of metadata are `author`, `year` and `keywords` for a `DOC` or the `title` for a `SEC` element. The fragment metadata block is used to describe its actual content by means of `content_type`, `language`, and possibly `title` (figures, tables).

To allow a semantic interpretation of the content of an element, a type hierarchy is proposed by Gövert [8]. An extension of the proposed type hierarchy for metadata is depicted in Fig. 4. There, types are derivated from a common base element. The first level in the hierarchy (bold) corresponds to database supported data types. Further types in subsequent levels in the hierarchy have one of the basic database types as supertype (e.g., `PersonName` is a String). In addition, data types predicates for comparison are defined. This allows to process section titles, phone numbers, and author names for instance.

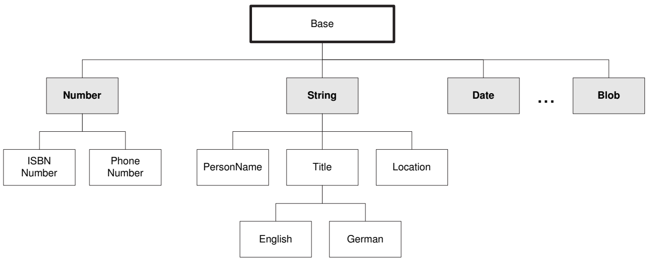


Fig. 4: Hierarchical metadata types

2.2 Content

Generally, the content block of DOCs and SECs are defined as ordered lists of further (sub)SECs and FRAs. The content block of FRAs is defined as bytecode or empty. For indexing and retrieval purposes, content is interpreted based on its type (metadata). Defining a fragment's content as text block (paragraph) only might be too restrictive. Therefore, a fragment in our sense refers to paragraphs, enumerations, lists, figures, tables, formulas, images, sounds, videos, definitions, theorems, etc. On the other hand, a fragment (FRA) defines the smallest retrievable unit of a document. It can be understood as building block (elementary content container). However, the granular unit is application specific and can be set at wish to fit sentences as well as the whole text of a chapter.

From the structuring point of view, additional markup within a FRA's content might be needed. Our schema supports mathematical environments (using MATH) and two types of links (using LINK), internal and external links. Internal links are links within the same document, e.g., citations, figure/table references within the text and the table of contents. External links refer to external resources, including reference entries in the references section, references to email/internet addresses and file references.

While the content block in DOCs and SECs is mandatory, in FRAs it is not. This allows to include external content by its metadata only. An external source attribute within the metadata block can be used to refer to the content somewhere else (e.g., a picture file). In contrast to SEC elements, which define their own context based on their path, e.g., /DOC/SEC/SEC, fragments define a separate context. From the indexing and retrieval point of view, a fragment in a section lies within the same context as a fragment in a chapter or subsubsection. This difference is important in the context of a dynamic term space, discussed in Sec. 5.3.

3 Storage

For efficiency purposes, we use a relational database to store the XML documents. The goal is to accelerate the access to various structural neighbors of each node in the document (descendants, ancestors, and siblings). Being a tree, an XML document can easily and unambiguously traversed. Therefore, each node is represented by its document ID and preorder/postorder. We depart from the idea of *preorder* and *postorder* introduced in [9, 10], supporting non-recursive ancestor/descendant detection and access. Table 1 shows an excerpt of the structural information of a document representation. Likewise, we designed another for the corresponding content.

A structural entry is described by the tuple (*docID*, *pre*, *post*, *parentID*, *tagID*, *pathID*). The root element has $pre = 1$ and $parentID = 0$ (no parent) per definition. The attribute *tagID* is included for fast name lookup and access. For the sake of performance, we added the elements path (XPath without positional information) *pathID* to circumvent recursive path generations by using the *parentID* relation.

Table 1: Structural entries

docID	*pre*	*post*	*parentID*	*tag*	*path*
d_1	1	36	0	`Doc`	`/Doc`
d_1	2	5	1	`Sec`	`/Doc/Sec`
d_1	3	4	2	`Fra`	`/Doc/Sec/Fra`
d_1	6	7	1	`Fra`	`/Doc/Fra`
d_1	8	25	1	`Sec`	`/Doc/Sec`
d_1	9	10	8	`Fra`	`/Doc/Sec/Fra`
d_1	11	18	8	`Fra`	`/Doc/Sec/Sec`
d_1	12	13	11	`Fra`	`/Doc/Sec/Sec/Fra`
d_1	14	15	11	`Fra`	`/Doc/Sec/Sec/Fra`

The content of nodes (in particular leaf nodes) is stored in a separate table, as suggested in [11]. However, the content of inner nodes can always be recovered from their descendants as will be discussed in Sec. 4. Note that some content entries do not have a corresponding representation entry (e.g. figures, tables).

To improve retrieval performance, metadata handling is completely shifted to the database. This is achieved by grouping all metadata according to its element. Instead of having multiple structural and content entries, a single row (*docID*, *pre*, $meta_1$, ..., $meta_n$) is used to store all metadata together. Metadata of nodes (`DOC`, `SEC`, `FRA`) are stored in separated but very similar tables as shown in Tab. 2 for the case of sections. The reason of supporting only a single set of `SEC` metadata is that all `SEC` elements (chapters, sections, subsections, etc.) are assumed to have quite homogenous metadata (e.g., `title`). Although this may lead to some '*NULL*' values (unavailable metadata for some elements) in the database, the whole set can be accessed by a single select statement. This simplifies and speeds up querying of metadata considerably.

Table 2: Metadata entries for `SEC`)

docID	*pre*	*title*	*author*	...
d_1	2	`Introduction`	`R. Smith`	
d_1	8	`XMl Retrieval`	`J. Alf`	
d_1	11	`Granularity`	*NULL*	

A global view is depicted in Fig. 5. Both, metadata and content entries, are optional. Additional types of representations (e.g. semantic concepts, figure representations, etc.) can easily be integrated.

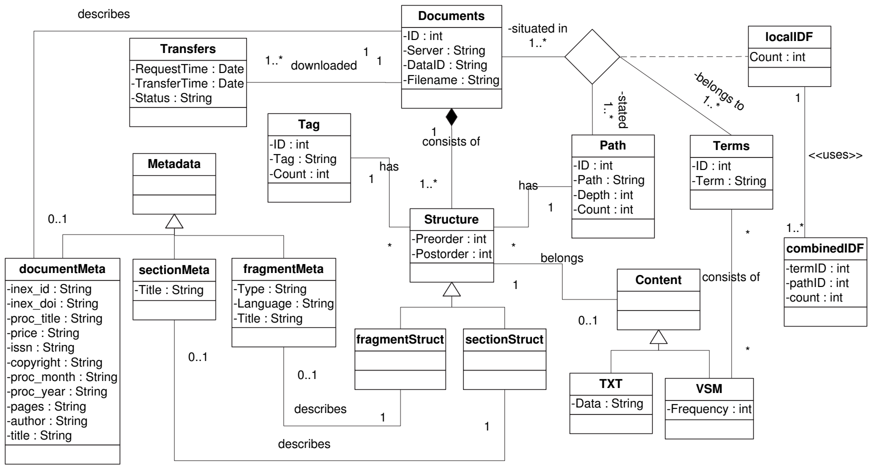


Fig. 5: Conceptual database schema

4 Indexing

To represent texts as a vector of terms and their term frequencies, our natural language processing (NLP) involves several subtasks containing tokenization, tagging, term extraction, stemming, filtering and term frequency calculation. Our implementation is based on abstract components. Taking advantage of the the modularity aspect, different implementations of the same component can be instantiated and selected during runtime. Hence, our system can easily be adapted to process documents in other languages. Our prototype also involves ready made-components like the tagger, and the stemmer.

During the indexing process, only the content of leaf nodes need to be parsed. Their representation, a term frequency vector, is stored in the database (`VSM` table). Consequent updates of the `localIDF`, `combinedIDF` table, and `Terms` table are immediately done. These update operations are also carried out during re-indexing or removal of documents.

The index of inner nodes is obtained by simply merging the index of its descendants. This is done by summing up their term frequencies. This operation is equivalent to process the concatenated contents of the descendant nodes. It is also possible to store the result of the merge operation so that no index computation is required later during the retrieval process. This reduces search time, but increases the size of the database. It is important to stress that the weight vectors are computed during retrieval using the available term frequency vectors.

We define the *context of a node* as the set of all elements having the same path (all chapters, all sections, etc.). In order to dynamically characterize both, the granularity during indexing and retrieval, we applied a propagation of term statistics (e.g. tf), in contrast to the weight propagation methodology [12]. In addition, the inverse document frequency (*idf*) for each node is calculated dynamically based on the node's context. Term weights are computed based on the term frequencies and the *idf* in this context. This allows to perform focussed retrieval on any level in the document tree. To achieve that in a given context, tf of all nodes lying at this level will require tfs of their descendants. Using term statistic propagation, the descendants' tf are simply summed up. We avoid recursive data accesses by exploiting preorder and postorder of document elements (only one SQL select statement).

As to term weighing, we use different $idf_{j,c}$s of the same term j in different contexts c. This strategy weighs the same term with the same term frequency differently depending on c (e.g. chapter vs. subsection). Clearly our approach puts more attention on the actual context during retrieval. If the unit of retrieval is defined explicitly, elements in this context are focussed and compared only among them. Representations of elements in other contexts do not influence the result.

To implement this idea, we use two tables (see Fig. 5): a table `localIDF` stores tuples of the form ($docID$, $pathID$, $termID$, n_j), where n_j refers to the number of elements containing term $termID$ in the path $pathID$ within a document $docID$. Consider the example given in Tab. 3, the first Tab. 3a indicates that the term "car" occurs twice in `/DOC/SEC` nodes of document d_1. To calculate the $idf_{j,c}$ of a term j in a context c, we have to define N_c and n_j. N_c is the number of nodes with $pathID = c$. N_c can simply be derived via the table holding the structural entries (see Tab. 1). n_j is given by counting the rows containing $pathID = c$ and $termID = j$. In the above example, this results in an inverse document frequency for the term "car" in the node `/DOC/SEC` of $idf_{car,/DOC/SEC} = \log \frac{3}{2}$. This definition of $idf_{j,c}$ leads to different idfs in different contexts.

Table 3: *idf* calculation

(a) Table `localIDF`

docID	*path*	*term*	n_j
d_1	`/DOC/SEC`	car	2
d_1	`/DOC/SEC/SEC`	mouse	1
d_2	`/DOC/SEC`	car	1
d_2	`/DOC/SEC/SEC`	dog	1
d_2	`/DOC/SEC/SEC`	mouse	3
d_3	`/DOC/SEC`	water	1
d_3	`/DOC/SEC/SEC`	dog	2
d_3	`/DOC/SEC/SEC`	dog	2

(b) Table `combinedIDF`

path	*term*	n
`/DOC/SEC`	car	3
`/DOC/SEC`	water	1
`/DOC/SEC/SEC`	mouse	4
`/DOC/SEC/SEC`	dog	3
`/DOC/SEC/SEC`	frog	1

Since Tab. 3a is quite large, we introduced a summarized shortcut-table `combinedIDF` Tab. 3b with the overall goal to reduce the time access to *idf* values. Same paths associated with the same terms are precalculated (e.g. term "car"). For the sake of dynamic document environments (adding, removing and re-indexing), we still need the information provided by Tab. 3a to adjust the n values correctly. In addition, all N_c values, the numbers of elements with the same path, are stored in the `Path` table (see Fig. 5).

Given a particular context (e.g. `/DOC/SEC`), our indexing strategy allows on-the-fly computation of the representations associated with these nodes (considered as documents). Hence, our indexing method stores only term frequency vectors in the database; weight computation is totally executed on the fly during the retrieval process. The advantages of this methodology are:

- It behaves exactly like the traditional models at the document level.
- There is no need for empirical parameters as augmentation weights.
- Elements of smaller granularity do not automatically have sparser feature vectors (leading to smaller similarity), hence they define their own context.
- Documents can dynamically be added, removed, and re-indexed, without impacting the weights of other representations.

5 Retrieval

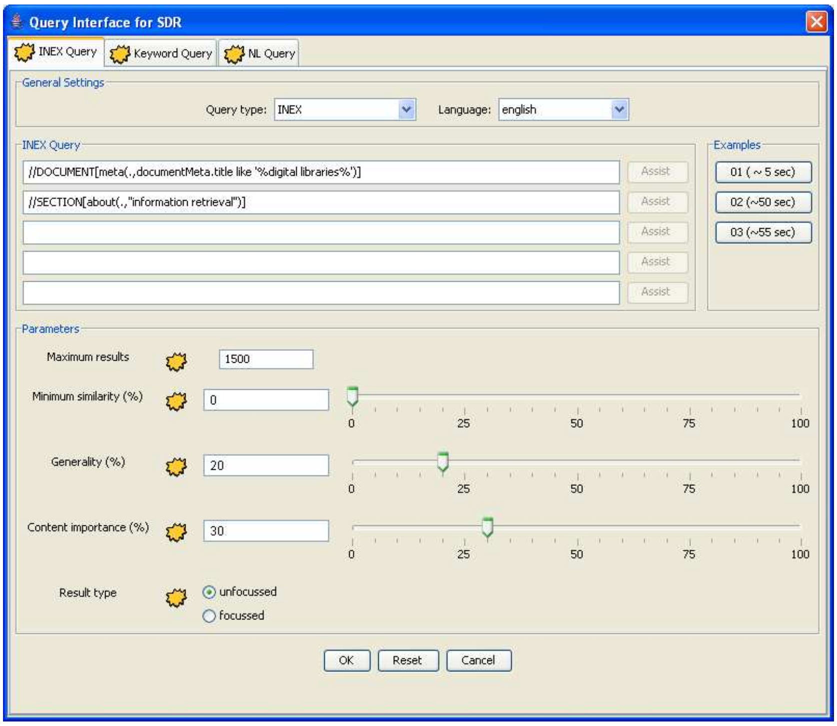
This section explains the retrieval process. In particular, it describes how and which information is required by the system to answer a user query appropriately. This includes formulation of the query, setting of specific parameters, matching, filtering, and presentation of the result.

5.1 Query formulation

The actual query input is done via an input interface which allows to enter different types of queries: `KWD` (keyword) and `NLQ` (natural language query, free text), which are translated into `INEX` queries. The `INEX` query supports NEXI-like inputs. Hence, we distinguish between metadata and content, we adapted our query parser to support both kinds of information. Similar to the `about(path,terms)` syntax, we added a construct: `meta(path,condition)`. This allows us, for example, to efficiently deal with queries like: "return all documents written by Einstein" using the command `//DOC[meta(.,author like '%Einstein%')]`.

In order to avoid long and confusing single-line queries, we use chains of INEX queries. In our opinion, this concept is also closer to the natural way of questioning, by successively refining the list of results. Each subquery result works as a strict filter, allowing only elements of the same or smaller granularity to be retrieved. This improves the performance without skipping searched elements. Furthermore, we use these chains for reweighing elements regarding to a user-defined generality factor (gf), described below. In addition to the INEX-query chains, several query parameters can be specified by the user (see Fig. 6):

- **Maximum results (*$maxRes$*):** Defines the maximum number of returned results ranging from 1 to $MAXINT$.
- **Minimum similarity (*$minSim$*):** Defines the minimum similarity of returned results ranging from 0 to 1, truncating the list of results below a given similarity threshold.
- **Content importance (*ci*):** Defines the importance of the content similarity to calculate the retrieval status value (rsv). This parameter ranges from 0 (only meta similarity) to 1 (only content similarity). The final similarity is computed as $rsv = simCont * ci + simMeta * (1 - ci)$.
- **Generality factor (*gf*):** This parameter ($\in [0, 1]$) influences the retrieval granularity. The higher the parameter, the more importance of first subqueries, computed as $sim_{new} = sim_{old} * gf + sim_{new} * (1 - gf)$.
- **Result type (*rt*):** Defines which kind of results we wish to obtain: thorough or focussed (see Sec. 5.6).



Query type: INEX
Language: english

//DOCUMENT[meta(.,documentMeta.title like '%digital libraries%')]

//SECTION[about(.,"information retrieval")]

Maximum results: 1500

Minimum similarity (%): 0

Generality (%): 20

Content importance (%): 30

Result type: ● unfocussed ○ focussed

Fig. 6: Query Interface

5.2 Search and Retrieval paths

The search path specifies which elements are to be investigated and matched against the current query. In contrast, the retrieval path specifies which elements are to be returned to the user. Generally these two path are equal, e.g. `//SEC[about(.,wine)]`. This means that the retrieval path is always the same or more general as the search path. So first relevant documents, then relevant sections within those documents, and at a last stage relevant fragments within those sections are identified. Difficulties arise when relevant ancestor elements contain smaller elements that are further specified. For instance, a query that retrieves sections containing paragraphs about a certain topic is not easy given the recursive structure that a section can have.

Our parser for NEXI-like queries implements the following strategy: if the searched element satisfies the retrieval path, only the element itself is returned. Otherwise, the closest parent satisfying the retrieval path condition is returned. In all cases, at most one element is retrieved. So a query like `//SEC[about( ./FRA,global warming)]` retrieves all `SEC` elements at any level (retrieval paths) containing `FRA` paragraphs about "global warming". A more complex example is `//(DOC|SEC)[about(./SEC,anything)]`. Here only sections containing sections about "anything" are to be retrieved, not the sections themselves that are about "anything".

5.3 Dynamic term space

In the context of structured documents, the idea of representing elements at different structural levels within the same term space has to be reconsidered. Assume a number of document sections $\mathbb{S} = \{s_1 \dots s_n\}$ containing a set of unique terms $\mathbb{T}_s$ and a set of chapters $\mathbb{C} = \{c_1 \dots c_m\}$ containing a set of unique terms $\mathbb{T}_c$. Note the implicit relation between term space $\mathbb{T}_s$ and term space $\mathbb{T}_c$: $\mathbb{T}_s \subseteq \mathbb{T}_c$. Let q be a query containing terms $\mathbb{T}_q$ addressing sections $\mathbb{S}$ and chapters $\mathbb{C}$. To calculate the similarity $sim(s_i, q)$ between a section and a query, both feature vectors have to be within the same term space. The same thing holds for comparing chapters and the query $sim(c_i, q)$.

Neglecting the context, sections and chapters are represented in the same (global) term space. As a consequence, the feature vectors of low level nodes become sparser and their similarities compared to nodes of higher levels drop. To overcome this problem, we adopted the concept of a "dynamic term space". In contrast to the global term space, and following the concept of context, nodes in the same context generate a term space. Using a static term space improves performance, but unfortunately decreases the similarity of low-level nodes compared with higher ones. Reducing zero weighted elements in the feature vectors leads to higher precision during the match of low-level nodes. The number of different indexing representations (different contexts) is expected to be quite limited. For instance, the mapped INEX collection does not exceed six structural levels (`/DOC/SEC/SEC/SEC/SEC/FRA`). During retrieval the term space for each context is constructed once, so retrieval performance drops insignificantly.

5.4 Result computation

INEX queries are stated using keywords in the `about(path,`kwd_1 kwd_2 `...`kwd_n`)` syntax. This syntax allows to express several different semantics of keywords that have to be considered:

- `information retrieval techniques`
- `+information +retrieval techniques`
- `information retrieval -techniques`
- `"information retrieval" techniques`
- `+"information retrieval" techniques`

'`+`' (MUST) and '`-`' (CANNOT) indicate whether a term has to be or should not be present in an element. Based on this, a fast preselection is systematically done on candidate elements. Hence, index terms are stemmed, also these terms have to be for comparison.

More complex is the treatment of quoted keywords. Are the keywords `books` and `"books"` equivalent? This depends on whether `"books"` should occur as it is (noun in plural form), or should it be stemmed and treated so.

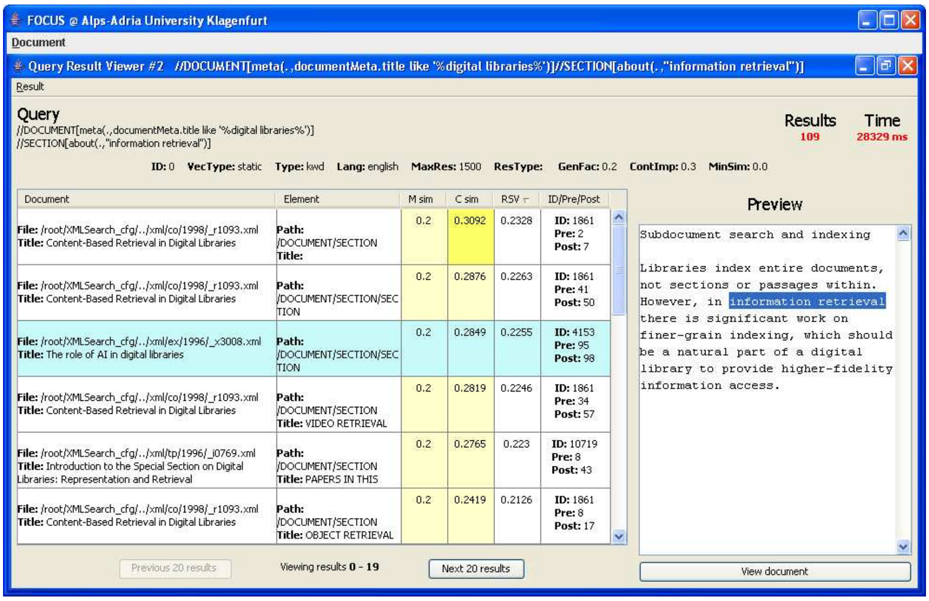
It is obvious that quoted expressions are particulary difficult to process. Consider `"red cars"`. The term `red` is an adjective, it is not included in the index. Furthermore, it is possible that in another context (e.g. "`Red Cross`"), it is (part of) a proper noun and, therefore, exists in the index. In our approach, we treat quoted keywords in two steps: First, we treat them as unquoted, calculating the similarity as given. Then, we apply a string matching strategy on the original text associated with the element to sort the results.

Combinations of MUST/CANNOT and quoted expressions are treated as if all terms within quotes are separately marked as MUST/CANNOT and an initial result set is computed. This result is reduced to those node containing exactly the quoted expression.

Note that the computed result consists of tuples of the form ($docID$, $preorder$, $postorder$, $simMeta$, $simCont$). $docID$ (document ID), $preorder$ and $postorder$ come directly from the database. $simMeta$ and $simCont$ are the calculated meta similarity and content similarity.

5.5 Ranking and result presentation

Ranking is the task by which retrieved elements are decreasingly ordered by their relevance. Therefore, we use a combination of metadata and content similarity to compute a retrieval status value rsv (see Sec. 5.1). The ranking process itself is impacted strongly by the desired granularity. Note that this granularity is either pre-specified or stated explicitly in the user query. For example, if the user specifies the document level (context), say section, the system should return only relevant sections. The similarity can be calculated using two strategies: First, it can be that of the document's (root node) generated recursively from the descendants. Second, it can be the maximum similarity of any of the document nodes. In our experiments, we applied the latter strategy.


FOCUS @ Alps-Adria University Klagenfurt
Document
Query Result Viewer #2 //DOCUMENT[meta(.,documentMeta.title like '%digital libraries%')]//SECTION[about(.,"information retrieval")]
Result
Previous 20 results
Next 20 results
View document

Query
//DOCUMENT[meta(.,documentMeta.title like '%digital libraries%')]
//SECTION[about(.,"information retrieval")]

Results 109 Time 28329 ms

ID: 0 **VecType:** static **Type:** kwd **Lang:** english **MaxRes:** 1500 **ResType:** **GenFac:** 0.2 **ContImp:** 0.3 **MinSim:** 0.0

Document	Element	M sim	C sim	RSV	ID/Pre/Post
File: /root/XMLSearch_cfg/../xml/co/1998/_r1093.xml **Title:** Content-Based Retrieval in Digital Libraries	**Path:** /DOCUMENT/SECTION **Title:**	0.2	0.3092	0.2328	**ID:** 1861 **Pre:** 2 **Post:** 7
File: /root/XMLSearch_cfg/../xml/co/1998/_r1093.xml **Title:** Content-Based Retrieval in Digital Libraries	**Path:** /DOCUMENT/SECTION/SECTION	0.2	0.2876	0.2263	**ID:** 1861 **Pre:** 41 **Post:** 50
File: /root/XMLSearch_cfg/../xml/ex/1996/_x3008.xml **Title:** The role of AI in digital libraries	**Path:** /DOCUMENT/SECTION/SECTION	0.2	0.2849	0.2255	**ID:** 4153 **Pre:** 95 **Post:** 98
File: /root/XMLSearch_cfg/../xml/co/1998/_r1093.xml **Title:** Content-Based Retrieval in Digital Libraries	**Path:** /DOCUMENT/SECTION **Title:** VIDEO RETRIEVAL	0.2	0.2819	0.2246	**ID:** 1861 **Pre:** 34 **Post:** 57
File: /root/XMLSearch_cfg/../xml/tp/1996/_i0769.xml **Title:** Introduction to the Special Section on Digital Libraries: Representation and Retrieval	**Path:** /DOCUMENT/SECTION **Title:** PAPERS IN THIS	0.2	0.2765	0.223	**ID:** 10719 **Pre:** 8 **Post:** 43
File: /root/XMLSearch_cfg/../xml/co/1998/_r1093.xml **Title:** Content-Based Retrieval in Digital Libraries	**Path:** /DOCUMENT/SECTION **Title:** OBJECT RETRIEVAL	0.2	0.2419	0.2126	**ID:** 1861 **Pre:** 8 **Post:** 17

Viewing results **0 - 19**

Preview

```
Subdocument search and indexing

Libraries index entire documents,
not sections or passages within.
However, in information retrieval
there is significant work on
finer-grain indexing, which should
be a natural part of a digital
library to provide higher-fidelity
information access.
```

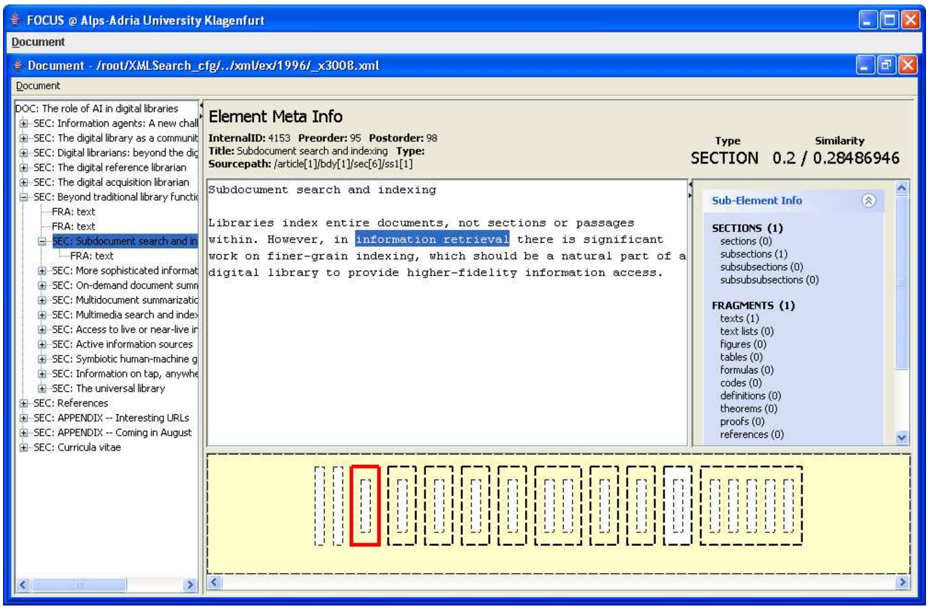
Fig. 7: Result Set

After all desired elements are matched against the user query, the combined similarity values *metaSim* and *contSim* are used for ranking. The results are presented to the user as a sorted list in decreasing order (see Fig. 7). The user is then able to select a particular result, enabling a display of whole document in an explorer-like view (see Fig. 8). The document structure is presented as an expandable tree, where the selected element is expanded and focused. Having similarity values available on the screen, the document can be efficiently browsed. Different colors are used to reflect the degree of similarity of the matched elements.

5.6 Result filtering

In INEX 2004, two kinds of retrieval strategies, thorough and focussed, were defined. *Thorough retrieval* returns all relevant elements of a document. Hence, all ancestors of a relevant element are relevant to a certain degree. This may lead to multiple result elements along the same path (e.g., a section and its contained paragraphs).

Focussed retrieval, on the other hand, aims at returning only the most relevant element along a path. Basically, it relies on two principles [13]: (a) if an element is relevant to a certain degree, so must be its parent; (b) only one node along a path of relevant elements is returned. Overlapping elements in the result set are discarded. This strategy is implemented as post filtering process to re-


FOCUS @ Alps-Adria University Klagenfurt
Document
Document - /root/XMLSearch_cfg/../xml/ex/1996/_x3008.xml
Document

DOC: The role of AI in digital libraries
- SEC: Information agents: A new chall
- SEC: The digital library as a communit
- SEC: Digital librarians: beyond the dig
- SEC: The digital reference librarian
- SEC: The digital acquisition librarian
- SEC: Beyond traditional library functio
  - FRA: text
  - FRA: text
  - SEC: Subdocument search and in
    - FRA: text
  - SEC: More sophisticated informat
  - SEC: On-demand document summ
  - SEC: Multidocument summarizatio
  - SEC: Multimedia search and index
  - SEC: Access to live or near-live in
  - SEC: Active information sources
  - SEC: Symbiotic human-machine g
  - SEC: Information on tap, anywhe
  - SEC: The universal library
- SEC: References
- SEC: APPENDIX -- Interesting URLs
- SEC: APPENDIX -- Coming in August
- SEC: Curricula vitae

Element Meta Info

InternalID: 4153 **Preorder:** 95 **Postorder:** 98
Title: Subdocument search and indexing **Type:**
Sourcepath: /article[1]/bdy[1]/sec[6]/ss1[1]

Type	Similarity
SECTION	0.2 / 0.28486946

```
Subdocument search and indexing

Libraries index entire documents, not sections or passages
within. However, in information retrieval there is significant
work on finer-grain indexing, which should be a natural part of a
digital library to provide higher-fidelity information access.
```

Sub-Element Info

SECTIONS (1)
sections (0)
subsections (1)
subsubsections (0)
subsubsubsections (0)

FRAGMENTS (1)
texts (1)
text lists (0)
figures (0)
tables (0)
formulas (0)
codes (0)
definitions (0)
theorems (0)
proofs (0)
references (0)

Fig. 8: Result Browser

fine the result set. We rely on preorder and postorder to do this efficiently. This strategy reduces the number of returned elements drastically.

5.7 Query Refinement

In most cases a final search result is achieved through iterative refinement of the query. The number of results is reduced step by step by adding new information to the query. To enable such a feature, we allow the user to include a list of preliminary results together with a query. If such a result is set within a query it acts as a strict filter during query computation.

6 Conclusion

The paper described the basic tasks of an XML retrieval system. Details on the methodology are provided. An initial experimental evaluation is already, but only partly, conducted showing promising results. However, a thorough empirical work is still needed along with some additional features of the system.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley, ACM Press, New York, Essex, England (1999)

2. Salton, G., Lesk, M.E.: Computer evaluation of indexing and text processing. Journal of the ACM **15** (1968) 8–36
3. Salton, G.: The SMART Retrieval System - Experiments in Automatic Document Processing. Prentice Hall Inc., Englewood Cliffs, NJ (1971)
4. Grosjohann, K., Fuhr, N., Effing, D., Kriewel, S.: A user interface for XML document retrieval. In: 32. GI-Jahrestagung. Springer (2002)
5. Grosjohann, K., Fuhr, N., Effing, D., Kriewel, S.: Query formulation and result visualization for XML retrieval. In: Proceedings ACM SIGIR 2002 Workshop on XML and Information Retrieval, ACM (2002)
6. Fuhr, N., Grosjohann, K., Kriewel, S. In: A Query Language and User Interface for XML Information Retrieval. Volume 2818 of LNCS. Springer (2003) 59–75
7. Fuhr, N., Grosjohann, K.: XIRQL: A query language for information retrieval in XML documents. [14] 172–180
8. Gövert, N.: Bilingual information retrieval with HyREX and Internet translation services. In: Cross-Language Information Retrieval and Evaluation. Volume 2069 of LNCS. (2001) 237–244
9. Grust, T.: Accelerating XPath location steps. In: Proc. of the 2002 ACM SIGMOD, ACM Press (2002) 109–120
10. Hiemstra, D.: A database approach to content-based xml retrieval. [15] 111–118
11. Florescu, D., Kossmann, D.: A performance evaluation of alternative mapping schemes for storing XML data in a relational database. Technical report (1999)
12. Abolhassani, M., Fuhr, N.: Applying the divergence from randomness approach for content-only search in XML documents. In: 26th European Conf. on Information Retrieval Research (ECIR), Springer Verlag (2004)
13. Kazai, G., Lalmas, M., Rölleke, T.: Focussed structured document retrieval. In: Proceedings of the 9 Retrieval (SPIRE 2002), Springer (2002) 241–247
14. Proc. of the 24th ACM SIGIR. In: Proc. of the 24th ACM SIGIR, ACM Press (2001)
15. INitiative for the Evaluation of XML Retrieval (INEX, Workshop). In: INitiative for the Evaluation of XML Retrieval (INEX, Workshop), ERCIM (2003)

TRIX Experiments at INEX 2005

Paavo Arvola[1], Jaana Kekäläinen[1], and Marko Junkkari[2]

[1] Deparment of Information Studies, Kanslerinrinne 1,
33014 University of Tampere, Finland
{jaana.kekalainen, paavo.arvola}@uta.fi
[2] Department of Computer Sciences, Kanslerinrinne 1,
33014 University of Tampere, Finland
junken@cs.uta.fi

Abstract. This paper presents results of our runs at INEX 2005, where the following tasks were involved: CO.Focussed, CO.FetchBrowse, CO.Thorough and all of the CAS tasks. Our retrieval system utilizes the natural tree structure of XML and is based on structural indices. While creating result lists, two different overlapping models have been applied according to task. The weights of the ancestors of an element have been taken into account in re-weighting in order to get more evidence about relevance. This paper shows also how CAS queries can be processed by utilizing structural indices.

1 Introduction to TRIX

The present study comprises of retrieval experiments conducted within the INEX 2005 framework addressing the following research questions: ranking of elements of 'best size' for CO queries, query expansion, and handling of structural conditions in CAS queries. In INEX 2005 we submitted runs for the following tasks: CO.focussed, CO.thorough, CO.FetchBrowse, and all of the CAS tasks.

Next we introduce the TRIX (Tampere information retrieval and indexing of XML) approach for indexing, weighting and re-weighting. Then, Sections 2 and 3 deal with the CO queries and CAS queries, respectively. Finally conclusions are given in Section 4.

1.1 Structural Indices and Basic Weighting Schema

In TRIX the management of structural aspects is based on the *structural indices* [2,4,5,8]. The idea of structural indices in the context of XML is that the topmost (root) element is indexed by ⟨1⟩ and its children by ⟨1,1⟩, ⟨1,2⟩, ⟨1,3⟩ etc. Further, the children of the element with the index ⟨1,1⟩ are labeled by ⟨1,1,1⟩, ⟨1,1,2⟩, ⟨1,1,3⟩ etc. This kind of indexing enables analyzing of the relationships among elements in a straightforward way. For example, the ancestors of the element labeled by ⟨1,3,4,2⟩ are associated with the indices ⟨1,3,4⟩, ⟨1,3⟩ and ⟨1⟩. In turn, any descendant related to the index ⟨1,3⟩ is labeled by ⟨1,3,ξ⟩ where ξ is a non-empty part of the index. In the present approach the XML documents in the collection are labeled by positive integers 1, 2, 3, etc. From the perspective of indexing this means that the documents are identified by indices ⟨1⟩, ⟨2⟩, ⟨3⟩, etc., respectively. The length of an index ξ is denoted by *len*(ξ). For example *len*(⟨1,2,2,3⟩) is 4. *Cutting operation* $\delta_i(\xi)$ selects the subindex of the index ξ consisting of its *i* first integers. For example if ξ = ⟨*a*,*b*,*c*⟩ then $\delta_2(\xi)$ = ⟨*a*,*b*⟩. In terms of the cutting operation the root index at hand is denoted by $\delta_1(\xi)$ whereas the index of the parent element can be denoted by $\delta_{len(\xi)-1}(\xi)$.

The retrieval system, TRIX, is developed further from the version used in the 2004 ad hoc track [3] and its basic weighting scheme for a key *k* is slightly simplified from the previous year:

$$w(k,\xi) = \frac{kf_\xi}{kf_\xi + v \cdot \left((1\text{-}b) + b \cdot \dfrac{\xi f_c}{\xi f_k} \right)} \cdot \frac{log\left(\dfrac{N}{m}\right)}{log(N)} \qquad \textbf{(1)}$$

where

- kf_ξ is the number of times *k* occurs in the ξ element,
- *N* is the total number of content elements in the collection,
- *m* is the number of content elements containing *k* in the collection,
- ξf_c is the number of all descendant content elements of the ξ element
- ξf_k is the number of descendant content elements of the ξ element containing *k*,
- *v* and *b* are constants for tuning the weighting.

The constants *v* and *b* allow us to affect the ‘length normalization’ (ξf_c / ξf) component and tune the typical element size in the result set. In our runs for INEX 2005 *b* is used for tuning, while v is set to 2. Small values of *b* (0-0.1) yield more large elements, whereas big values (0.8-1) yield more small elements.

The weighting formula above yields weights scaled into the interval [0,1]. The weighting of phrases and the operations for + and - prefixes have the same property. They are introduced in detail in [3]. A *query term* is a key or phrase with a possible prefix + or -. A CO query *q* is a sequence of query terms k_1, …, k_n. In relevance scoring for ranking the weights of the query terms are combined by taking the average of the weights:

$$w(q,\xi)=\frac{\sum_{i=1}^{n} w(k_i,\xi)}{n} \tag{2}$$

After this basic calculation elements' weights can be re-weighted. Next we consider the used re-weighting method, called contextualization.

1.2 Contextualization

In our runs we use a method called contextualization to rank elements in more efficient way in XML retrieval [1, see also 7]. Re-weighting is based on the idea of using the ancestors of an element as a context. In terms of a contextualization schema the context levels can be taken into account in different ways. Here we applied four different contextualization schemata.

1) Root (denotation: $c_{r1.5}(q, \xi)$)
2) Parent (denotation: $c_p(q, \xi)$)
3) Tower (denotation: $c_t(q, \xi)$)
4) Root + Tower (denotation: $c_{rt}(q, \xi)$)

A contextualized weight is calculated using weighted average of the basic weights of target element and its ancestor(s), if exists. Root contextualization means that the contextualized weight of an element is a combination of the weight of an element and its root. In our runs the root is weighted by the value 1.5. This is calculated as follows:

$$c_{r1.5}(q, \xi) = \frac{w(q,\xi)+1.5 * w(q,\delta_1(\xi))}{2.5} \tag{3}$$

Parent contextualization for an element is an average of the weights of the element and its parent.

$$c_p(q, \xi) = \frac{w(q,\xi)+w(q,\delta_{len(\xi)-1}(\xi))}{2} \tag{4}$$

Tower contextualization is an average of the weights of an element and all its ancestors.

$$c_t(q,\,\xi) = \frac{\sum_{i=1}^{len(\xi)} w(q,\delta_i(\xi))}{len(\xi)} \tag{5}$$

So called Root + Tower contextualizaton means the plain tower contextualization with root multiplied by two. This can be seen as a combination of parent and root contextualizations.

$$c_{rt}(q,\,\xi) = \frac{w(q,\delta_1(\xi)) + \sum_{i=1}^{len(\xi)} w(q,\delta_i(\xi))}{len(\xi)+1} \tag{6}$$

In Figure 1 the effects of the present contextualization schemata are illustrated. In it, XML tree with elements assigned initial weights (w) and contextualized weights: Root (wr), Parent (wp) and Root + Tower (wrt) is given. For instance, element with index ⟨1,1,2⟩ has an basic weight of 0.2. Parent contextualization means an average weight of ⟨1,1,2⟩ and ⟨1,1⟩. Root is the average of ⟨1,1,2⟩ and ⟨1⟩ and Root + Tower the weighted average of weights of ⟨1⟩ , ⟨1,1⟩ and ⟨1,1,2⟩, where the weight of ⟨1⟩ has been calculated twice.

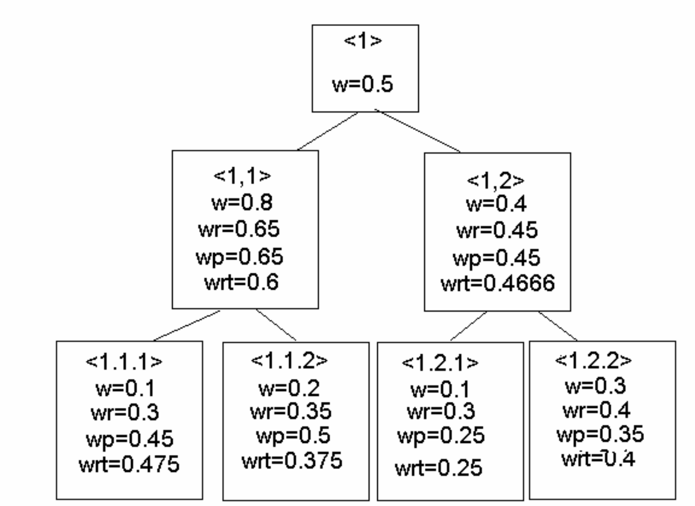


Fig. 1. A tree presentation of an XML document illustrating different contextualization schemata.

In [1] we have discovered that a root element carries the best evidence related to the topics and assessments of INEX 2004. However, contextualizing the root only has an effect on the order of elements in the result list, and it does not change the order of elements within a document. Generally, if we contextualize the weights of elements x and y with the weight of their ancestor z, the order of x and y will not change in the result list. Further, the mutual order of x, y and z will not change if no re-weighting (i.e. contextualization) method is applied to element z. The root element possesses no context in our approach. Hence in the CO.FetchBrowse task, where documents have to be ordered first, the Root contextualization will not have an effect on the rankings of other elements. However, within a document there are still several other context levels, and by utilizing those levels, it is possible to re-rank elements within a document. This finding has been utilized in the CO.FetchBrowse task.

1.3 Overlapping Models

In Figure 2 two overlapping models, which our system supports, are illustrated. First, an element to be returned is marked with a letter P. On the left there is a situation where all overlapping elements are excluded from the result list, even if their weight would be sufficient, but smaller than P. That means the overlapping percentage is 0. On the right side all elements can be accepted, regardless of their structural position in the document.

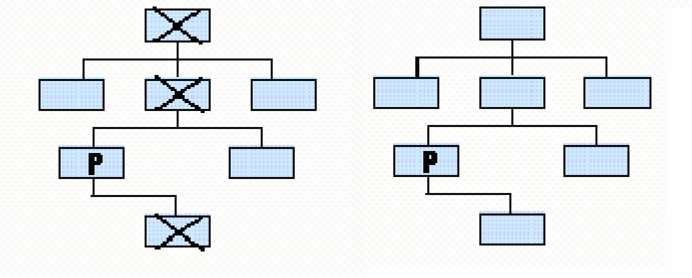


Fig. 2. Two overlapping models

We have used the former model in the CO.Focussed and CO.FetchBrowse tasks and the latter model in the CO.Thorough and all of the CAS tasks.

2 CO Runs

In the CO runs we have used Root+Tower contextualization (Tampere_..._tower), and Root contextualization (Tampere_..._root). In addition we have applied a query expansion method from Robertson [6], taking 5 or 10 expansion words from 7 top documents from the first result set (Corresponding runs: Tampere_exp5_b09_root, Tampere_exp10_b01_root). Figure 3 shows the slight improvement of the expanded run compared with a similar run without any expansion. Topic-specific anlysis will take place in the near future.

Because of the prevention of overlapping elements, promoting large elements may not be wise in the focussed task. That is because if a large element is returned, then every descendant is excluded from the results. However, in thorough task promoting large elements is not that risky. Hence, we used small *b* values for the thorough and large values for the focussed runs. Favoring small elements might have caused another kind of problem, though. In the relevance assessments many of the paragraph sized elements are marked as too small. That leads to a situation, where a whole relevant branch is paralyzed, when a too small leaf element is returned.

In the topic 229 there is a spelling error "latent semantic anlysis", which in our system would lead to a poor score. To minimize the error rate and also to improve recall, we have opened the phrases in all of the queries. For instance, query *"latent semantic anlysis*" would become *"latent semantic anlysis" latent semantic anlysis*. These features and also the effect of the contextualization improve recall and scores in generalized quantization, although the top precision suffers slightly (see figures 3 and 4).

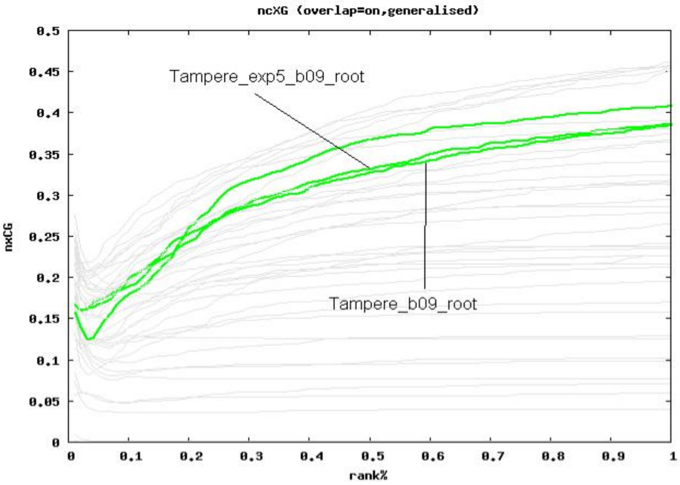


Fig. 3. The nXCG curves of runs in CO.Focussed task with generalized quantization

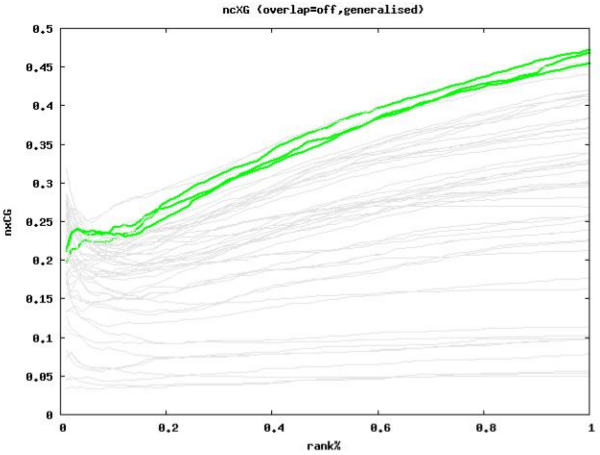


Fig. 4. The nXCG curves of runs in CO.Thorough task with generalized quantization

3 CAS Runs

3.1 Processing CAS Queries

In the CAS queries an element may have constraints concerning itself, its ancestors or descendants. These constraints may be only structural, or structural with content. For instance in query

//A[about(.,x)]//B[about(.//C,y)]

B is the structural constraint of a target element itself. A is a structural constraint of a target element's ancestor, and C target element's descendants. All of these structural constraints have also content constraints, namely *x* or *y*. So, to be selected to a result list, an element must fulfil these constraints. The processing of CAS queries can be divided into four steps:

- First step: Generate a tree according to the target element's content constraint, and weight elements, which fulfil the target element's structural constraint.

- Second step: Discard all the target elements which do not fulfil the structural ancestor and descendant constraints. Due to the nature of hierarchical data, ancestors are always about the same issue as their descendants, i.e. they share the descendants' keys. So the content constraints of descendant elements are taken into account here as well.

- Third step: Generate trees according to each ancestor element's content constraint. Discard elements, where the structural descendant and ancestor content constraint are not fulfilled, i.e. corresponding elements do not exist in the sub tree.

- Fourth step: Collect the indices of elements left in the third step having the ancestor structural constraint, and discard all of the target elements, which do not have such indices among ancestor elements.

To clarify this, processing of a CAS query can be demonstrated with a sufficiently complex example.

The query:

//article[about(.//abs, logic programming)]//bdy//sec[about(.//p, prolog)]

breaks down into following parts:

- an element with structural constraint **sec** is the target element with content constraint *prolog*
- **p** is a structural descendant constraint of the target element with the same content constraint as **sec :** *prolog*
- **article** is a structural ancestor constraint of the target element with a content constraint *logic programming*
- **abs** is a structural descendant constraint of **article** with the same content constraint *logic programming*
- **bdy** is a structural ancestor constraint of the target element without any content constraints

In the first step, shown in Figure 5, we form a tree of elements with non-zero weights according to the query *prolog*. In other words all the elements with zero weights are discarded from an XML tree structure.

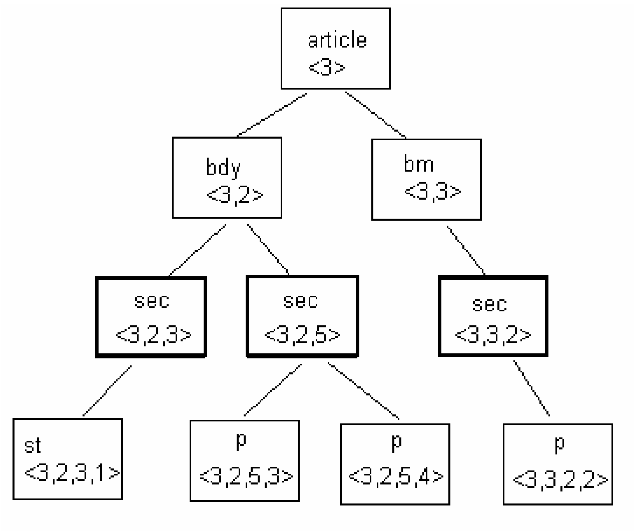


Fig. 5. A tree presentation of a sample XML document having only elements with a weight greater than 0 according to the query *prolog*.

In the second step (Figure 6), we exclude target element ⟨3,3,2⟩, because the structural ancestor constraint **bdy** is not fulfilled. Element ⟨3,2,3⟩ is also to be excluded, because the descendant constraint **p** is not fulfilled.

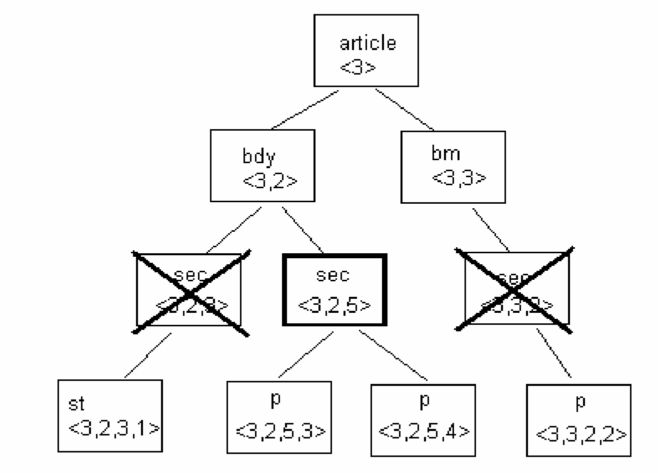


Fig. 6. A tree presentation of a sample XML document having only elements with a weight greater than 0 according to the query *prolog*, where target elements not fulfilling the constraints are excluded.

In the third step we form a tree with non-zero weights according to the query *logic programming*, as seen in Figure 7.

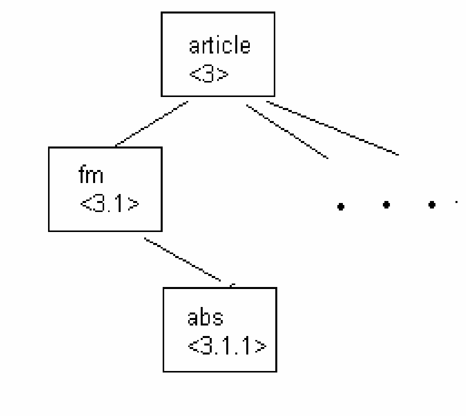


Fig. 7. A tree presentation of a sample XML document having only elements with a weight greater than 0 according to the query logic programming.

In the tree, there is an **abs** element as a descendant of **article**, so both of the structural and content constraints are fulfilled. Hence, we take the index of the article: ⟨3⟩, and see that the index belongs to a descendant of the remaining target element ⟨3,2,5⟩. So, this and only this element is to be returned from this document.

3.2 Taking Vagueness into Account in CAS

In the current evaluations there are four different kinds of interpretations for structural constraints for processing NEXI, in our approach the structural constraints are interpreted strictly. However for SVCAS, VSCAS and VVCAS the query has been modified. Our system handles vague interpretation such that the corresponding element names have been ignored. In NEXI language this can be implemented by replacing the names with a star. Thus we have modified CAS queries as follows:

The initial CAS query (and SSCAS):
//A[about(.,x)]//B[about(.,y)]

SVCAS:
//*[about(.,x)]//B[about(.,y)]

VSCAS:
//A[about(.,x)]//*[about(.,y)]

VVCAS would then logically correspond to:

//*[about(.,x)]//*[about(.,y)]

For simplification we have processed VVCAS like a CO query. In the present example VVCAS corresponds to the query:
//*[about(.,x y)]

3.3 Results of CAS Queries

In the content and structure queries, only elements which fulfil the constraints are accepted to the results. The ranking of the elements has been done according to the target element's textual content. Besides the target element, other content constraints have been taken into account as a full match constraint without any weighting. This full match content constraint within a structural constraint has been interpreted in disjunctive way. It means, that only one occurrence of any of the keys in a sub query is sufficient enough to fulfil the condition. For instance in the query

//A[about(.,x y z)]//B[about(.//C,w)]

for B to be returned, it is sufficient that the A element includes only one of the keys *x, y* or *z*. Naturally the B element should be about *w*, and also have a descendant C about *w*. This approach among others mentioned in the Section 2 leads to fairly good results with the generalized quantization (see Figures 8 and 9). However, if in SSCAS the number of highly relevant elements per topic is low and if there are only a few of topics assessed, then the evidence especially for the strict quantization is narrow.

There was a slight error in our submissions of results. Accidentally we sent runs intended for SVCAS for VSCAS, and vice versa. Because of the near zero overlap, in SVCAS this led to quite a rotten score. Surprisingly, despite the error, VSCAS results proved to be quite satisfactory, as Figure 10 in Appendix shows. Especially, according to the top precision of our runs, the ranking was as high as 3rd and 4th in the generalized quantization and 3rd and 8th in the strict quantization of the XCG metrics.

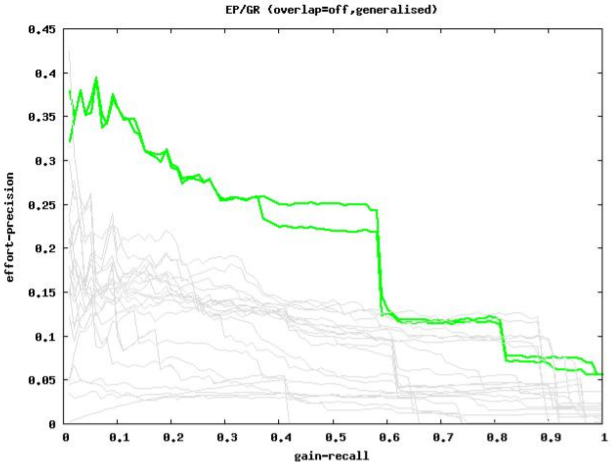


Fig. 8. The ep/gr curves of runs in SSCAS task with generalized quantization

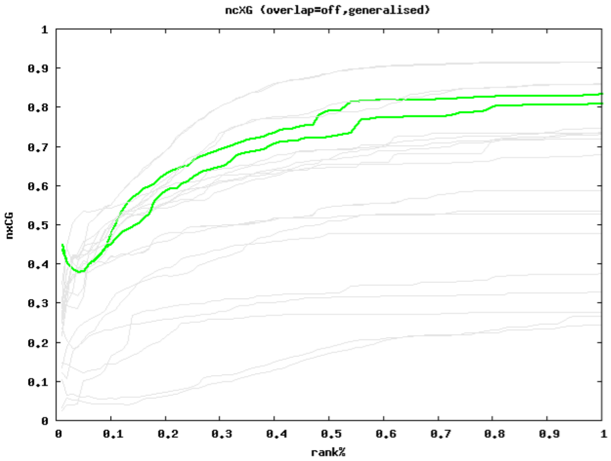


Fig. 9. The nXCG curves of runs in SSCAS task with generalized quantization

4 Conclusions

This paper presents our experiments and results at INEX 2005. The results for the CO task show that Root contextualization is not generally better than Root + Tower, except for the top precision. In general, our approach is in many runs quite recall oriented, and we also do better in the generalized than strict quantization. Therefore, improving top precision in all tasks and quantizations remains as one of our primary goals.

This was the first time we participated in (strict) CAS task. The analyzing power of structural indices enables a straightforward processing of CAS queries. In addition, results in INEX 2005 give a good baseline for future development. By the time this paper has been written, the results of CO.FetchBrowse are considered as preliminary, and our results are not yet included. That is because of a minor error in our result lists. The final results of the CO.FetchBrowse will show how different contextualizations within a document will affect the results.

References

1. Arvola, P., Junkkari, M., and Kekäläinen, J.: Generalized Contextualization Method for XML Information Retrieval, In Proceedings of ACM Fourteenth Conference on Information and Knowledge Management (CIKM'2005), (2005) 20-27.

2. Junkkari, M.: PSE: An object-oriented representation for modeling and managing part-of relationships. Journal of Intelligent Information Systems, 25(2), (2005) 131-157

3. Kekäläinen, J., Junkkari, M., Arvola, P., and Aalto, T.: TRIX 2004: Struggling with the overlap. In Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004. LNCS 3493. Springer, Heidelberg, (2005) 127-139

4. Knuth, D.: Fundamental Algorithms: The Art of Computer Programming. Vol. 1, Addison Wesley, (1968)

5. Niemi, T.: A Seven-Tuple Representation for Hierarchical Data Structures. Information Systems, 8(3), (1983) 151-157

6. Robertson, S.E. and Walker, S.: Okapi/Keenbow at TREC-8, Proc. NIST Special Publication 500-246: The Eighth Text Retrieval Conference Text (TREC), (1999) 151-162.

7. Sigurbjörnsson, B., Kamps J., and de Rijke, M.: An Element-Based Approach to XML Retrieval. In INEX 2003 Workshop Proceedings (2003) 19-26

8. Tatarinov, I., Viglas, S., Beyer, K.S. Shanmugasundaram, J., Shekita, E.J., and Zhang C.: Storing and Querying Ordered XML Using a Relational Database System. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, (2002) 204-215

Appendix

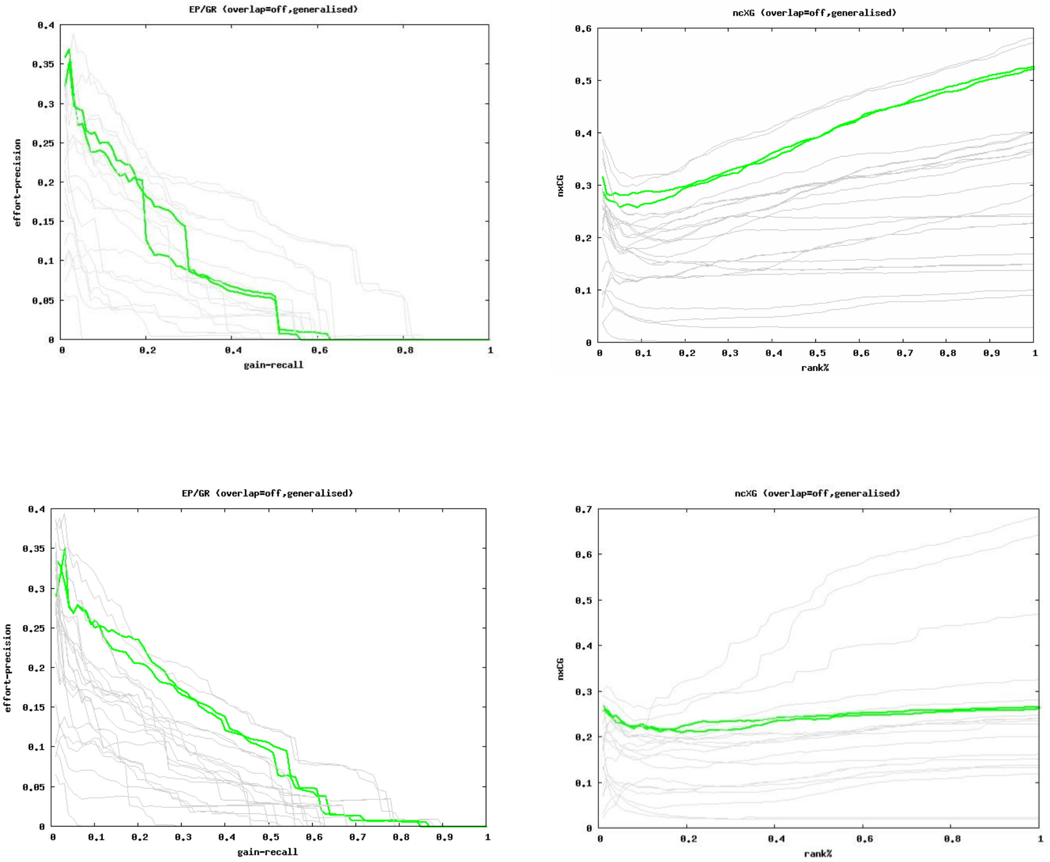


Fig. 10. The EP/GR and nXCG curves of the generalized quantization. First row: VVCAS, Second row: VSCAS.

B^3-SDR: Basic Building Blocks for Structured Document Retrieval.

Roelof van Zwol

Utrecht University, Department of Computer Science, Center for Content and Knowledge Engineering, Utrecht, the Netherlands
roelof@cs.uu.nl

Abstract. Structured document retrieval, or XML element retrieval as it is referred to within INEX, the INitiative for the Evaluation of XML retrieval, allows for the retrieval of XML elements containing highly specific relevant information. INEX provides an evaluation platform where retrieval strategies for structured documents are evaluated. This is the second year that Utrecht is participating in INEX, with a completely revised system called B^3-SDR. The B^3-SDRsystem is a modular system that uses basic building blocks (B^3) to evaluate different strategies for structured document retrieval. The kernel of the system is based on the model introduced by [1]. Their heuristic model was simple, yet effective, and provided several options for extensibility.

This article presents the various extensions that are defined on top of the basic model for the different tasks within the INEX Ad-hoc track. Due to the heuristic nature of the retrieval model, various configurations are possible, depending on the retrieval tasks that is specified. The underlying motivation is discussed for the different tasks, and evaluated.

References

1. Geva, S.: Gpx - gardens point xml information retrieval at inex 2004. In: Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004. Volume 3493 of Lecture Notes in Computer Science., Dagstuhl Castle, Germany, Springer (2005) 211–223

Field-Weighted XML Retrieval Based on BM25

Wei Lu[1], Stephen Robertson[2], Andrew Macfarlane[3]

[1] Center for Studies of Information Resources
School of Information Management
Wuhan University, China
sa713@soi.city.ac.uk
[2] Microsoft Research
Cambridge, U.K.
ser@microsoft.com
[3] Centre for Interactive Systems Research
Department of Information Science
City University London
andym@soi.city.ac.uk

Abstract. This is the first year for the Centre for Interactive Systems Research participation of INEX. Based on a newly developed XML indexing and retrieval system on Okapi, we extend Robertson's field -weighted BM25F for document retrieval to element level retrieval function BM25E. In this paper, we introduce this new function and our experimental method in detail, and then show how we tuned weights for our selected fields by using INEX 2004 topics and assessments. Based on the tuned models we submitted our runs for CO.Thorough, CO.FetchBrowse, the methods we propose show real promise. Existing problems and future work are also discussed.

1. Introduction

Being an important data exchange and information storage standard, XML is now widely used, especially for scientific data repositories, Digital Libraries and on the Web, which has brought about an explosion in the research of information retrieval for XML. Many sophisticated systems [1, 2, 3, 4, 5] and retrieval models [6, 7, 8, 9, 10] for XML documents have been proposed.

XML documents often contain sub-fields (elements), eg. INEX collections from IEEE contain fields such as title, abs, bdy, bm, st etc. Practitioners have found it beneficial to exploit the documents' internal structure to improve retrieval performance [11]. Researchers have looked at various techniques in order to investigate this problem. Wilkinson [12] and Ogilvie et al [13] have proposed and tested different ways to weight and combine the scores obtained on different fields of a document; Kraaij et al [14] propose a flexible algorithm based on language models but have not implemented it;

and Myaeng et al [15] combine terms found in different document representations using Bayesian inference networks. Robertson et al [11] give a more detailed review of this area in their paper.

In practice, many systems use a linear combination of the scores obtained from scoring every field due to the complexity of the ranking algorithms deployed. Robertson et al [11] discuss the dangers of linear combination in detail and propose an alternative solution, the linear combination of term frequencies based on BM25 (BM25F will be used in the rest of the paper instead of " field-weighted models based on BM25"), to extend standard ranking functions to multiple weighted fields. Their experiment based on two existing collection Reuters vol. I collection and the 2002 TREC Web-Track crawl of the .gov for document level retrieval shows that the method was beneficial. Some related work using Okapi, BM25 or field combination in INEX 2004 are documented in [16, 17, 18, 19, 20].

In this paper, we extend this method further to element level XML retrieval based on INEX 05 collections. In section 2, we discuss in detail the field-weighted models. Section 3 further illustrates the experiment of this method on INEX 05 and Evaluation results are reported in section 4. A conclusion and further work to be undertaken are described at the end.

2. BM25F model

In this section we describe BM25F model in detail. We first introduce the models for document level weighting in section 2.1. And then we further discuss the implementation of the model to XML element level retrieval.

2.1 BM25F for document level weighting

BM25F is the field-weighted version of BM25. It is derived from Robertson et al [11] for document level retrieval. For ad-hoc retrieval, and ignoring any repetition of terms in the query, BM25 can be simplified to [11]:

$$w_j(\overline{d},C) = \frac{(k_1+1)tf_j}{k_1((1-b)+b\dfrac{dl}{avdl})+tf_j}\log\frac{N-df_j+0.5}{df_j+0.5} \qquad (1)$$

where C denotes the document collection, tf_j is the term frequency of the *j*th term in $\overline{d}$, df_j is the document frequency of term *j*, *dl* is the document length, *avdl* is the average document length across the collection, and $k1$ and b are tuning parameters.
Then the document score is obtained by term weights of terms matching the query q:

$$W(\overline{d},q,c) = \sum_j w_j(\overline{d},C)\cdot q_j \qquad (2)$$

Being a linear weighted combination of term frequency of in these fields, function BM25F is shown as follows:

$$wf_j(\overline{d},C) = \frac{(k'_1+1)tf'_j}{k'_1((1-b)+b\frac{dl'}{avdl'}+tf'_j}\log\frac{N-df_j+0.5}{df_j-0.5} \qquad (3)$$

where tf'_j denotes the weighted term frequency of the *j*th term in $\overline{d}$, dl' is the weighted document length, $avdl'$ is the weighted average document length across the collection. k'_1 is the weighted free parameter.

Suppose we have *nF* fields *f = 1, . . . , nF*. In a given document *d*, term *t* has frequency $tf_{d,t,f}$ in field *f*. There are various ways of defining the length of fields or documents, but the simplest way is to use the number of indexed terms (tokens). This means that the length of the field in this document is

$$dl_f = \sum_{t\in v} tf_{d,t,f}$$

where V is the vocabulary, i.e. all indexed terms.

With no field weighting, the term frequency of t in the whole document is

$$tf_{d,t} = \sum_f tf_{d,t,f}$$

and the document length is

$$dl = \sum_f dl_f = \sum_f \sum_t tf_{d,t,f} = \sum_t tf_{d,t}$$

Average document length is

$$avdl = \frac{1}{N}\sum dl$$

With field weights W_f, these are modified as follows:

$$tf'_{d,t} = \sum_f w_f tf_{d,t,f}$$

$$dl^{'} = \sum_{f} w_f dl_f = \sum_{f}\sum_{t} w_f tf_{d,t,f} = \sum_{t} tf^{'}_{d,t}$$

$$avdl^{'} = \frac{1}{N}\sum dl^{'}$$

and

$$k^{'}_1 = k_1 \frac{atf_{weighted}}{atf_{unweighted}} = k_1 \frac{avdl^{'}}{avdl}$$

where *atf* is the average term frequency.

Function (3) is used for document weighting. However XML retrieval requires not only document level but also element level retrieval. This means an algorithm for element weighting is required. In section 2.2, we further discuss the field-weighted weighting function for element level retrieval (BM25E) derived from function (3).

2.2 Proposed model BM25E for element weighting

From function (3), we can see that linear combination of weighted field frequencies is used instead of original term frequency in specified document. We hypothesize that this method could also be applied to element retrieval. Our basic view is that an element is to be treated like a document, except that it may inherit information from other elements in the document. Thus each element has (in addition to its own text, which is treated as one field) extra fields consisting of text inherited from other elements. The details of our idea are as follows:

Suppose we have *nE* elements *e = 1, . . . , nE* in given collection C. Term *t* has frequency $tf_{d,t,e}$ in element *e*. *el* is the element length and *avel* is the average element length. Then we simply extend BM25 to element retrieval as follows:

$$w_j(e,\overline{d},C) = \frac{(k_1+1)tf_{e,j}}{k_1((1-b)+b\frac{el}{avel})+tf_{e,j}} \log\frac{N-df_j+0.5}{df_j+0.5} \qquad (4)$$

Accordingly, Function BM25E would be,

$$wf_j(e,\overline{d},C) = \frac{(k^{'}_1+1)tf^{'}_{e,j}}{k^{'}_1((1-b)+b\frac{el^{'}}{avel^{'}})+tf^{'}_{e,j}} \log\frac{N-df_j+0.5}{df_j+0.5} \qquad (5)$$

where $tf'_{e,j}$ denotes the weighted term frequency of *j*th term *t* in *e*, el' is the weighted element length, $avel'$ is the weighted average element length across the collection. k'_1 is the weighted free parameter. Similar to those parameters in section 2.1, given a field weights W_f to elements which contributes to a given elements' Weig ht,

$$tf'_{f,t} = \sum_{f \in e} w_f tf_{f,t}$$

$$el' = \sum_{f \in e} w_f el = \sum_{f \in e} \sum_{t} w_f tf_{f,t} = \sum_{f,t} tf'_{f,t}$$

$$avel' = \frac{1}{M} \sum el'$$

and

$$k'_1 = k_1 \frac{atf_{weighted}}{atf_{unweighted}} = k_1 \frac{avel'}{avel}$$

where M is the total number of element in collection C.

(5) implies that given an element e in collection C, if it exists some fields(element) f contributing to the weight of the element, then a linear combination of field-weighted term frequency of field are applied based on BM25F. Theoretically, *f* could be any element in collection C. In fact, if all elements in a document d contribute to a given element in this document, then we come back to BM25F (3). And if all W_f equal 1, then we further come back to BM25 (1).

What we need to say is that this statement does not in any way define the implementation, but merely the principle of how elements are to be treated. Detail implementation of our experiment is further discussed in section 3.

3 Experiment of BM25E on INEX 2005

In this section, INEX collection and its structure will be introduced. We will then describe the assumptions we used for our experiments. Finally, our experiment environment and procedures are introduced.

3.1 Data sets

There are 2 data sets have been used for our experiment: INEX 1.4 and INEX 1.7. Both of these two collections are from IEEE Computer Society publications.

Inex 1.4: This data set is INEX collection for 2004 which contains 12107 articles of IEEE Computer Society publications from 1995 to 2002.

Inex 1.6: This data set is INEX collection for 2005 which contains 16819 articles of IEEE Computer Society publications from 1995 to 2004.

More details of these collections can be found in table 1.

Table 1: figures of INEX collections

Data sets	**INEX 1.4**	**INEX 1.6**
Size of Data(MB)	494	705
# of elements	8,239,873	11,411,135
# of attributes	2,204,688	4,669,699
# of Articles	12,107	16,819
Avg. Path Level	8	8

3.2 Data structures

As stated in section 1, being academic collections, most of the articles in it contain elements tags which represent articles' title, abstract, body text, section, section title, paragraph, bibliography and appendix etc. These tags in INEX collection are shown in Table 2:

Table 2: INEX important tags and its meaning

Content Name	**Tags**
article title	atl
article abstract	abs
body text	bdy
section	sec, ss1, ss2, ss3
section title	st
paragraph	ilrj, ip1, ip2, ip3, ip4, ip5, item-none, p, p1, p2, p3
bibliography	bib
appendix	bm

As it's discussed in [11], $W_{f,}$ needs to be tuned for each selected field which contributes to the document's weight in BM25F. The same method should also be used for BM25E. Although in theory, every context element would contribute to given element e, in practice, there are more than about ten-million elements in each INEX collections and it is very difficult to tune every element's W_f. The problem then lies in what elements should be chosen for optimisation.

Robertson et al [11] chose title as the tuned field. In this experiment, consider the data structures of INEX, we choose **atl, abs** and **st** as the tuned elements. We believe that title and abstract in some extent reflect the content of an article, and section title in some extent tells us the section and its sub-elements'content. We believe these el e-ments could contribute to the weight of relevant elements. This issue will be discussed in more detail in section 3.3.

3.3 Some assumptions for BM25E on INEX 2005

Due to the costs of implementation and some other factors such as time limitations, we declare our assumptions for the experiments on the elements which should be inherited for other retrievable ones and the ways to compute $avel'$ and k'_1. They are as follows:

Assumption 1: elements in one document do not have effect on elements in other documents. Elements except **atl, abs** and **st** also don't have effect on other elements which are not their ancestors in the same document.

Assumption 2: Elements **atl** and **abs** contributes to the weight of elements **bdy**, **bm** and their child elements. Elements **st** contributes to the weight of the section it belongs to, and also of the sections' child elements and article element. All **st** elements have the same W_f without considering the level they belong to.

Assumption 3: Due to the complexity to compute parameters $avel'$ and k'_1, we believe the values of the article level can be used instead of them for all elements.

Assumption 1 is simple and easy to understand. In Assumption 2, the question may lie in that what role element **st** plays in the relevant sections' other parent elements except article element. And the question in Assumption 3 is that whether the simple replacement of the parameters would affect much of the result. These issues will be tackled in further research.

3.4 Experiment environment and procedures

This is the first year that the CISR has taken part in INEX. We largely conduct our work on Okapi in a Linux environment (using Red Hat 9). Being a traditional retrieval experiment system, Okapi undertake all the processing which was required by INEX experimentation. We have therefore done significant development work for both XML indexing and element level XML retrieval in order to participating in INEX.

Our experimental procedure is as follows: firstly, we tune W_f for selected elements **atl**, **abs** and **st**; secondly, we use Okapi's Basic Search System (BSS) to get a doc ument result set; and finally we use a newly designed XML element weighting and displaying interface to get our final submissions required by INEX, among which, selected W_f parameters are used to get optimized runs. We should also state that only article, **abs**, **bdy**, **bm** and **section** and **paragraph** elements are considered as potential relevant elements for our final runs in our experiment. This may lose some relevant elements, but some small irrelevant elements are filtered at the same time. In the next section, we report our evaluation result for INEX 05.

4. Evaluation

In order to examine the new data structures and algorithms build for our INEX experiments, we used INEX 04 ad-hoc topics and assessment to tune W_f for **atl**, **abs** and **st** on document level by using the average precision score, (we did not evaluate using the INEX methodology at the element level). Our method shows that tuning W_f for these selected elements contributes to an improvement in retrieval performance on the INEX 04 collection. The tuning values for W_f are all integers. We first tuned W_f {**atl**, **abs st}** from {1, 1, 1} to {10, 10, 10} using increments of 1. Result shows that the values of{10, 3, 10} for W_f get the highest average precision score. The best tuning results were obtained when the tuning values for **atl** and **st** are both 10 and tuning values for **abs** are all between 3 to 6, we therefore investigated the tuning scope for **atl** and **st**. We then tried to tune W_f {**atl**, **abs st}** from {1, 1, 1} to {50, 10, 50} in increments of 1. The results shows that a higher value for **atl** yielded better results, the best scope for **st** is from 12 to 25, while the best scope for **abs** was about the same for the first set of tuning experiments conducted. We conducted some further tuning experiments with a larger scope for **atl** and the ranges for **abs** and **st** set to between 1~10 and 10~30 respectively. In these experiments we tuned **atl** from 1 to 300 using increments of 10 and then used increments of 50 for **atl**, to a maximum value of 3000.. We believed that there was no point in investigating larger values. The best average precision score was recorded when the tuned value for **atl** is around 2400. Finally, we tuned **atl** from 2100 to 2700 in increments of 1 in order to obtain the best optimized results. Our experiment shows when using the values of 2356, 4 and 22 for W_f in elements **atl, abs** and **st** respectively we obtained the highest performance for article level retrieval on INEX 04 data. We are a little surprised that the best tuned value for **atl** is so high. The implication is that the selected elements, particularly **atl** and **st** contributed much to the document level XML retrieval in the INEX collection. See table 3 for some of our tuned result for INEX 04.

Table 3: tuned results for INEX 04 on document level

W_f **{atl, abs, st}**	**Sum of** **(Avg precision for co all topics)**
2356, 4, 22	0.143698
2416, 5, 22	0.143678
2668, 5, 25	0.143435
10, 4, 9	0.129819
1, 1, 1	0.124023

Due to the time and resource limitations, we only submited runs for CO.Thorough and CO.FetchBrowse. Based on these tuning experiments and considering the difference between document level retrieval and element level retrieval, and also being concerned that tuned W_f values for **atl** and **st** would be to high, we choose 3 sets of tuning constants of values for W_f {atl, abs, st}, namely {2356, 4, 22}, {1000, 4, 22} and {15, 4, 8} , for submitting CO.Thorough runs; and chose another 3 sets of tuning constants of values for W_f {atl, abs, st}, namely {1000, 4, 22}, {300, 4, 18} and {98, 4, 13}, for submitting CO.FetchBrowse runs.

Though we tuned W_f in document level, we are still pleased to see that our official runs for CO.Thorough rank at the top of the total 39 official runs, especially for "Metric: nxCG(25), Quantization: strict, Overlap=off" , our 3 runs ranks 1st, 2nd and 22nd respectively; for "Metric: nxCG(50), Quantization: strict, Overlap=off" , our 3 runs ranks 1st, 2nd and 10th respectively; and for "Metric: ep-gr, Quantization: strict,Overlap=off" , our 3 runs ranks 1st, 5th and 13th respectively. See Fig. 1, Fig. 2 and Fig. 3 [21] for more information. We also tried to use metric nxCG to compare our 3 official runs for CO.Thorough with the non field-weighted runs whose W_f{ atl, abs, st } is {0, 0, 0}. Result shows that non field-weighted one ranks at the last while the former two runs rank at the top.

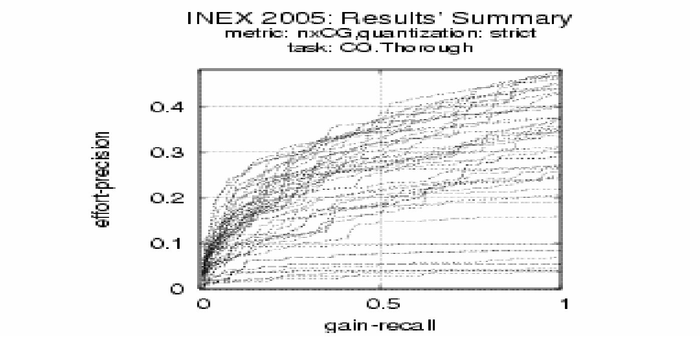


Fig. 1 Metric nxCG(25), Quantization: strict, Overlap=off

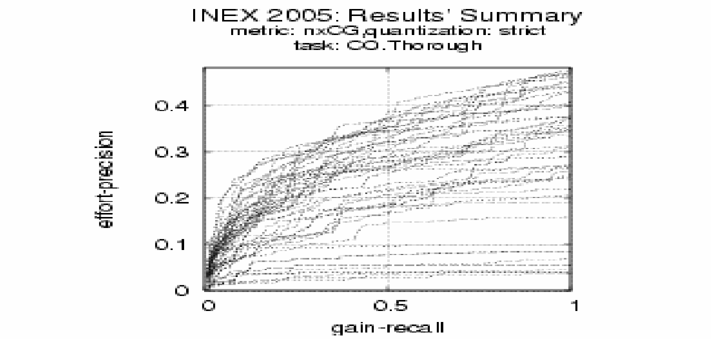


Fig. 2 Metric nxCG(50), Quantization: strict, Overlap=off

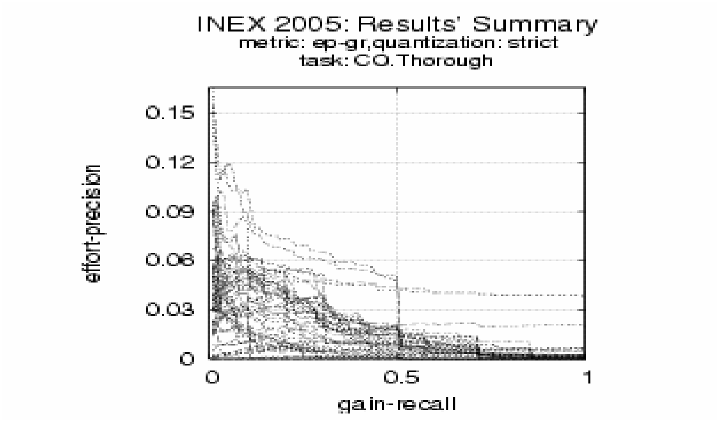


Fig. 3 Metric ep-gr, Quantization: strict,Overlap=off

The experiment shows that the first two sets of tuning constants, W_f {1000, 4, 22} and W_f {2356, 4, 22}, ranks better than the third groups W_f (15, 4, 8). The evidence is that **atl** and **st** does contribute to retrieval performance and it also implies that combining field-weighted term frequencies of selected elements is a beneficial method. Tuning constant set W_f {1000, 4, 22} rank first for Metric "nxCG(25 and 50), Quantization: strict, Overlap=off" also suggests that it may be better if W_f is tuned on element level. This behaviour may also be caused by the difference of the topics and data sets between INEX 2004 and INEX 2005 etc. It is worth doing a further set of tuning experiments on the INEX 2005 topics and data sets.

Results also show that our method performs better for models which consider only fully specific and highly exhaustive components than those models which considering varying levels of relevant components. The reason may be because the selection of elements we chose to submit for our experiments. We intend to investigate this issue further.

5 Conclusion

We extend document level field-weighted retrieval function BM25F to element level retrieval function BM25E. We have applied this method to INEX 2005 CO XML retrieval and results show that our method is beneficial.

However there are still some limitations in our element level retrieval function. Firstly, values for $avel'$ and k'_1 are used at the article level, not element level. The creation of a practical algorithm to generate values for tuning parameters at the element level is a challenging task. Secondly, parameter tuning is undertaken at document level by using average precision method, not on element level by using INEX official metrics. It should be noted that the element **st** has the same weight at different levels, and further experiments need to be undertaken to investigate this problem. Thirdly, we only submit runs for CO.Thorough and CO.FetchBrowse tasks, so more tasks need to be done to test our method. And also our system for XML element retrieval needs to be upgraded. We will investigate these problems in further research.

Acknowledgements
Thanks to Chinese Scholarship Council (CSC) for funding the visitor of the first author to City Univesity, London in order to conduct this research.

References

[1] A. Deutsch, M. Fernandez and D. Suciu. Storing semistructured data with STORED. In Proc. SIGMOD, 1999.

[2] J. Harding, Q. Li, B. Moon. XISS/R: XML Indexing and Storage System Using RDBMS. In Proceedings of the 29th VLDB Conference, 2003

[3] Software AG. Tamino XML database. http://www.softwareag.com/tamino/.

[4] XYZFind. XML Database. http://www.xyzfind.com.

[5] HYREX. http://ls6-www.cs.uni-dortmund.de/ir/projects/hyrex/.

[6] N. Fuhr and K. Groß johann. XIRQL: A Query Language for Information Retrieval in XML Documents. In Research and Development in Information Retrieval, 2001.

[7] J. E. Wolff, H. Florke, and A. B. Cremers. Searching and Browsing Collections of Structural Information. In Proc. IEEE Forum on Research and Technology Advances in Digital Libraries, 2000.

[8] T. Schlieder and H. Meuss. Querying and Ranking XML Documents. Special Topic Issue Journal American Society for Informations Systems on XML and Information Retrieval, 2002.

[9] T. Schlieder. Similarity Search in XML Data using Cost-Based Query Transformations. In Proc. 4th Intern. Workshop on the Web and Databases, 2001.

[10] A. Theobald and G. Weikum. The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking. In Proc. 8th Internation Conf. on Extending Database Technology, 2002.

[11] S. Robertson, H. Zaragoza, M. Taylor. Simple BM25 Extension to Multiple Weighted Fields. CIKM04, 2004.

[12] R. Wilkinson. Effective retrieval of structured documents. In Research and Development in Information Retrieval, 1994.

[13] P. Ogilvie and J. Callan. Combining document representations for known item search. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003), 2003.

[14] W. Kraaij, T. Westerveld, D. Hiemstra. The importance of prior probabilities for entry page search. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002.

[15] S.Myaeng, D. Jang, M. Kim, Z. Zhoo. A flexible model for retrieval of SGML documents. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,1998.

[16] L. A. Clarke, L. Tilker. MultiText Experiments for INEX 2004. In INEX 2004 Proceedings, 2004.

[17] J. Vittaut, B. Piwowarski, Patrick Gallinari. An algebra for Structured Queries in Bayesian Networks. In INEX 2004 Workshop Proceedings, 2004.

[18] J. Kekäläinen, M. Junkkari, P. Arvola. TRIX 2004 – struggling with the overlap. In INEX 2004 Workshop Proceedings, 2004.

[19] R. Larson. Cheshire II at INEX '04: Fusion and Feedback for the Adhoc and Heterogeneous Tracks. In INEX 2004 Workshop Proceedings, 2004.

[20] P. Ogilvie, J. Callan. Hierarchical Language Models for XML Component Retrieval. In INEX 2004 Workshop Proceedings, 2004.

[21] Evaluation results of CO.Thorough. http://inex.is.informatik.uni-duisburg.de/2005/internal/results/CO.Thorough.html.

XML retrieval based on direct contribution of query components

Gilles Hubert

IRIT/SIG-EVI, 118 route de Narbonne, 31062 Toulouse cedex 9
hubert@irit.fr

Abstract. This paper describes the retrieval approach proposed by the SIG/EVI group of the IRIT research centre at INEX'2005. This XML approach is based on direct contribution of the components constituting an information need. This paper focuses on the method evolutions since previous participation to INEX. It describes the experiments done for each subtasks and some corresponding results.

1 Introduction

Due to the growing use of XML (eXtensible Markup Language) to describe documents, a growing number of systems intend to provide solutions to retrieve relevant components among XML documents. These systems are mostly evolutions of either database systems [3] or Information Retrieval (IR) systems. Among IR-based systems two main categories of proposals can be distinguished: systems based on a probabilistic model [7][12] and systems based on the vector space model [2][5]. XML retrieval needs to take into account both content and structural aspects.
In this context of various proposals, a framework such as INEX is useful. On one hand, it offers testbeds and evaluation methods that allow comparing different systems according to common criteria. On the other hand, it allows participants to try to estimate a global efficiency of their system and to determine the contexts adapted to their system.
Among the systems that participated to INEX previous year and that obtained globally good results there are approaches based on the vector space model [8] or close principles [4][6], probabilistic methods [1][13][10][11] and database systems [9]. [8] presents an approach based on the vector space model using multiple indexes, using a document ranking method with document pivot normalization and including a possible automatic query refinement. [4] proposes an approach using inverted lists for terms stored in a database and based on different scoring formulas for leaf elements and branch elements. Our method [6] is based on direct contribution of query components. The main principles of the method are recalled in this paper. This method obtained better results for CAS (Content and Structure) topics. [1] experimented a method based on the Okapi BM25 measure only on the CO (Content Only) topics. [11] uses a multinomial language model with smoothing and associated to documents indexes at different levels (article, element). [10] proposes a hierarchical language

model to represent XML documents as trees and where a model is estimated for each XML component using linear interpolation of the component content, its children's models and its parent model. The approach proposed in [13] represents hierarchies of documents as bayesian networks and computes recursively scores from network root to leaves. [9] describes an extended version of the TIJAH system that follows a three-level database architecture and that has been extended to handle phrase modelling and to support structural relevance feedback.

In this paper, we present an IR method using principles close to approaches based on the vector space model. However, this approach is based on direct contribution of each component of the query and particularly on the presence of each term constituting the query. The paper focuses on the method evolutions done since the previous participation to INEX last year.

In the remainder of this paper a short presentation of the main ideas on which relies the retrieval method is done in Section 2. Section 3 presents how contributions of query components are mapped into scoring principles. Section 4 details the submitted runs and the obtained results. Finally, an analysis of the experiments and an introduction of future works that ensue from it are given in Section 5.

2 Participation objectives

Participating to INEX this year has multiple objectives:

- a first interest was to evaluate the benefit of evolutions brought to the method since last participation. Evolutions intervene in the definition of the function computing the score of an XML element and the score propagation principle through the hierarchical structure of a document,
- in addition, different new subtasks corresponding to different retrieval strategies that could interest a user have been defined in INEX 2005. The experiments carried out in this context can help us to determine the strategies for which our method seems to be a possible response,
- finally, it was interesting to estimate the influence of changes introduced in the INEX 2005 framework regarding metrics and the assessment process.

3 Method principles

The IR method described in this paper is based on principles close to approaches based on the vector space model. Document and query representations are comparable to vectors. However, the correspondence between documents and query is not estimated using a "classical" similarity measure. The method presented is based on direct contribution of each query term appearing in an XML element. The contribution can be modulated according to other components of the query such as structural constraints. A principle of score propagation completes the method with regard to the hierarchical structure of XML documents.

3.1 Representation of INEX elements

From the document point of view, documents are represented as sets of n-tuples (xpath, term, occ) where xpath is the location of the node containing the term from the root of an XML document and occ is the number of occurrences of the term is the textual content of the node. For each XML component, concepts are extracted automatically. Concept extraction involves notably stop word removal and optionally other processes such as stemming using for example the Porter's algorithm. For INEX 2005 experiments all XML tags have been taken into account.
From the topic point of view, according to the INEX 2005 requirements, we used only the title part for CO topics and castitle part for CO+S and CAS topics. However our method can use the other parts constituting CO and CAS topics. For both topic types, stop words are removed and optionally terms can be stemmed. Topics are represented as pairs (target contraint, set of content indications). A content indication is a triplet (term, preference, location constraint). Target constraints and location constraints can be restrictive xpaths for CAS and CO+S topics or generic paths (i.e. matching all elements) notably for CO topics.

3.2 Scoring function

The scoring function is defined as a combination of three values. The scoring function can be globally defined as follows:

$$Score(T,E) = \left(\sum_{\forall t \in T} f(t,E) \cdot g(t,T) \right) \cdot p(T,E)$$

where

T is the topic

t is a term representing the topic T

E is an XML element

$f(t,E)$ This factor measures the importance of the term t in the XML element E.

$g(t,E)$ This factor measures the importance of the term t in the topic representation T.

$p(T,E)$ This factor measures the global presence of the topic T in the XML element E.

On one hand, the function is defined as an addition of contributions of the concepts constituting a query. This principle allows giving relevance to elements dealing about either only one concept or several concepts. The addition tends to promote elements containing several concepts. However, depending on the different chosen functions

an element dealing strongly about one concept can be evaluated higher than an element dealing lightly about many concepts.
On the other hand, the function estimates globally the relevancy of an element according to a query.

The function f that measures the importance of a term in an XML element is based on the number of occurrences of the term in the element or on the relative presence of the term regarding all the occurrences of query terms appearing in the element. This function can be defined as follows:

$$f(t,E) = \frac{Occ(t,E)}{Occ(T,E)^{\alpha}}$$

where

t is a term representing the topic T

E is an XML element

$\alpha \in (0,1)$

$Occ(t,E)$ Number of occurrences of the term t in the element E.

$Occ(T,E)$ Total number of occurrences of all the query terms in the element E

The function g that measures the importance of a term in a topic representation is based on the frequency of the term in the topic. The frequency can be moderated by the number of XML elements containing the term. The function can also use the rank of the term according to the number of elements containing this term and regarding the numbers of elements containing the other query terms.
This function is defined as follows:

$$g(t,T) = \frac{Occ(t,T)}{Size(T)} \cdot \frac{IndRnk(t)^{2}}{NbElts(t)}$$

where

$Occ(t,T)$ Number of occurrences of the term t in the element E.

Size(T) Size of the topic T i.e. total of occurrences of all the terms representing T.

NbElts(t) Number of elements containing the term t

IndRnk(t) Rank of the term t according to the number of elements containing each term of the topic.

This function increases the contributions of terms appearing in few XML elements through the factor NbElts(t) and IndRnk(t).

The function p that measures the global presence of a topic in an XML element is based on the number of terms describing the topic and that appear in the XML element.
This function is defined as follows:

$$p(T,E) = \varphi^{(\frac{NbT(T,E)}{NbT(T)})}$$

where

T is the topic

E is an XML element

φ is a real, φ ≥ 0.0

NbT(T,E) Number of terms describing the topic T and that appear in the XML element E.

NbT(T) Number of terms describing the topic T.

When φ is set to 1.0 the function p has no effect on the final score. The value of φ determines the influence of the function g on the final score. The influence increases with the value of φ. Using a function power intends to clearly distinguish the elements containing a lot of terms describing the topic and the elements containing few terms of the topic.

Additional notions complete the scoring function: the notion of coverage and prefix coefficients. The coverage is a threshold corresponding to the percentage minimum of topic terms that have to appear in an element to select it. It aims at ensure that only documents in which the topic is represented enough will be selected for this topic. Prefix coefficients intend to increase or reduce term contributions according to sign '+' and '-' associated to terms in the query. These notions and their integration in the scoring function are detailed in [6].

3.3 Score propagation according to XML structure

The hierarchical structure of XML has to be taken into account. The hypothesis on which is based our method is that an element containing a component selected as relevant is also relevant. Our approach takes into account this hypothesis propagating the score of an element to the elements it composes. The score propagated to the composed elements is decreased applying a reducing factor. The propagation principle is the following:

$$\forall E_a \textit{ ancestor of } E \quad \textit{and} \quad d(E_a,E)\cdot\alpha<1$$

$$Score(E_a,T) = Score(E_a,T) + (1-2\cdot\lambda\cdot\left(\frac{d(E_a,E)}{d(E_a,E)+d(E_R,E)}\right)^2)\cdot Score(E,T)$$

where

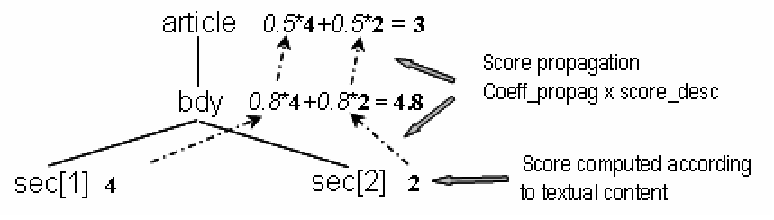
λ is a constant coefficient real ≥0.0 and E is an XML element

$d(E_a,E)$ is the distance between E_a and E in the xpath associated to E (e.g. in the xpath /article/bdy/s/ss1/p the distance between p and bdy is equal to 3 i.e. d(bdy,p)=3)

$d(E_R,E)$ is the distance between the root E_R and E in the xpath associated to E

This process tends to consider a composed element less relevant than the element it is composed of. However, an element composed of several relevant elements can obtain a score greater than one of its components. The coefficient λ allows to vary the score contribution of an element in its ancestors. When λ=0.0 the score of an element is totally propagated towards its ancestors.

The following figure illustrates the score propagation principle:



3.4 Structural constraints

Two types of structural constraints can be used to define to INEX CAS topics:

- constraints on content that is to say xpath of elements which are expected to contain searched concepts (e.g. about(.//p,'+XML +"information retrieval"),
- constraints on the granularity of elements expected as result (e.g //article[….]).

Structural constraints on content are taken into account adding a coefficient varying the contribution given by a query term. If the XML element does not verify the constraint associated to the term, the contribution given by the term is reduced. The coefficient intervenes in the function f that measures the importance of a term in an XML element (cf section 3.2) as follows:

$$f(t,E) = \beta \cdot \frac{Occ(t,E)}{Occ(T,E)^{\alpha}}$$

where

if E does not verify the structural constraint defined on t then 0.0<β<1.0

else β=1.0

This principle constitutes a first solution. However, only XML elements with textual content that verify the constraints on content are affected and by propagation the elements containing them. This could be a limitation to fully respond to CAS tasks with strict verification of content constraints notably the task SSCAS. This principle should be extended to take into account XML elements without textual content and that verify the constraints on content but composed of components containing query terms and not verifying the associated constraints.

In addition, structural constraints on the granularity of elements expected as result are handled adding a coefficient varying the global score computed for an XML element according to content. If the XML element does not verify the constraint on result granularity associated to the query, the score computed is reduced. The coefficient intervenes in the scoring function as follows:

$$Score(T,E) = \gamma \cdot \left(\sum_{\forall t \in T} f(t,E) \cdot g(t,T) \right) \cdot p(T,E)$$

where

if E does not verify the structural constraint defined on T then 0.0≤γ<1.0

else γ=1.0

This solution allows attaching variable importance to structural constraints on result granularity. When γ=0.0 the structural constraints on result are strictly taken into account.

4 Experiments

At least one run based on our XML retrieval method was submitted to INEX 2005 for each subtask. For the subtasks, CO.Thorough, CO.FetchBrowse, COS.Focussed two runs were submitted.

Our experiments aim at evaluating the efficiency of the evolution given to the scoring function, the adaptation of the method regarding the different tasks (Thorough, Focussed, Fetch and Browse, SSCAS, VVCAS, …), the new metrics and the evolution of assessment process.

4.1 Experiment setup

One run for all the subtasks except the subtasks Focussed uses the following scoring function:

$$Score(T,E)=\left(\sum_{\forall t\in T} Occ(t,E)\cdot\frac{Occ(t,T)}{Size(T)}\cdot\frac{IndRnk(t)^2}{NbElts(t)}\right)\cdot 400^{(\frac{NbT(T,E)}{NbT(T)})}$$

The runs based on this function are named using the following principle: V2005T<subtask_name> e.g. V2005TCO.Thorough.

Additional runs for the subtasks CO.Thorough, CO.FetchBrowse and COS.Focussed use a scoring function with a function f that measures the importance of a term in an XML element slightly different i.e.:

$$Score(T,E)=\left(\sum_{\forall t\in T}\frac{Occ(t,E)}{Occ(T,E)}\cdot\frac{Occ(t,T)}{Size(T)}\cdot\frac{IndRnk(t)^2}{NbElts(t)}\right)\cdot 400^{(\frac{NbT(T,E)}{NbT(T)})}$$

The runs based on this function are named using the following principle: V2005T**f**<subtask_name> e.g. V2005T**f**CO.Thorough.

For all submitted runs the parameters of the scoring method were the same. The coefficient used to propagate a component score through the hierarchical structure of the XML document was fixed to 0.1. The coverage threshold was fixed to 35% (i.e. more than a third of terms describing the topic must appear in the text to keep the XML component). The coefficients applied to take into account the signs '+' and '-' were fixed to respectively +5.0 or -5.0 to increase or reduce 5 times the contribution of wanted respectively unwanted terms.

The values of the parameters are those which gave the best results during a training phase done with INEX 2003 and INEX 2004 CO topics using the INEX 2004 official metrics.

4.1.1 Subtasks Focussed

The runs submitted for the subtasks Focussed use scoring functions without function p effect (φ=1.0) i.e. without factor measuring the global presence of the topic in the XML element, as follows:

$$Score(T,E)=\sum_{\forall t\in T} f(t,E)\cdot g(t,T)$$

No propagation of score is done to have result without overlapping as requested for the subtask Focussed.

4.1.2 Subtasks CO+S and CAS

For all the subtasks CO+S and CAS the castitle part of topic definition has been used to define queries.

The coefficient taking into account structural constraints on content was fixed to 0.5 (i.e. the contribution of a query term is divided by 2 when the element does not verify the structural constraint associated to the term) for all the subtasks. Since the actual solution implemented in our method cannot fully take into account the structural constraints, we decided to handle them as vague even for XSCAS subtasks.

The coefficient taking into account structural constraints on result granularity was fixed to:

- 0.5 (i.e. the scores of elements not verifying the structural predicates are divided by 2) when expecting vague verification of the constraints i.e. VVCAS and VSCAS,
- 0.0 (i.e. the scores of elements not verifying the structural predicates are reset to zero) when expecting strict verification of the constraints i.e. SSCAS and SVCAS,

The value 0.5 of the two coefficients was fixed arbitrarily.

4.2 Results

Results of the runs for CO subtasks are detailed in the following tables:

	Run	V2005TCO.Focussed				V2005TCO.Thorough				V2005TfCO.Thorough			
	Quantis^{ion}	strict		generalized		strict		generalized		strict		generalized	
Metric		precision	rank	prec.	rank	prec.	rank	prec.	rank	prec.	rank	prec.	rank
	nxCG@10	0.1266	5/44	0.1848	19/44	0.0231	23/55	0.1927	18/55	0.0192	24/55	0.1855	21/55
	nxCG@25	0.0997	8/44	0.1735	17/44	0.0606	15/55	0.206	15/55	0.0409	19/55	0.1785	23/55
	nxCG@50	0.1176	9/44	0.1566	21/44	0.1298	3/55	0.1893	18/55	0.1222	4/55	0.1761	21/55
	ep/gr (MAP)	0.0332	10/44	0.0504	24/44	0.0009	35/55	0.0509	29/55	0.0006	41/55	0.0475	32/55

	Run	V2005TCOS.Focussed				V2005TfCOS.Focussed				V2005TCOS.Thorough			
	Quantis^{ion}	strict		generalized		strict		generalized		strict		generalized	
Metric		precision	rank	prec.	rank	prec.	rank	prec.	rank	prec.	rank	prec.	rank
	nxCG@10	0.0632	7/27	0.1279	19/27	0.0282	16/27	0.054	26/27	0.0269	19/33	0.2178	9/33
	nxCG@25	0.1045	3/27	0.1333	13/27	0.0251	20/27	0.0585	23/27	0.0576	12/33	0.186	12/33
	nxCG@50	0.0924	5/27	0.1334	10/27	0.0621	12/27	0.0754	20/27	0.0874	5/33	0.164	13/33
	ep/gr (MAP)	0.0233	6/27	0.0525	14/27	0.0086	16/27	0.0353	19/27	0.0007	20/33	0.0482	13/33

	Run	V2005TCO.FetchBrowse				V2005TfCO.FetchBrowse				V2005TCOS.FetchBrowse			
	Quantis^{ion}	strict		generalized		strict		generalized		strict		generalized	
Metric		MAP	rank	MAP	rank	MAP	rank	MAP	rank	MAP	rank	MAP	rank
	ep/gr {element}	0.004	8/31	0.071	5/31	0.003	8/31	0.069	6/31	0.003	6/19	0.071	4/19
	ep/gr {article}	0.0195	24/31	0.1399	24/31	0.0152	25/31	0.1309	25/31	0.0188	12/19	0.1304	11/19

Our method seems to be more efficient for the subtasks Focussed than for the subtasks Thorough notably for strict quantisation. For nxCG metric and strict quantisation, the results are particularly good for ranking up to 100.

For the Thorough subtasks the results are on average better for generalised quantisation than for strict quantisation. However, results progress for strict quantisation while they remain stable for generalised quantisation.

For the Fetch and Browse subtasks partial results shows better results at the element level than at the article level.

For CAS topics, results of the runs for CO subtasks are detailled in the following tables:

	Run	V2005TSSCAS				V2005TSVCAS			
	Quantisation	strict		generalized		strict		generalized	
Metric		precision	rank	precision	rank	precision	rank	precision	rank
	nxCG@10	0.1250	11/25	0.3643	4/25	0.1800	4/23	0.324	2/23
	nxCG@25	0.1500	13/25	0.4816	1/25	0.24	7/23	0.3357	3/23
	nxCG@50	0.4078	2/25	0.5192	1/25	0.4422	3/23	0.3799	1/23
	ep/gr (MAP)	0.0156	18/25	0.1265	13/25	0.0127	14/23	0.1301	3/23

	Run	V2005TVSCAS				V2005TVVCAS			
	Quantisation	strict		generalized		strict		generalized	
Metric		precision	rank	precision	rank	precision	rank	precision	rank
	nxCG@10	0.0333	17/24	0.2427	9/24	0.1000	12/28	0.248	14/28
	nxCG@25	0.0600	12/24	0.2435	9/24	0.1267	11/28	0.2544	9/28
	nxCG@50	0.0567	3/24	0.2436	9/24	0.1162	10/28	0.2373	9/28
	ep/gr (MAP)	0.0090	10/24	0.0929	5/24	0.0031	14/28	0.0824	7/28

The results for CAS subtasks are globally good particularly for generalized quantisation. Considering that CAS runs are based on the same scoring function than Thorough runs for CO topics and considering that results of Thorough runs are better for generalised quantisation, it is not surprising to have the same behaviour for CAS runs.

5 DISCUSSION AND FUTURE WORKS

A first analysis of the experiments performed and the obtained results, shows that:

- the chosen functions and parameters for the scoring method seem to be globally adapted to the actual INEX framework. However, the results obtained for the subtasks Thorough show that our method handle overlap not well enough to fully respond to this kind of search. A future work will consist in evolving the method to integrate overlap handling according to different strategies.
- the solutions used to extend our method to handle structural constraints seem to be adequate. However, structural constraints on content are not fully handled by our method actually. To complete the method to handle structural constraints completely is another next step.
- other experiments have to be done to determine the method configurations adapted to each subtask. Furthermore, analyses must be carried out to determine queries processed well by our method and those leading to weaker results. This would enable to evolve the method to better respond to this last type of queries.

Acknowledgments

The research presented in this paper results partly from work undertaken within the framework of the project QUEST: Query reformulation for structured document retrieval, PAI Alliance N°05768UJ. However, this publication only reflects the author's view.

References

1. Clarke C. L. A., Tilker P. L., MultiText Experiments for INEX 2004, Advances in XML Information Retrieval, LNCS 3493, 3rd International Workshop INEX, 2004, p. 85 – 87.
2. Crouch C. J., Apte S., Bapat H., An Approach to Structured Retrieval Based on the Extended Vector Model, Proceedings of the 2nd INEX Workshop, Dagstuhl, Germany, 2003, p. 89-93.
3. Fuhr N., Großjohann K., XIRQL: An XML query language based on information retrieval concepts. ACM Transactions on Information Systems (TOIS), vol. 22, Issue 2, 2004, p. 313-356.
4. Geva S., GPX – Gardens Point XML Information Retrieval at INEX 2004, Advances in XML Information Retrieval, LNCS 3493, 3rd International Workshop INEX, 2004, p. 211 – 223.
5. Grabs T., H.-J. Schek H.-J., Generating Vector Spaces On-the-fly for Flexible XML Retrieval, XML and Information Retrieval Workshop - SIGIR'2002, Tampere, 2002.

6. Hubert G., A voting method for XML retrieval, Advances in XML Information Retrieval, LNCS 3493, 3rd International Workshop INEX, 2004, p. 183-195.
7. Larson R. R., Cheshire II at INEX'03: Component and Algorithm Fusion for XML Retrieval, Proceedings of the 2nd INEX Workshop, Dagstuhl, Germany, 2003, p. 38-45.
8. Mass Y., Mandelbrod M., Component Ranking and Automatic Query Refinement for XML Retrieval, Advances in XML Information Retrieval, LNCS 3493, 3rd International Workshop INEX, 2004, p. 73 – 84.
9. Mihajlović V., Ramírez G., de Vries A. P., Hiemstra D., Blok H. E., TIJAH at INEX 2004 Modeling Phrases and Relevance Feedback, Advances in XML Information Retrieval, LNCS 3493, 3rd International Workshop INEX, 2004, p. 276 – 291.
10. Ogilvie P., Callan J., Hierarchical Language Models for XML Component Retrieval, Advances in XML Information Retrieval, LNCS 3493, 3rd International Workshop INEX, 2004, p. 224 – 237.
11. Sigurbjörnsson B., Kamps J., de Rijke M., Mixture Models, Overlap, and Structural Hints in XML Element Retrieval, Advances in XML Information Retrieval, LNCS 3493, 3rd International Workshop INEX, 2004, p. 196-210.
12. Trotman, A., O'Keefe, R. A.: Identifying and Ranking Relevant Document Elements, Proceedings of the 2nd INEX Workshop, Dagstuhl, Germany, 2003, 149-154.
13. Vittaut J.-N., Piwowarski B., Gallinari P., An Algebra for Structured Queries in Bayesian Networks, Lecture Advances in XML Information Retrieval, LNCS 3493, 3rd International Workshop INEX, 2004, p. 100 – 112.

Experimenting various user models for XML Retrieval

Yosi Mass, Matan Mandelbrod

IBM Research Lab
Haifa 31905, Israel
{yosimass, matan}@il.ibm.com

Abstract. While in previous INEX workshops the ad-hoc task was divided roughly to CO (Content Only) task and CAS (Content and Structure) task, the focus this year was to further refine those tasks so as to experiment with different user behaviors. This paper summarizes the algorithms and approaches used by the IBM Haifa Research Lab for the various CO and CAS sub tasks.

1 Introduction

The challenge in XML retrieval is to return the most relevant components that satisfy the query concepts. While in previous INEX workshops XML retrieval was divided roughly to CO (Content Only) task and CAS (Content and Structure) task, the focus this year was to further refine those tasks so as to measure different user behaviors.

This resulted in three CO sub tasks – CO.Thorough which aims at returning all relevant components, CO.Focussed which aims at returning a single element along any path and the CO.FetchBrowse which is targeted toward browsing model where the full document is browsed first and then its components. To test the importance of structure in queries, each CO query was reformulated with structural hints resulting in CO+S (Content Only plus Structure) topics. Similar to CO, three CO+S sub tasks were defined namely – CO+S.Thorough, CO+S.Focussed and CO+S.FetchBrowse. Finally the CAS task was divided to four sub tasks checking combinations of Strict vs Vagueness in Target elements and in the other query structure elements.

To accommodate those ten sub tasks (3 CO, 3 CO+S & 4 CAS) we applied our component ranking algorithm (see Fig 1) as defined in [5, 6]. The idea is to build different indices for the most informative component types where each index contains elements of the same type. The indices we used this year where for article, abs, bdy, sec, ss1, ss2 and for p and ip1.

We outline below the component ranking algorithm while full details can be found in [6]. Given a query Q, we run the query in parallel on each index (step 1) and then apply an Automatic Query Refinement (AQR) phase (step 2) on each result set. The AQR algorithm we used is a Lexical Affinity (LA) Refinement algorithm which is fully described in [2, 6]. Then in step 3, scores of elements in each result set are normalized to same range so that scores from the different indices can be compared. In step 4 we apply a document pivot scaling where scores of elements from each index are scaled by the score of their parent article. Finally all the results sets are merged into a single result set of all element types.

```
For each index i
    1. Compute the result set $R_i$ of running Q on index i
    2. Apply AQR algorithm on $R_i$
    3. Normalize scores in $R_i$ to [0,1] by normalizing to score(Q,Q)
    4. Scale each score by its containing article score from $R_0$
Merge all $R_i$'s to a single result set R composed of all components sorted by their score
```

Fig. 1. Component ranking algorithm

In this paper we report how we applied this algorithm in the various CO, CO+S and CAS tasks. The rest of the paper is organized as follows: In section 2 we describe our approach and results for the CO runs. In section 3 we describe the CO+S runs and their results and we discuss our findings on the importance of structure in XML queries. Then in section 4 we describe our CAS approach and report results. We conclude in sec 5 with summary and some conclusions.

2 CO runs

We submitted 3 runs for each CO sub task experimenting combinations of using phrases vs ignoring phrases (i.e. treating their words as simple words) and using ‘+’ vs ignoring ‘+’ on words. In general the submission that ignored phrases and ignored ‘+’ outperformed other runs. We detail below our approach for each CO sub task.

2.1 CO.Thorough

This is the traditional CO task as was used in previous INEX workshops. We used the base component algorithm as depicted in Fig 1. Our runs were ranked 1st in the ep/gr generalized metric and quite high in the various nxCG metrics.

2.2 CO.Focussed

A valid CO.Focussed run as defined in [4] should have only one element along any path namely no overlapping elements are allowed. To satisfy this requirement we first perform a regular CO.Thorough run and then filter out the overlaps. The filtering is done in two stages.

In the first stage we try to identify 'clusters' of highly ranked results in the XML tree and pick the most relevant element from each cluster. At the end of this stage there still can be left some overlapping elements so we perform a second filtering stage that picks the highly scored element along each path.

The first stage is performed as follows: We take the result set of the CO.Thorough run and group all elements by their containing article. For each such group we construct a tree with nodes that correspond to the result components and edges that represent the parent-child relationship of the components from the original XML article. We keep for each node its assigned run score and the total number of its descendant in the original article.[1]

To tolerate variations in result scores we compare the scores of two nodes (N_1, N_2) as follows: We compute diff(N_1, N_2) = |score(N_1)–score(N_2)| / score(N_1) and define the following relations between the nodes –

- N_1=N_2 if diff(N_1, N_2) ≤ *ScoreTreshold*
- Otherwise (if diff(N_1, N_2) > ScoreTreshold) then
    - N_1 > N_2 if score(N_1) > score(N_2)
    - N_1 < N_2 if score(N_1) < score(N_2)

In our runs we used *ScoreThreshold* = 0.4. The algorithm processes the result tree bottom up and at each level diagnoses the correlation between the currently examined node (N_1) and its descendents. An example such intermediate tree after score comparison is depicted in Fig 2 where color represents relations to the root N_1 node such that black > gray > white.

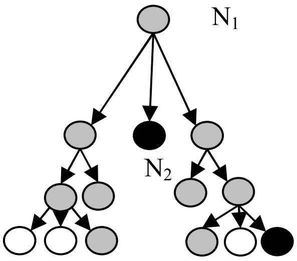


Fig. 2. result tree

The algorithm distinguishes between three main cases-

1. There is some descendant node N_2 with N_2 > N_1. (See fig 2). This means that N_2 is clearly higher than N_1 so we remove N_1 from the result tree.
2. There is some direct child node N_2 such that most of the “good” nodes (descendant nodes that are ≥ N_1) are concentrated under it (see Fig 3 below). This can be measured by defining |Good(N)| as the number of descendant

[1] This number is extracted as part of the indexing procedure, and is stored in the index.

nodes ≥ N_1 and checking if |Good(N_2)|/|Good(N_1)|>*ConcentratedThreshold* for some configured *ConcentratedThreshold.* This means that most of the good results are concentrated under N_2 so we remove N_1 from the result tree. In our runs we used *ConcentratedThreshold* = 0.4.

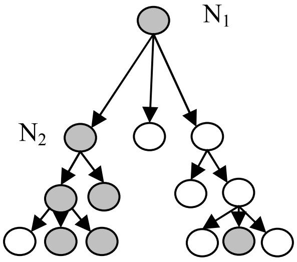


Fig. 3. concentrated child

3. There are enough good results which are evenly distributed below N_1 as depicted in Fig 4 below. This can be measured by checking if |Good(N_1)|/|Descndnt(N_1*)*|>*DescendantTreshhold* where |Descndnt (N_1)| is number of all descendants of N_1 as kept in the index. In our runs we used *DescendantTreshhold* = 0.25. This means that a relative significant part of N_1 is relevant and is not concentrated under a single child so we remove all the descendants from the result tree and keep only N_1.

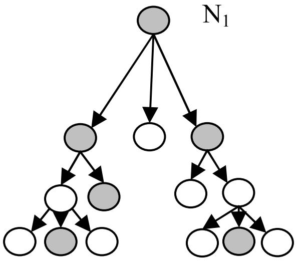


Fig. 4. evenly distributed results

In all other cases (e.g. if there are too few good results under N1) no decision is taken so at the end of this stage there still can be left some overlapping elements.

In the second filtering stage we scan again the reminded result tree from bottom up and at each Node N compare score(N) to the score of all its descendants. If score(N) is bigger we take N and remove all its descendants. Otherwise we remove N from the result set.

Note that the second stage could be performed even without the first stage and return a valid Focussed run. We submitted one run with both stages and second run with only the second stage. As expected the run with both stages performed better and for example in the ep/gr, generalized metric it was ranked 1st with MAP 0.968 while our second run got MAP 0.0909.

2.3 Fetch & Browse

In this task we first run a regular CO.Thorough run. We then pick the article elements by their score and for each article we group its returned elements ranked by their assigned score. We use <rank> instead of <rsv> to order the elements in that submission. Our runs were ranked among the top 10 but not at the top so we still need to investigate this task.

3 CO+S runs

The aim of the CO+S task was to investigate the usefulness of structural hints. For all three sub tasks (CO+S.Thorough, CO+S.Focussed & CO+S.FetchBrowse) we used similar algorithms as in the CO runs applying a VCAS approach on the topic's <castitle>.

The results of most participants show that in general the CO runs performed better than the CO+S runs. Specifically for our submissions the structural hints improved results for the Thorough runs but not for the Focussed runs.

For the Thorough runs our CO+S performed better than the CO in all metrics. For example for the ep/gr, generalised metric our CO+S run got MAP 0.0925 while our CO run got MAP 0.0896. It should be noted that both were ranked 1st in their corresponding metric.

For the Focussed runs our CO performed better than the CO+S in all metrics. For example for the ep/gr, generalised metric our CO.Focussed run got MAP 0.0968 while our CO+S.Focuissed run got MAP 0.0809. Again both were ranked 1st in their corresponding metric.

For the Fetch & Browse runs there was a slight improvement in the CO+S runs.

Maybe the conclusion is that structural hints help only when used as a real filter while having only structure as hints does not help.

4 CAS runs

Similar to previous years the CAS topics were expressed by an XPath[7] expression extended with the *about* vague predicate. XPath defines the last element in the path as a *target element* while all other query elements can be referenced as *support elements*. While in previous years the CAS task was sub classified to Vague (VCAS) and Strics (SCAS) sub tasks, an attempt was made this year to separate the vagueness of the target element from the vagueness of the support elements. As a result a combination of four sub tasks were defined :

- VVCAS – Both target and support are vague
- SSCAS - Both target and support are strict
- SVCAS – Target is strict and support is vague
- VSCAS – Target is vague and support is strict.

We think this separation is artificial so we run our traditional SCAS and VCAS runs using the following mapping from the four INEX tasks to our tasks –

INEX task	**Our submission**
VVCAS	VCAS
SSCAS	SCAS
SVCAS	SCAS
VSCAS	VCAS

Similar to previous years the difference between SCAS and VCAS was in the synonyms. In VCAS runs we use all the considered elements (except the article and the abs) as synonyms to each other namely {bdy, sec, ss1, ss2, p, ip1, bdy}. In SCAS runs we use two synonym groups: {bdy, sec, ss1, ss2} and {p, ip1}.

For each of CAS tasks we submitted two runs. In both runs we treat phrases as simple words and we ignore plus on content. The difference between the two runs was in the plus on the structure.

For example topic 244 –

```
//article[about (.//fm, "query optimization")]//sec[about (., "join query optimization")]
```

Is translated to XML Fragments[1, 3] as

```
<article>
    +<fm>query optimization</fm>
   +<sec>join query optimization</sec>
</article>
```

A '+' on a tag means that the tree below the tag is mandatory. So in the above example a result (<sec>) is returned only if it's containing article has both the <fm> constraint and the <sec> constraint. The default semantics in XML fragments is 'or' so removing the '+' as in

```
<article>
    <fm>query optimization</fm>
   <sec>join query optimization</sec>
</article>
```

will return <sec> even if the containing article does not have the <fm> constraint. Having the <fm> constraint will only increase the score of the containing article and as a result using our document pivot (step 4 in Fig 1) will increase score of the <sec> itself.

This means that for CAS runs we have two levels of vagueness. The first is in the synonym definition and the second is through the '+' on structure. For each of the VCAS and SCAS runs we submitted one run with '+' on the structure and a second run without '+' on structures.

In the sequel we show our performance on the four CAS tasks and it can be clearly seen that having the '+' on structure performs better on the SCAS runs while removing the '+' from structure performs better on the VCAS runs.

4.1 VVCAS results

Both our runs (with and without '+' on structure) were ranked top in the nxCG and the ep/gr in the generalized metric. Still the run which ignored the plus on structure preformed clearly better. This makes sense since it allows more vagueness in the structure.

4.2 SSCAS results

Both our runs were at the top ten and there was no clear preference to the one with the plus on structure or to the other one.

4.3 VSCAS results

Both our runs won top results (1st and 2nd) on both nxCG and ep/gr metrics but with no clear distinctions which of the two is better.

4.4 SVCAS results

Again both runs won top results in most of the metrics where in most cases the run which treat structure strictly was better in most cases. This makes sense for SCAS since it assumes more strictness in the structure.

5 Discussion and summary

We described our approach and algorithms for the various CO, CO+S and CAS tasks. Our main findings are that our component ranking algorithms performed quite well and our runs in all 10 tasks were ranked at top places mostly in the ep/gr generalized metric. We found out that ignoring phrases and '+' gives best results. Regarding structural hints for CO runs we found out that they helped in the Thorough task but disturbed in the Focussed task. Maybe the conclusion is that structure helps only when it is strict (namely as a real filter) while having only structural hints does not help. For CAS runs we found out that the separation of strict/vagueness in target element vs rest of the elements was artificial. Another conclusion is that XML Fragments [1,3] enables another level of strict/vagueness through the '+' on structure.

6 Acknowledgment

We would like to thank the INEX organizers for the assessment tool and for the evalJ tool they have supplied.

References

1 Broder A.Z., Maarek Y., Mandelbrod M. and Y. Mass (2004): "Using XML to Query XML – From Theory to Practice". In Proceedings of RIAO'04, Avignon France, Apr , 2004.

2 Carmel D., Farchi E., Petruschka Y., Soffer A.: Automatic Query Refinement using Lexical Affinities with Maximal Information Gain. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002.

3 Carmel D., Maarek Y., Mandelbrod M., Mass Y., Soffer A.: Searching XML Documents via XML Fragments, In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada, Aug. 2003

4 M. Lalmas, INEX 2005 Retrieval Task and Result Submission Specification, http://inex.is.informatik.uni-duisburg.de/2005/internal/pdf/INEX05_Tasks_v2.pdf, June, 2005

5 Y. Mass, M. Mandelbrod, Retrieving the most relevant XML Component, Proceedings of the Second Workshop of the Initiative for The Evaluation of XML Retrieval (INEX), 15-17 December 2003, Schloss Dagstuhl, Germany, pg 53-58

6 Y. Mass, M. Mandelbrod, Component Ranking and Automatic Query Refinement for XML Retrieval, INEX 2004, Lecture Notes in Computer Science, Springer-Verlag GmbH Volume 3493 / 2005, pg 73-84

7 XPath – XML Path Language (XPath) 2.0, http://www.w3.org/TR/xpath20/

The University of Kaiserslautern at INEX 2005

Philipp Dopichaj

dopichaj@informatik.uni-kl.de
University of Kaiserslautern
Gottlieb-Daimler-Str.
67663 Kaiserslautern
Germany

Abstract. In this paper, we present the two retrieval strategies used by the University of Kaiserslautern at INEX 2005. One strategy uses background knowledge to make better use of inline elements, and the other one attempts to exploit typical structural patterns in the retrieval results.

1 Introduction

The Initiative for the Evaluation of XML Retrieval (INEX)[1] provides a testbed for comparing the effectiveness of competing content-based XML retrieval systems. The University of Kaiserlautern actively participated in the INEX workshop for the first time in 2005. We wanted to evaluate the effects of two orthogonal retrieval approaches, element relationship and context patterns.

We first present a brief description of our baseline retrieval system in Section 2 and then proceed to explain our improvements to this basis in Section 3. Finally, we discuss the performance of our baseline and enhanced results as evaluated in the INEX workshop.

2 Baseline Search Engine

The basic structure of our retrieval system is very simple [2]: We use the Apache Lucene information retrieval engine[2] as the basis and add XML retrieval functionality on top of that. Instead of only storing only the complete articles from the document collection in the index, we store each element's textual contents as a (Lucene) document, enriched with some metadata (most notably, the enclosing XML document and the XPath within that document); see Fig. 1 for an example.

Obviously, directly searching this index using Lucene would lead to bad results—overlap isn't taken into account at all, and many elements on their

[1] see `http://inex.is.informatik.uni-duisburg.de/`

[2] see `http://lucene.apache.org`

```
<sec>Hello, <b>world!</b> How <i>are</i> you?</sec>
```

(a) Input document.

XPath	Indexed contents
/sec[1]	Hello, world! How are you?
/sec[1]/b[1]	world!
/sec[1]/i[1]	are

(b) Indexed documents.

Fig. 1. Source document and correspondig indexed documents as seen by Lucene.

own are useless—, so we need to post-process the Lucene results. We regard the results from different input documents as independent, so we can post-process the results from each document separately (even concurrently); Fig. 2 shows an overview of the retrieval process.

Operation	Output
1. Process query and send it to Lucene	Raw retrieval results (fragments)
2. Rearrange retrieval results	One result tree per document
3. Post-process the result trees	One result tree per document
4. Merge the results	Flat list of results

Fig. 2. The search process in our basic search engine. The enhancements from Section 3 are applied in step 3.

2.1 Query Processing

The queries in the INEX topics are formulated in NEXI, an XML query language derived from XPath with additional information retrieval functions [4]. For content-only (CO) queries, we support the full syntax of NEXI with the following modifications:

- We discard query terms with the "`-`" qualifier, instead of enforcing they do not occur.
- Query terms prefixed with "`+`" are assigned a higher weight, instead of enforcing they do occur.
- The modifiers "`and`" and "`or`" are ignored.

For content-and-structure (CAS) queries, only the last tag name in paths is used for searching (for example, given `//article//fm//atl`, we search all `atl` elements, not only those contained in `//article//fm`. Because of this, we only participated in the VVCAS task, where structural constraints for both the target and support elements are interpreted as vague.

2.2 Length-Based Score Correction

As we have seen above, our search engine stores *all* elements in the index, even inline elements consisting of only a few words (we shall see in Section 3 why we need to store these elements). Of course, these elements are of little use to the searcher, but they might get very high scores—for example, an element that exactly matches the user's query will get a perfect score in the vector space model.

In order to avoid this situation, we multiply each element's score by a factor that solely depends on the length of the element's textual contents. In addition to reducing the score of very short elements (shorter than about 10–20 lines of text), we also reduce the scores of extremely long elements (longer than a typical article). We do this because returning very long elements is typically not useful in XML retrieval, where it is the aim to return the shortest fragments still answering the user's query.

3 Enhancements to the Baseline Search Engine

The search engine we described in the previous section provides the basis for the implementation of our new approaches. On top of it, we implemented two different enhancements that are executed as a post-processing step; they are mostly orthogonal, so they can be applied in any combination.

3.1 Element Relationship

Many XML schemas for document authoring specify tags for semantic markup. DocBook, for example, has a `filename` tag that is used to specify that the contained text designates a file name. This markup is useful, in particular for (CAS) queries, because it enables the searcher to more exactly specify what he wants to retrieve. When we examined real-world documents, we realized that this markup is often not used correctly (possibly because of lazyness on part of the authors, possibly because no tag exactly matching the author's intention exists). We had the idea to create a graph for allowing near misses of the markup specified in structural queries, the *element relationship graph* (ERG) [1,2].

The ERG contains as leaves the tag names from the document schema and places them in a semi-hierarchical graph that captures semantic relations between the tag names. Each category is assigned a coherence value in the range zero to one that denotes to what degree the contained tag names are similar; this information is used for similarity calculation, see below for an example.

The approach is not well suited to visual markup that only denotes how the marked-up text should look, instead of what the semantics are. Unfortunately, the collection of IEEE magazine articles that is used for INEX uses only visual markup for the body of the text (the bibliography is more structured, but it is rarely the target of queries); we tried to construct an ERG for this data anyway to see how well element relationship can cope with situations it wasn't designed for.

We based our ERG on the information available in `xmlarticle.dtd`. In addition to the purely syntactic information used by the XML parser, the DTD also contains consistently formatted comments that indicate a two-level hierarchical structure, as we can see in Fig. 3. We wrote a small script to convert this DTD to an ERG, assigning a coherence of 0.5 to all second-level headings and of 0.2 to the first-level headings.

```
<!-- ============ -->
<!-- FRONT MATTER -->
<!-- ============ -->

<!ELEMENT fm   (hdr?, (edinfo|au|tig|pubfm|abs|edintro|kwd|fig|figw)*)>

<!-- ++++++ -->
<!-- HEADER -->
<!-- ++++++ -->

<!ELEMENT hdr  (fig?, hdr1, hdr2)>

<!ELEMENT hdr1 (#PCDATA|crt|obi|pdt|pp|ti)*>
<!ELEMENT hdr2 (#PCDATA|crt|obi|pdt|pp|ti)*>
```

Fig. 3. Excerpt from `xmlarticle.dtd`. We can see that the comments indicate the semantic structure of the elements: *Front Matter* is a first-level heading, and *Header* is a second-level heading.

The last tag name from the path in the NEXI query is taken as the category to search in. If a retrieval result is embedded in an element with that tag name, its score is taken as is, otherwise we go up in the ERG and try to match any tag name from the same category, reducing the score by multiplying it with the corresponding coherence. For example, if we search in `hdr`, but a match is in a `hdr1` element, we halve the original score because we needed to generalize to a second-level category.

For more details about applying element relationship, see our previous work on this topic [1,2].

3.2 Context Patterns

Exploiting element relationships is only feasible if the schema(s) of the document collection are fixed and one is willing and able to create an element relationship graph. If this is not the case, one needs schema-independent methods to improve retrieval results. Fortunately, although tag names may differ, there are several telltale signs what the role of a given element in a text is—without even looking at the tag name.

We can achieve this by looking at *result contexts* of the retrieved nodes. For each non-leaf node, the result context consists of this node and its children, and the following data is stored for each node:

- The retrieval score of the node,
- the length of the node (in tokens/words), and
- the position of the node in the parent node.

This information can be visualized in two dimensions, one for the lengths and positions of the text fragments and the other for the score. Fig. 4 shows an example XML fragment and how it can be visualized. The horizontal position of the left-hand side of each rectangle denotes the starting position in the text of the parent element, and its width corresponds to the length of the text it contains (this implies that the parent element occupies the width of the diagram). The parent element (in the Fig. 1, the root element `/sec[1]`) is the reference for the scale of the horizontal axis.

```
<sec>
Hello, <b>world!</b>
How <i>are</i> you?
</sec>
```

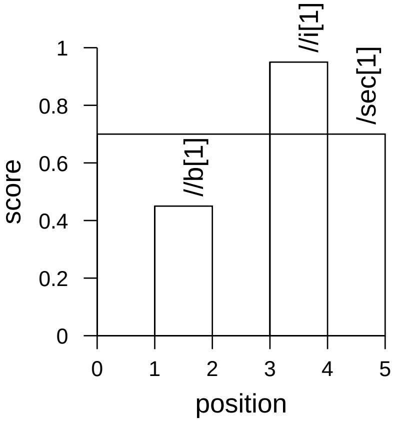


Fig. 4. XML text and corresponding context diagram. The horizontal axis denotes the positions and lengths of the text fragments, and the vertical axis shows the RSV (in this case random numbers).

When we examined context graphs of some trial retrieval results, we realized that we could often determine what elements were section titles or inline elements, without referring to the original XML documents. Based on this observation, we defined a set of *context patterns* for formalizing the recognition of certain structures. Such a pattern looks like, “if the first child in the context is short and the parent is long, the first child is a title” (see Fig. 5 for an example);

obviously, this is too vague for Boolean logic, but fuzzy logic is perfectly suitable for this task.

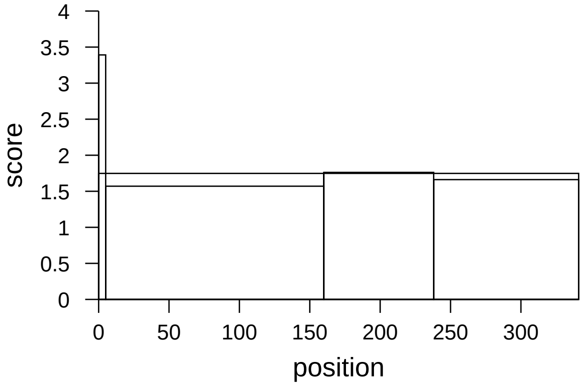


Fig. 5. Example context graph for the title pattern. The short peak at the left is the section title.

Fuzzy logic enables us to assign degrees of membership for the features, instead of Boolean values [3]. For example, a fragment containing only one word is definitely short, and a fragment containing 5000 words is definitely not short, but what about one containing 20 words? With fuzzy logic, we do not need to make a firm decision, but we can say that this fragment is short to a degree of (for example) 50 %. Similarly, the Boolean operators like *and*, *or*, and *not* can be expressed in terms of these degrees.

Obviously, the patterns alone are not of much help, we need to take actions for modifying the relevant scores. For a match in a title, an appropriate action is to increase the parent's score (because a match in the title indicates that the corresponding section is highly relevant) and decrease the first child's score (because the title itself contains too little information to be of any use).

We defined and examined several patterns; apart from the title pattern mentioned above, the inline pattern proved to be the most worthwhile. It is based on the assumption that single words or short phrases directly contained in any markup denote some form of emphasis (in the IEEE collection, very short marked-up elements are typically embedded in `b` or `i` elements, denoting bold and italics). If the author of the text decided to apply such emphasizing markup to phrases, this is often an indication that the surrounding element is especially relevant for queries mentioning the phrases. Because of this, if many of an element's children are very short and have high scores, we increase the element's score. Fig. 6 shows an example of an occurance of this pattern.

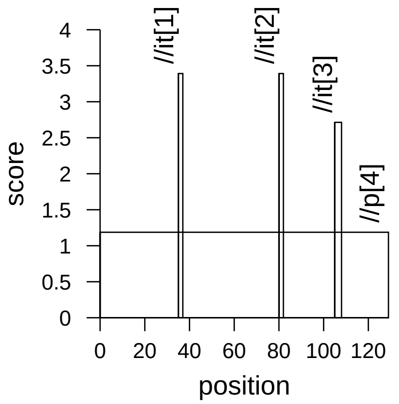


Fig. 6. An example for the inline pattern.

4 Evaluation

One important aspect of INEX is the comparison of XML search engines. In this section, we shall describe what runs we submitted, examine the official results and present some post-INEX improvements of our methods.

4.1 Submitted Runs

We only participated in several sub-tasks of the ad-hoc task. For each of the CO and CO+S tasks, both focused and thorough, we submitted three runs:

1. *Basic,* which applied both element relationship and length-based score correction to the Lucene results (this was our baseline).
2. *Pattern,* which applied element relationship, length-based score correction and context patterns.
3. *Pattern-NoERG,* which applied length-based score correction and context patterns.

For the runs based on element relationship, we searched for the query terms in the category *Emphasis*, which contains inline elements for printing in bold or italics. As we shall later see, the selection of runs turned out to be a bad choice, since element relationship actually downgraded the retrieval quality of our systems for the content-only (CO) tasks. Because of this, we have no baseline for our best-performing run in the official results.

Our system does not support any type of strict CAS queries, as the element-relationship approach was designed with vague structural matching in mind, so we did not submit any runs to the VSCAS and SSCAS sub-tasks. For the

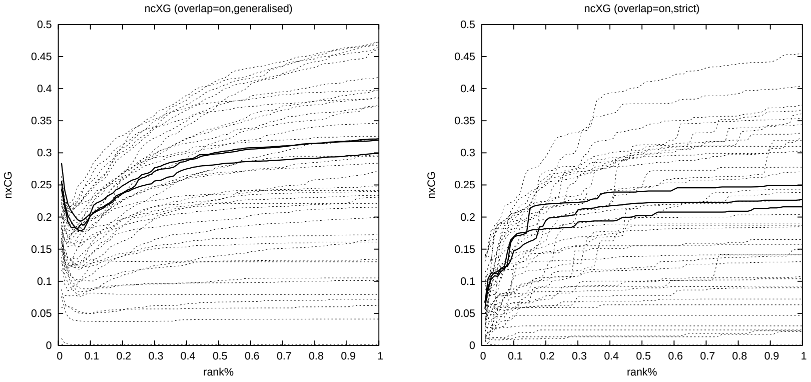
VVCAS sub-task, where only two runs per organization were permitted, we included only the last two of the runs mentioned above, pattern-based retrieval with resprectively without using element relationship.

For the focused sub-tasks, we used our thorough results and applied some post processing to each result tree: We repeatedly added the result with the highest RSV to the retrieval result and removed all results that overlapped this one.

4.2 Official Results

This year's INEX workshop offered a plethora of retrieval tasks and evaluation metrics because there are different views on what constitutes a good retrieval result; because of this, it is difficult to make clear statements. Nevertheless, the following points are fairly clear (see Figures 7 and 8):

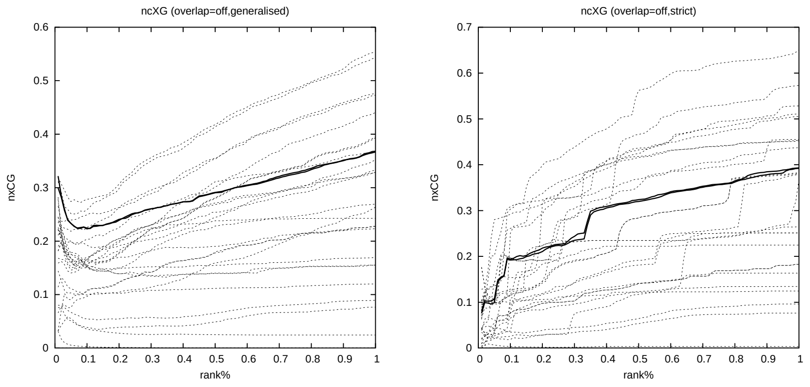
- Our system is more competitive with generalized quantization; with strict quantization, our ranks drop significantly.
- For the top-ranked results up to roughly the 30th place, we fare better compared to the competition than for the lower-ranked results.
- Only for the top-ranked results, the pattern-based approach is better than the corresponding baseline; as we later found out, this is due to undesirable interactions of several context patterns (see the next section).
- Employing element relationships did not lead to noticeable improvements for VVCAS, and actually degraded retrieval quality for the CO runs; although the schemas of the IEEE document collection that is used for INEX is not well-suited to our approach, we has expected a better outcome and will need to investigate further what the cause is.



(a) quantization: generalized

(b) quantization: strict

Fig. 7. Official INEX 2005 results for CO.Focussed, metric nxCG.



(a) quantization: generalized

(b) quantization: strict

Fig. 8. Official INEX 2005 results for VVCAS, metric nxCG.

In the best case, for the CO tasks (both thorough and focused) and generalized quantization, our pattern-based approach managed to outperform the other submissions by a small margin up to roughly the 30th result rank.

4.3 Post-INEX Evaluation

The counterintuitive results for our pattern-based runs—better than the baseline at low cut-off points, significantly worse at higher cut-off points—prompted us to perform further analysis. The original implementation that was used for the runs submitted to INEX evaluated several patterns without properly isolating them, so we re-implemented that short after the deadline has passed. An evaluation of this new implementation based on the INEX assessents reveals that this does indeed appear to be the cause for the bad quality at higher cut-off values (see Fig. 9).

We also evaluated the effect of the patterns we had used for the INEX submissions and found that only two of them have any noticeable effect on retrieval quality, the title pattern and the inline pattern described in Section 3.2.

Another interesting observation is that applying the two patterns in combination leads to worse results than applying the title pattern alone for the top-ranked documents (it does improve results for the lower ranks), as we can see in Fig. 10.

5 Conclusions

As we have seen, the runs applying element relationships failed badly for the CO tasks and did not produce a consistent improvement even for the VVCAS and CO+S runs; this can in part be explained by the mismatch of the type of

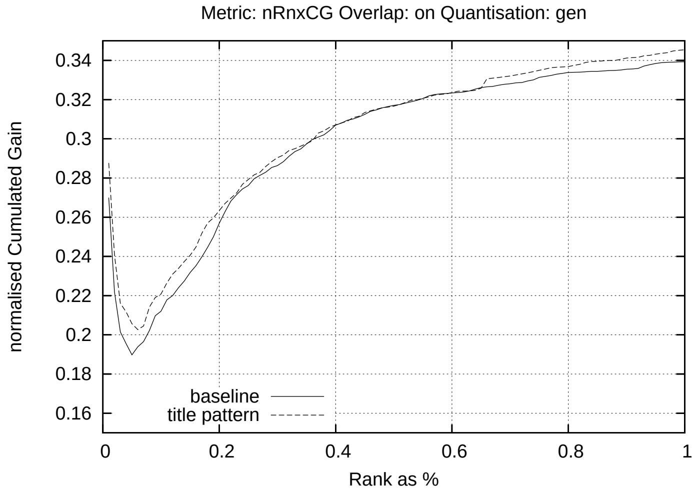


Fig. 9. Post-INEX evaluation of baseline versus only the title pattern (without element relationship, task CO.Focussed).

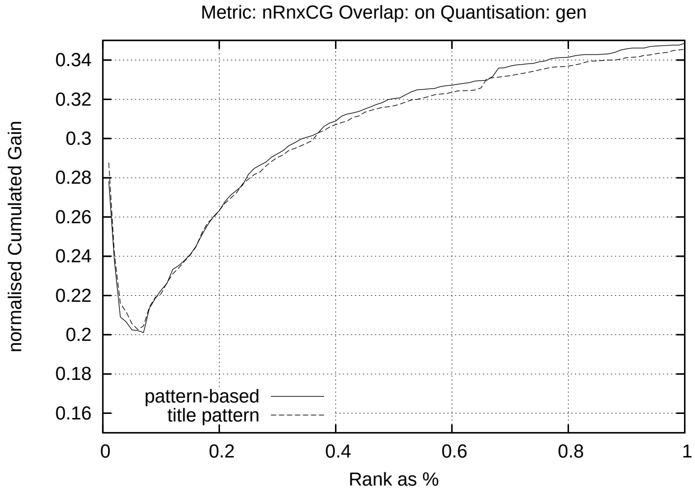


Fig. 10. Post-INEX evaluation of inline and title versus only the title pattern (without element relationship, task CO.Focussed).

markup expected by this method and the markup supplied by the document collection.

Context patterns showed more promising results, but we still need to investigate why the quality of our retrieval results declines more rapidly than those of the other participants.

References

1. Philipp Dopichaj. Element relationship: Exploiting inline markup for better XML retrieval. In Gottfried Vossen, Frank Leymann, Peter C. Lockemann, and Wolffried Stucky, editors, *Datenbanksysteme in Business, Technologie und Web, 11. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), Karlsruhe, 2.-4. März 2005*, volume 65 of *LNI*, pages 285–294. GI, 2005.
2. Benedikt Eger. Entwurf und Implementierung einer XML-Volltext-Suchmaschine. Master's thesis, University of Kaiserslautern, 2005.
3. Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*, chapter 13. Springer, 2nd edition, 2004.
4. Andrew Trotman and Börkur Sigurbjörnsson. Narrow extended XPath I (NEXI). In Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltán Szlávik, editors, *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004*. Springer, 2005.

Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval

Paul Ogilvie and Jamie Callan

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
pto@lti.cs.cmu.edu, callan@lti.cs.cmu.edu

Abstract. This paper explores the possibility of using a modified Expectation-Maximization algorithm to estimate parameters for a simple hierarchical generative model for XML retrieval. The generative model for an XML component is estimated by linearly interpolating statistical language models estimated from the text of the component itself, the parent component, the document component, and its children. We modify EM to allow the incorporation of negative examples, then attempt to maximize the likelihood of the relevant components while minimizing the likelihood of non-relevant components found in training data. This provides an effective algorithm to estimate the parameters in the linear combination mentioned above. Some experiments are presented on the CO.Thorough task that support these claims.

1 Introduction

In previous work [1][2][3], we proposed using hierarchical language models for ranking XML document components for retrieval. However, we left the problem of estimating parameters as future work. In this work, we present a parameter estimate method for a simplified version of the hierarchical language models.

Similar to the language models we estimated in the past, we construct a language model for each component in the document. What is different from previous work is that we do not recursively smooth the language models. Instead, we linearly interpolate the parent's unsmoothed language model, each child's unsmoothed language model, the document's unsmoothed language model, and the collection language model. This simplification allows us to formulate the parameter estimation problem simply so that we can apply the Generalized Expectation Maximization algorithm.

However, we observed that this approach places the most weight on the document language model, which results in very poor retrieval performance. We modify the likelihood we wish to maximize by including negative examples. These negative examples are non-relevant components that come from documents that contain relevant components. To include these negative examples in the likelihood we raise one minus the probability of the unsmoothed language model generating the query term to a very large power so that it is on a similar scale to that of the language models for relevant components. While this is an ad-hoc method for including the negative examples, we have found it to work well in practice.

The next section describes the model in detail, and Section 3 presents the Generalized Expectation Maximization (GEM) algorithm for the model. Section 4 presents our adaptation of the GEM algorithm to include negative examples. We present experimental methodology and describe our system in Section 5. Section 6 contains our experiments with using the GEM algorithm on CO.Thorough task, and conclusions and discussion is contained in Section 7.

2 Model

We rank components by estimating the probability that the a language model estimated for the component generated the query. We use simple unigram language models, which are multinomial probability distributions over words in the vocabulary. That is, a language model μ specifies $P\left(w\left|\mu\right.\right)$. Document components (or elements) are then ordered by $P\left(Q\left|\mu_e\right.\right) = \prod_{i=1}^{|Q|} P\left(q_i\left|\mu_e\right.\right)$ where μ_e is the language model estimated for a particular element e.

In order to estimate the language model μ_e, we note that we would like to incorporate evidence from the document, its parent, and its children. With that in mind, we estimate μ_e as a linear combination of several language models:

$$\begin{array}{rl} P\left(w\left|\mu_e\right.\right) = & \lambda_P P\left(w\left|\theta_{P(e)}\right.\right) \\ & +\lambda_D P\left(w\left|\theta_{d(e)}\right.\right) \\ & +\lambda_C P\left(w\left|\theta_C\right.\right) \\ & +\lambda_O \frac{|s(e)|}{|s(e)|+\sum_{j'\in c(i)} \alpha_{t(j')}|j'|} P\left(w\left|\lambda_{s(e)}\right.\right) \\ & +\lambda_O \sum_{j\in c(e)} \frac{\alpha_{t(j')}|j|}{|s(e)|+\sum_{j'\in c(i)} \alpha_{t(j')}|j'|} P\left(w\left|\lambda_j\right.\right) \end{array} \qquad (1)$$

where θ_x refers to a language model estimated for x, $P\left(x\right)$ refers to the parent of x, $d\left(x\right)$ refers to the document containing x, $s\left(x\right)$ refers to the component x (self), $c\left(x\right)$ returns a list containing the children of x, $t\left(x\right)$ refers to the type of the element x, and C refers to the entire collection. We choose to set the λ parameters in the interpolation to be constant across all elements in the collection to reduce the number of parameters we must estimate. The α parameters allow us to provide additional weight to the children of components, where the weight is dependent on the type of the child component. Note that we also multiply alpha by the length of the component, which results in an assumption that the extra value of a child component is dependent on both the type and length of the child.

In this work, we will take θ_x to be the Maximum Likelihood Estimate from the text contained in x, which is given by:

$$P\left(w\left|\theta_x\right.\right) = \frac{\text{count of } w \text{ in text of } x}{\text{length in words of text of } x} \qquad (2)$$

Note that this is different than our previous work. In our previous work, we excluded the text of the child's components when performing hierarchical smoothing. In this model we include that text. This allows a more clear and consistent parameter estimation scheme. The α_t parameters represent the *additional* value of a word in components of type t. Additionally, we do not recursively smooth the components. This is a limiting factor in current work that simplifies the parameter estimation process.

Unfortunately, due to a bug in our system we did not rank components by $P\left(Q\left|\mu_e\right.\right)$. In our official submissions, we ranked by

$$\begin{gathered} P\left(Q\left|\theta'_{P(e)}\right.\right)^{\lambda_P} \times P\left(Q\left|\theta'_{d(e)}\right.\right)^{\lambda_D} \\ \times P\left(Q\left|\theta'_{s(e)}\right.\right)^{\lambda_O \frac{|s(i)|}{|s(i)|+\sum_{j'\in c(i)} \alpha_{t(j')}|j'|}} \\ \times \prod_{j\in c(e)} P\left(Q\left|\theta'_j\right.\right)^{\lambda_O \sum_{j\in c(e)} \frac{\alpha_{t(j')}|j|}{|s(e)|+\sum_{j'\in c(i)} \alpha_{t(j')}|j'|}} \end{gathered} \tag{3}$$

where

$$P\left(w\left|\theta'_x\right.\right) = (1-\lambda_C)P\left(w\left|\theta_x\right.\right) + \lambda_C P\left(w\left|\theta_C\right.\right) \tag{4}$$

This model does allow relative weighting of the different structural components of messages in the thread. However, it does not have the intended effect of combining evidence at the word level; it only combines query level evidence. This model corresponds to the linear weighted combination of log probabilities, which we investigated in [4]. We will refer to ranking by $P\left(Q\left|\theta_e\right.\right)$ as the mixture method and Equation 3 as the post query combination approach.

Rather than discuss our official submissions in Section 6, we will present experiments using the corrected $P\left(Q\left|\theta_e\right.\right)$. We also apply a linear length prior [5] to our rankings. That is, we multiply $P\left(Q\left|\theta_e\right.\right)$ by $\mathit{length}\,(e)$ to obtain the retrieval status values used in our rankings.

3 Parameter Estimation Using EM

This section describes how we estimate parameters for ranking results by $P\left(Q\left|\theta_e\right.\right)$.

Suppose there are M language models in the collection, which we will denote

$$\theta_1, \theta_2, \ldots, \theta_M.$$

Suppose that we are given some queries and rankable components that are relevant to these queries. We will treat words in these queries as observations from the relevant components:

$$\mathbf{x} = \left(x_1, x_2, \ldots, x_N\right),$$

where we denote the relevant components as

$$\mu_1, \mu_2, \ldots, \mu_N.$$

Note that there may be repeated query terms and components in these lists; this is not an issue in the estimation process.

Let us now assume that the μ components are linear interpolations of the components, giving:

$$P\left(x\left|\mu_i\right.\right) = \sum_{j=1}^{M} \lambda_{ij} P\left(x\left|\theta_j\right.\right). \tag{5}$$

This results in a model where we do not know the $\Lambda = (\lambda_{11}, \ldots, \lambda_{NM})$ parameters.

We would like to maximize the probability of $P\left(\mathbf{x}\left|\mu\right.\right)$. In order to reduce the number of parameters we must estimate in this model, we will assume that each μ_i is estimated from using a small number of components we will call the $family$ of i. In relation to the model presented before, the $family$ of i will be child components, the collection component, the parent component, the document component and the component itself:

$$\mathit{family}\left(i\right) = \left(\theta_1, \theta_{document(i)}, \theta_{parent(i)}, \theta_{self(i)}\right) \cup_{k \in children(i)} \left(\theta_k\right)$$

or using the first letter as an abbreviation for the *document*, *parent*, $self$ and *children* functions:

$$\mathit{family}\left(i\right) = \left(\theta_1, \theta_{d(i)}, \theta_{p(i)}, \theta_{s(i)}\right) \cup_{k \in c(i)} \left(\theta_k\right). \tag{6}$$

where θ_1 is the special collection model used for smoothing. Given the $family$ of component i, we can rewrite Equation 5 as

$$P\left(x\left|\mu_i\right.\right) = \sum_{j \in f(i)} \lambda_{ij} P\left(x\left|\theta_j\right.\right), \tag{7}$$

greatly reducing the number of parameters we must estimate. Note that we also place the constraints

$$\lambda_{ij} \geq 0, \quad \sum_{j \in f(i)} \lambda_{ij} = 1 \tag{8}$$

upon the Λ parameters.

However, there are still many cases where we must estimate λ parameters for texts and we have no training data, as the $\mathbf{x}$ vector is very small in comparison to the total number of rankable texts in the corpus. We must make further assumptions to reduce the parameter space. Given our understanding of the XML retrieval domain, we will assume constant parameters across all models for the combination with the *collection*, *document* and *parent* components. For the *children* components, we will assume that the weight placed should be a simple function of the t = *type* of the child component and its length. Under these assumptions:

$$\lambda_{ij} = \begin{cases} \lambda_C & \text{if } j = 1, \\ \lambda_D & \text{if } j = d\left(i\right), \\ \lambda_P & \text{if } j = p\left(i\right), \\ \lambda_O \frac{|j|}{|s(i)|+\sum_{j' c(i)} e^{\beta_{t(j')}} |j'|} & \text{if } j = s\left(i\right), \\ \lambda_O \frac{e^{\beta_{t(j)}} |j|}{|s(i)|+\sum_{j' c(i)} e^{\beta_{t(j')}} |j'|} & \text{if } j \in c\left(i\right), \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

where the *type* function returns a value in $(1, 2, \ldots, T)$. This now greatly reduces the number of parameters we must estimate to $T + 4$. In addition to the constraints in Equation 8, we place this additional constraint:

$$\lambda_C + \lambda_D + \lambda_P + \lambda_O = 1 \tag{10}$$

Note that we reparameterized α_k as e^{β_k} as this will ensure that α_k is positive. Given Equation 9, we can rewrite Equation 7 using the parameters we must estimate:

$$\begin{aligned} P\left(x\left|\mu_i\right.\right) &= \lambda_C P\left(x\left|\theta_1\right.\right) + \lambda_D P\left(x\left|\theta_{d(i)}\right.\right) + \lambda_P P\left(x\left|\theta_{p(i)}\right.\right) \\ &+ \lambda_O \begin{pmatrix} \frac{|i|}{|i|+\sum_{j\in c(i)} e^{\beta_{t(j)}} |j|} P\left(x\left|\theta_{s(i)}\right.\right) + \\ \sum_{j\in c(i)} \frac{e^{\beta_{t(j)}} |j|}{|i|+\sum_{j\in c(i)} e^{\beta_{t(j)}} |j|} P\left(x\left|\theta_j\right.\right) \end{pmatrix} \end{aligned} \tag{11}$$

We would like to maximize the likelihood of the observed data, which is

$$\mathcal{L}\left(\mathit{\Lambda}\left|\mathcal{X}\right.\right) = P\left(\mathbf{x}\left|\mu\right.\right) = \prod_{i=1}^{N} P\left(x_i\left|\mu_i\right.\right) = \prod_{i=1}^{N}\sum_{j=1}^{M} \lambda_{ij} P\left(x_i\left|\theta_j\right.\right) \tag{12}$$

Unfortunately, the summation within the product makes it difficult to differentiate, so we must use an alternative approach to maximizing the likelihood. We choose to use the Expectation-Maximization method to optimizing the likelihood.

Suppose we were given additional information $\mathcal{Y} = (y_1, \ldots, y_N)$ which specify that the θ_{y_i} distribution generated the x_i query term. Given knowledge of $\mathbf{y}$, the likelihood becomes

$$\mathcal{L}\left(\mathit{\Lambda}\left|\mathcal{X},\mathcal{Y}\right.\right) = \prod_{i=1}^{N} \lambda_{iy_i} P\left(x_i\left|\theta_{y_i}\right.\right) \tag{13}$$

and the log-likelihood of the data is then

$$\log \mathcal{L}\left(\mathit{\Lambda}\left|\mathcal{X},\mathcal{Y}\right.\right) = \sum_{i=1}^{N} \log\left(\lambda_{iy_i} P\left(x_i\left|\theta_{y_i}\right.\right)\right) \tag{14}$$

The problem is now that we do not know the values of $\mathcal{Y}$. However, we may treat it as a random vector and apply Expectation-Maximization.

Suppose we have a guess at the $\mathit{\Lambda}$ parameters we shall call $\mathit{\Lambda}^g$. Using $\mathit{\Lambda}^g$ we can compute $P\left(x_i\left|\mu_j^g\right.\right)$. Applying Bayes rule, we calculate

$$P\left(y_i\left|x_i,\mathit{\Lambda}^g\right.\right) = \frac{\lambda_{iy_i}^g P\left(x_i\left|\theta_{y_i}\right.\right)}{P\left(x_i\left|\mathit{\Lambda}^g\right.\right)} = \frac{\lambda_{iy_i}^g P\left(x_i\left|\theta_{y_i}\right.\right)}{\sum_{j=1}^{M} \lambda_{ij}^g P\left(x_i\left|\theta_j\right.\right)} = \frac{\lambda_{iy_i}^g P\left(x_i\left|\theta_{y_i}\right.\right)}{\sum_{j\in family(i)} \lambda_{ij}^g P\left(x_i\left|\theta_j\right.\right)} \tag{15}$$

and

$$P\left(\mathbf{y}\left|\mathcal{X},\mathit{\Lambda}^g\right.\right) = \prod_{i=1}^{N} P\left(y_i\left|x_i,\mathit{\Lambda}^g\right.\right) \tag{16}$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ is independently drawn value of the random vector.

We may now estimate the expectation of $\mathit{\Lambda}$ given $\mathit{\Lambda}^g$:

$$\begin{aligned} Q\left(\mathit{\Lambda},\mathit{\Lambda}^g\right) &= \textstyle\sum_{\mathbf{y}\in\mathit{\Upsilon}} \log\left(\mathcal{L}\left(\mathit{\Lambda}\left|\mathcal{X},\mathbf{y}\right.\right)\right) P\left(\mathbf{y}\left|\mathcal{X},\mathit{\Lambda}^g\right.\right) \\ &= \textstyle\sum_{l=1}^{M}\sum_{i=1}^{N} \log\left(\lambda_{il} P\left(x_i\left|\theta_l\right.\right)\right) P\left(l\left|x_i,\mathit{\Lambda}^g\right.\right) \end{aligned} \tag{17}$$

At this point we observe that to maximize this equation, we must take the partial derivative of $Q\left(\mathit{\Lambda},\mathit{\Lambda}^g\right)$ with respect to each of the $\mathit{\Lambda}$ parameters.

To maximize λ_C, we must introduce the Lagrange multiplier ϕ with the constraint that $\lambda_C + \lambda_D + \lambda_P + \lambda_O = 1$ and solve the following equation:

$$\begin{gathered} \tfrac{\partial}{\partial\lambda_C}\left[\textstyle\sum_{l=1}^{M}\sum_{i=1}^{N} \log\left(\lambda_{il} P\left(x_i\left|\theta_l\right.\right)\right) P\left(l\left|x_i,\mathit{\Lambda}^g\right.\right) + \phi\left(\lambda_C + \lambda_D + \lambda_P + \lambda_O - 1\right)\right] = 0 \\ \tfrac{\partial}{\partial\lambda_C}\begin{bmatrix} \sum_{i=1}^{N} \log\left(\lambda_C\right) P\left(y = 1\left|x_i,\mathit{\Lambda}^g\right.\right) + \phi\lambda_C \\ \\ +\text{some constants with respect to } \lambda_C \end{bmatrix} = 0 \\ \tfrac{1}{\lambda_C}\textstyle\sum_{i=1}^{N} P\left(y = 1\left|x_i,\mathit{\Lambda}^g\right.\right) + \phi = 0 \end{gathered} \tag{18}$$

Similarly, to maximize λ_D, λ_P, and λ_O, we use

$$\begin{gathered} \tfrac{1}{\lambda_D}\textstyle\sum_{i=1}^{N} P\left(y = d\left(i\right)\left|x_i,\mathit{\Lambda}^g\right.\right) + \phi = 0 \\ \tfrac{1}{\lambda_P}\textstyle\sum_{i=1}^{N} P\left(y = p\left(i\right)\left|x_i,\mathit{\Lambda}^g\right.\right) + \phi = 0 \\ \tfrac{1}{\lambda_O}\textstyle\sum_{i=1}^{N}\sum_{j\in(s(i))\cup c(i)} P\left(y = j\left|x_i,\mathit{\Lambda}^g\right.\right) + \phi = 0 \end{gathered} \tag{19}$$

By summing these equations we get $\phi = -N$. We can then obtain the following update rules:

$$\begin{aligned}
\lambda_C^{[t]} &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} P\left(y = 1\left|x_i, \mathit{\Lambda}^{[t-1]}\right.\right)\\
\lambda_D^{[t]} &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} P\left(y = d\left(i\right)\left|x_i, \mathit{\Lambda}^{[t-1]}\right.\right)\\
\lambda_P^{[t]} &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} P\left(y = p\left(i\right)\left|x_i, \mathit{\Lambda}^{[t-1]}\right.\right)\\
\lambda_O^{[t]} &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \sum_{j\in(s(i))\cup c(i)} P\left(y = j\left|x_i, \mathit{\Lambda}^{[t-1]}\right.\right)
\end{aligned} \tag{20}$$

Let us continue to the β_k parameters. Let

$$\begin{aligned}
a_{ik} &= \textstyle\sum_{j'\in c(i),t(j')=k} \left|j'\right|\\
b_{ik} &= \left|s\left(i\right)\right| + \textstyle\sum_{j'\in c(i),t(j')\neq k} \beta_{t(j')}\left|j'\right|\\
f_{ik} &= P\left(y = s\left(i\right)\left|x_i, \mathit{\Lambda}^g\right.\right) + \textstyle\sum_{j\in c(i),t(j)\neq k} P\left(y = j\left|x_i, \mathit{\Lambda}^g\right.\right)\\
h_{ik} &= \textstyle\sum_{j\in c(i),t(j)=k} P\left(y = j\left|x_i, \mathit{\Lambda}^g\right.\right)
\end{aligned} \tag{21}$$

Then we can rewrite the above as

$$\tfrac{\partial}{\partial\beta_k}\left[\begin{array}{c}\textstyle\sum_{i:j\in c(i),t(j)=k}\left[\begin{array}{c}\log\left(\tfrac{\left|s(i)\right|}{b_{ik}+e^{\beta_k}a_{ik}}\right)P\left(y = s\left(i\right)\left|x_i, \mathit{\Lambda}^g\right.\right)\\
+\textstyle\sum_{j\in c(i),t(j)=k}\log\left(\tfrac{e^{\beta_k}\left|j\right|}{b_{ik}+e^{\beta_k}a_{ik}}\right)P\left(y = j\left|x_i, \mathit{\Lambda}^g\right.\right)\\
+\textstyle\sum_{j\in c(i),t(j)\neq k}\log\left(\tfrac{e^{\beta_{t(j)}}\left|j\right|}{b_{ik}+e^{\beta_k}a_{ik}}\right)P\left(y = j\left|x_i, \mathit{\Lambda}^g\right.\right)\end{array}\right]\\
+\text{some constants with respect to } \beta_k\end{array}\right] = 0 \tag{22}$$

We first take the chain rule, resulting in the multiplier β_k, then take the partial derivative of the summation with respect to β_k

$$e^{\beta_k}\textstyle\sum_{i:j\in c(i),t(j)=k}\left[\begin{array}{c}\tfrac{-a_{ik}}{b_{ik}+\beta_k a_{ik}}P\left(y = s\left(i\right)\left|x_i, \mathit{\Lambda}^g\right.\right)\\
+\textstyle\sum_{j\in c(i),t(j)=k}\tfrac{b_{ik}}{e^{\beta_k}\left(b_{ik}+e^{\beta_k}a_{ik}\right)}P\left(y = j\left|x_i, \mathit{\Lambda}^g\right.\right)\\
+\textstyle\sum_{j\in c(i),t(j)\neq k}\tfrac{-a_{ik}}{b_{ik}+e^{\beta_k}a_{ik}}P\left(y = j\left|x_i, \mathit{\Lambda}^g\right.\right)\end{array}\right] = 0 \tag{23}$$

$$\beta_k \sum_{i:j\in c(i),t(j)=k} \frac{-a_{ik}f_{ik} + \frac{b_{ik}h_{ik}}{e^{\beta_k}}}{b_{ik} + e^{\beta_k}a_{ik}} = 0 \tag{24}$$

Since we cannot solve directly solve this equation for β_k, we will use a linear approximation around the point β_k^g:

$$\frac{\partial}{\partial \beta_k} Q\left(\Lambda, \Lambda^g\right) \approx \frac{\partial}{\partial \beta_k} Q\left(\Lambda, \Lambda^g\right)_{\beta_k=\beta_k^g} + \left(\beta_k - \beta_k^g\right) \frac{\partial^2}{\partial \beta_k^2} Q\left(\Lambda, \Lambda^g\right)_{\beta_k=\beta_k^g} \tag{25}$$

Since we set $\frac{\partial}{\partial \beta_k} Q\left(\Lambda, \Lambda^g\right) = 0$,

$$\beta_k \approx \beta_k^g - \frac{\frac{\partial}{\partial \beta_k} Q\left(\Lambda, \Lambda^g\right)_{\beta_k=\beta_k^g}}{\frac{\partial^2}{\partial \beta_k^2} Q\left(\Lambda, \Lambda^g\right)_{\beta_k=\beta_k^g}} \tag{26}$$

where

$$\frac{\partial}{\partial \beta_k} Q\left(\Lambda, \Lambda^g\right)_{\beta_k=\beta_k^g} = e^{\beta_k^g} \sum_{i:j \in c(i), t(j)=k} \frac{-a_{ik} f_{ik} + \frac{b_{ik} h_{ik}}{e^{\beta_k^g}}}{b_{ik} + e^{\beta_k^g} a_{ik}} \tag{27}$$

and

$$\frac{\partial^2}{\partial \beta_k^2} Q\left(\Lambda, \Lambda^g\right)_{\beta_k=\beta_k^g} = e^{\beta_k^g} \begin{pmatrix} \frac{\partial}{\partial \beta_k} Q\left(\Lambda, \Lambda^g\right)_{\beta_k=\beta_k^g} + \\ \\ e^{\beta_k^g} \sum_{i:j \in c(i), t(j)=k} \frac{a_{ik}^2 f_{ik} - \frac{b_{ik}^2 h_{ik}}{e^{\beta_k^g 2}}}{\left(b_{ik} + e^{\beta_k^g} a_{ik}\right)^2} \end{pmatrix} \tag{28}$$

Thus, we will have the following update rule for our β_k parameter estimates:

$$\beta_k^{[t]} = \beta_k^{[t-1]} - \frac{\frac{\partial}{\partial \beta_k} Q\left(\Lambda, \Lambda^g\right)_{\beta_k=\beta_k^{[t-1]}}}{\frac{\partial^2}{\partial \beta_k^2} Q\left(\Lambda, \Lambda^g\right)_{\beta_k=\beta_k^{[t-1]}}} \tag{29}$$

4 Incorporating Negative Examples

While the above presentation of EM to learn parameters attempts to maximize the likelihood of training examples, doing so using only relevant components results in very poor parameter estimation. This is a direct result of the fact that optimizing the likelihood of relevant components may also increase the likelihood of components that are not relevant. In our own experiments, using only relevant components during training will result in most of the weight being placed in λ_D. We feel this may be a side effect of the bias-variance problem in estimation. The document language model has more bias than the language models estimated from the components, but the variance is lower as the sample sizes are larger for documents than for components. When combining the language models during smoothing, the document language models tend to have a higher likelihood of generating the query terms due to this lower variance.

In order to combat these effects, we also include negative examples in our training data. However, we do not wish to optimize the likelihood of the negative examples. We would prefer to maximize the likelihood that the language models estimated for the non-relevant components *do not* generate the query terms. To model this one might include for each non-relevant component and query term an example where we use $(1 - P(x\,|\theta_j\,))$ in place of $P(x\,|\theta_j\,)$. Note that this is not quite the same as what we one might wish to optimize, as:

$$1 - P(Q\,|\mu_i\,) \neq \prod_{l=1}^{|Q|} (1 - P(q_l\,|\mu_i\,)) \tag{30}$$

However, this is a useful and effective approximation that requires only the above substitution for negative examples. A complication in learning using the inclusion of negative examples given above is that $P(x\,|\mu_i\,)$ tends to be very small in relation to $1 - P(x\,|\mu_i\,)$. That means that when when maximizing the log likelihood, a small improvement of a positive example may outweigh a large degradation in performance in a negative example.

To accommodate for that effect, we weight the negative probabilities by raising them to a large power. For a negative example, we replace

$$P(x\,|\theta_j\,) \tag{31}$$

with

$$(1 - P(x\,|\theta_j\,))^{\nu\delta} \tag{32}$$

where ν is a user chosen parameter that specifies how much emphasis the negative examples have relative to the positive examples and δ is chosen so that the average probability of a term given the relevant examples is equal to the average probability of a term given the non-relevant examples when $\nu = 1$:

$$\delta = \frac{\log\left(\frac{1}{|positive|}\sum_{positive} P(x_i\,|\mu_i\,)\right)}{\log\left(\frac{1}{|negative|}\sum_{negative} P(x_i\,|\mu_i\,)\right)} \tag{33}$$

This approach for the incorporation of non-relevant components is ad-hoc but effective, as we will see in the next section.

5 Experimental Methodology

We use a locally modified version of the Indri search engine of the Lemur toolkit [6] that supports the hierarchical shrinkage. The hierarchical shrinkage support will be made available in a December release. Release of the parameter estimation code is scheduled for a later release as the estimation methods are still in flux. We indexed the INEX collection using the InQuery stopword list and the Krovetz stemmer. To process queries we removed all quotes from the query (thus ignoring phrasal constraints) and all terms with a minus in front.

We will focus on the CO.Thorough task and present results using the strict and generalized quantizations for nxCG[10], nxCG[25], nxCG[50], and MAP of ep/gr to facilitate comparison to the official results presented at INEX.

6 Experiments

In this section we present experiments on the CO.Thorough task. We will disregard our official submissions as they were run with the desired model and they were not run on the entire corpus. We had some problems with using the system that prevented us from indexing the entire corpus which have since been resolved.

We trained our parameters using the INEX 1.8 corpus and CO topics 162-201 using one non-relevant document component as a negative example for each relevant component as a positive example. Components were considered relevant if and only if they were highly exhaustive and highly specific. The non-relevant examples were taken from the same documents as the relevant examples. Ten iterations were used for the EM algorithm. α_k values were updated only for cases where there were at least ten examples for type k in the update rule.

Table 1 shows the a sample of the parameters the EM algorithm learned on the training topics. As ν increases, the weight on the collection language model (λ_C) decreases while the weight in the parent (λ_P) slightly increases and λ_O, the weight on the component and its children, noticeably increases.

With regards to the α parameters, the type specific length proportional weights on children, a few parameters start with relatively low values and increase rapidly as ν increases. Table 1 shows a few examples of this behavior. However, most parameters that are learned are very close to zero across all values of ν.

There seems to be some undesirable variation in the parameters, as we can see with the α value for the p type. This may be a side effect of the algorithm being trained on relatively few examples for some types, but this should not be the case for the p tag. However, as it only really matters what the value is relative to the other tags at the same level, perhaps this variance is not an issue.

Table 1. Some parameters learned from training data. As ν increases, λ_C decreases and λ_O increases. Some α parameters seem fairly stable, such as that of the footnote type. Others increase greatly with larger ν while some seem somewhat erratic (e.g. p).

	λ				α				
ν	(C)ol	(D)oc	(P)ar	O-self	st	p	sub	footnote	ss1
1.0	0.475	0.222	0.035	0.268	0.38	0.23	0.00	0.28	0.50
2.0	0.385	0.212	0.037	0.365	1.07	0.00	0.22	2.49	0.37
3.0	0.342	0.210	0.040	0.408	22.75	9.77	7.77	2.22	1.75
4.0	0.321	0.210	0.041	0.428	189.28	0.00	9.42	1.83	6.01
5.0	0.309	0.213	0.043	0.435	48623.30	0.61	146.46	1.65	13289.10

Table 2 shows the effects of using the learned parameters for the CO task on the training topics 162-201. Note that we use the new INEX-2.2 corpus, so these results are not directly comparable to previous results on these topics. As there are many documents that in the INEX-2.2 corpus that were not available for assessment for the topics, one should regard the evaluation numbers as a suboptimal estimate of performance. Nevertheless, we are mostly interested in the relative performance of the parameters learned for different values of ν, and the values in Table 2 should be adequate for that purpose.

In Table 2 we see that setting $\nu = 1$ yields the most consistently good results for both quantizations. There also seems to be some variation in the columns that does not follow a nice curve. This is an undesirable property which could be a result of variance in the learning algorithm, a sign of instability in the evaluation metrics, or a symptom of too few topics to get a reliable point estimate given the topic variance of the system.

Table 2. Results of varying the negative weight ν on the CO task using training topics 162-201. Values in bold font indicate the largest value for a measure.

	Strict				Generalized			
	nxCG			MAP	nxCG			MAP
ν	10	25	50	ep/gr	10	25	50	ep/gr
1.0	0.0704	0.0880	**0.1307**	**0.0034**	**0.2946**	**0.2950**	**0.2944**	**0.0852**
1.5	0.0593	**0.0906**	0.1266	0.0032	0.2938	0.2803	0.2710	0.0753
2.0	0.0593	0.0766	0.1237	0.0032	0.2899	0.2878	0.2816	0.0791
2.5	0.0704	0.0832	0.1226	0.0032	0.2922	0.2760	0.2637	0.0716
3.0	0.0704	0.0876	0.1210	0.0031	0.2911	0.2671	0.2536	0.0667
3.5	0.0704	0.0837	0.1218	0.0031	0.2920	0.2649	0.2490	0.0640
4.0	0.0593	0.0820	0.1197	0.0028	0.2903	0.2695	0.2447	0.0612
4.5	**0.0741**	0.0835	0.1219	0.0026	0.2857	0.2554	0.2383	0.0561
5.0	0.0630	0.0732	0.1087	0.0025	0.2791	0.2464	0.2256	0.0520

Table 3 shows the performance of the learned parameters on this year's CO.Thorough task. Performance for the generalized quantization peaks at $\nu = 2$ and around $\nu = 4$ for the strict quantization. This is quite a bit different from our observations on the training data. We would like to investigate this behavior in more detail. This could simply be the result of a training topic set that is too small or not representative enough. An alternative cause for difference is the change in the assessment methodology this year, which could result in assessors behaving giving different scores.

If we had submitted the system optimized to the training data ($\nu = 1$), then our results would have been in the top 10 official submissions for the strict quantization nxCG@50 metric and the generalized quantization MAP ep/gr metric. Supposing we had worked out our kinks in training (whether they be a result of the algorithm or the assessments) and we had selected the runs with $\nu = 2, 4$ for evaluation, then we would have had a run performing in the top 10 official submissions for the strict quantization nxCG@10,50 and MAP ep/gr metrics and for the generalized quantization nxCG@25,50 and MAP ep/gr metrics.

Table 3. Results of varying the negative weight ν on the CO.Thorough task using test topics 202-241. Values in bold font indicate the largest value for a measure.

	Strict				Generalized			
	nxCG			MAP	nxCG			MAP
ν	10	25	50	ep/gr	10	25	50	ep/gr
1.0	0.0200	0.0639	0.1051	0.0021	0.2225	0.2298	0.2286	0.0854
1.5	0.0440	0.0623	0.0911	0.0022	0.2207	0.2218	0.2197	0.0801
2.0	0.0440	0.0639	0.1006	0.0022	**0.2464**	**0.2421**	**0.2340**	**0.0882**
2.5	0.0440	0.0655	0.1127	0.0026	0.2200	0.2215	0.2224	0.0813
3.0	0.0440	0.0712	0.1184	0.0027	0.2164	0.2221	0.2167	0.0771
3.5	0.0400	0.0744	0.1192	0.0022	0.2131	0.2189	0.2149	0.0717
4.0	**0.0691**	**0.0747**	**0.1225**	0.0028	0.2445	0.2248	0.2172	0.0751
4.5	0.0651	0.0715	0.1131	**0.0029**	0.2301	0.2144	0.2126	0.0701
5.0	0.0651	0.0731	0.1116	**0.0029**	0.2326	0.2183	0.2089	0.0682

7 Conclusions

We have derived a Generalized Expectation Maximization algorithm to learn the parameters of a simple hierarchical language modeling system for the ranking and retrieval of XML components. We showed a way to effectively incorporate non-relevant components during training.

We investigated the interaction of the relative weight on the negative training examples ν and retrieval effectiveness on the CO.Thorough task. Experimental evidence suggests that the optimal ν parameter may depend on the quantization function used in evaluation. However, we have not done a full investigation of the choice of positive and negative examples during training. In training, we relied only on components that were highly exhaustive and highly specific. This assumption is essentially the assumption of the strict quantization function. We have not done experiments where we use components deemed relevant by the generalized quantization function. While we leave this to future work, we recognize this may change the optimal choice of ν for optimizing performance for measures using the generalized quantization function.

Our incorporation of negative examples is ad-hoc. As future work, we plan to simulate replication of negative examples rather than directly modifying the probabilities of the language models we are combining. This is a minor change to the algorithm and will not change the maximum likelihood derivation presented in Section 3, but it will be more technically sound than the current incorporation of negative evidence presented in Section 4. We would also like to consider the possibility of performing the negative evidence at the query level, rather than negating probabilities at the level of query terms.

For these experiments, we worked with a simplified hierarchical model. Our previous work [1][2][3] presented a hierarchical model where components were smoothed recursively up and down the tree for a document. We would like to adapt the training algorithm to model recursive smoothing and learn parameters with that optimize the likelihood under that condition.

Up to this point we have discussed only flat text queries. We would like to adapt this approach to work with structured queries to learn approaches to weight components of the query. For example, we may learn that satisfaction of a phrasal constraint should receive higher weight than a constraint on the document structure.

8 Acknowledgments

This research was sponsored by National Science Foundation (NSF) grant no. CCR-0122581. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implicit, of the NSF or the US government.

References

1. Ogilvie, P., Callan, J.: Language models and structured document retrieval. In: Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX). (2003)
2. Ogilvie, P., Callan, J.P.: Using language models for flat text queries in xml retrieval. In: Proc. of the Second Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX), Dagstuhl, Germany (2003)
3. Ogilvie, P., Callan, J.: Hierarchical language models for xml component retrieval. In: Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Springer-Verlag (2005) 224–237
4. Ogilvie, P., Callan, J.P.: Combining document representations for known-item search. In: Proc. of the 26th annual int. ACM SIGIR conf. on Research and development in informaion retrieval (SIGIR-03), New York, ACM Press (2003) 143–150
5. Kamps, J., de Rijke, M., Sigurbjörnsson, B.: Length normalization in xml retrieval. In: Proceedings of the Twenty-Seventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (2004) 80–87
6. http://lemurproject.org/: (The Lemur Toolkit for Language Modeling and Information Retrieval)

Probabilistic Retrieval, Component Fusion and Blind Feedback for XML Retrieval

Ray R. Larson

School of Information Management and Systems
University of California, Berkeley
Berkeley, California, USA, 94720-4600
ray@sims.berkeley.edu

Abstract. This paper describes the retrieval approaches used by UC Berkeley in our official submissions for the various Adhoc tasks. As in previous INEX evaluations, the main technique we are testing is the fusion of multiple probabilistic searches against different XML components using different probabilistic retrieval algorithms. In addition this year we began to use a different fusion/combination method from previous years. This year we also continued to use re-estimated Logistic Regression (LR) parameters for different components of the IEEE document collection, estimated using relevance judgements from the INEX 2003 evaluation. All of our runs were fully automatic with no manual editing or interactive submission of queries, and all used only the title element of the INEX topics.

1 Introduction

When analyzing the results of the 2004 INEX evaluation we discovered a number of interesting approaches to XML retrieval that we had not previously explored. In particular we were struck by the work of Mass and Mandelbrod[14] adjusting the weights of component-level search results using the weights of document-level matching for the same documents. This seemed to have a natural affinity for the fusion approaches that we had already tried[12]. We ran a large number of experiments using the INEX 2004 relevance data and various combinations of components and weights for our version of the "pivot" value. In addition, we participated this year in CLEF and the GeoCLEF evaluations, where we were able to analyze the differences in performance between our fusion approaches and the alternative version of the Berkeley Logistic regression algorithm that has been used there for a number of years (See [3]) The best performing of those approaches (according to the incomplete analysis using the new evaluation methods for INEX that we were able to do in the short period between the end of CLEF and the submission date for INEX) were used in this year's various INEX adhoc tasks with no modification. This is the first time that we have used blind feedback and the "TREC2" version of Logistic regression in addition to using the re-estimated parameters for the "TREC3" model based on the relevance judgements from INEX 2003. In addition, element and collection fusion are going to be used for the heterogeneous track (which are not being submitted until after this paper is submitted).

In this paper we will first discuss the algorithms and fusion operators used in our official INEX 2005 adhoc runs. Then we will look at how these algorithms and operators were used in the various submissions for the adhoc and heterogeneous tracks, and finally we will examine the results and discuss possible problems in implementation, and directions for future research.

2 The Retrieval Algorithms and Fusion Operators

This year we did not use the Okapi BM-25 algorithm in our official INEX adhoc runs. Instead we used a new approach to combining and weighting the elements using only Logistic regression-based algorithms for retrieval.

In the remainder of this section we will describe the Logistic Regression algorithms that were used for the evaluation as well as the blind relevance feedback method used in combination with the TREC2 algorithm. In addition we will discuss the methods used to combine the results of searches of different XML components in the collections. The algorithms and combination methods are implemented as part of the Cheshire II XML/SGML search engine [11, 12, 9] which also supports a number of other algorithms for distributed search and operators for merging result lists from ranked or Boolean sub-queries.

2.1 TREC3 Logistic Regression Algorithm

The basic form and variables of the *Logistic Regression* (LR) algorithm used was originally developed by Cooper, et al. [6]. It provided good full-text retrieval performance in the TREC ad hoc task and in TREC interactive tasks [8] and for distributed IR [9]. As originally formulated, the LR model of probabilistic IR attempts to estimate the probability of relevance for each document based on a set of statistics about a document collection and a set of queries in combination with a set of weighting coefficients for those statistics. The statistics to be used and the values of the coefficients are obtained from regression analysis of a sample of a collection (or similar test collection) for some set of queries where relevance and non-relevance has been determined. More formally, given a particular query and a particular document in a collection $P(R \mid Q, D)$ is calculated and the documents or components are presented to the user ranked in order of decreasing values of that probability. To avoid invalid probability values, the usual calculation of $P(R \mid Q, D)$ uses the "log odds" of relevance given a set of S statistics, s_i, derived from the query and database, such that:

$$\log O(R \mid Q, D) = b_0 + \sum_{i=1}^{S} b_i s_i \tag{1}$$

where b_0 is the intercept term and the b_i are the coefficients obtained from the regression analysis of the sample collection and relevance judgements. The final ranking is determined by the conversion of the log odds form to probabilities:

$$P(R \mid Q, D) = \frac{e^{\log O(R|Q,D)}}{1 + e^{\log O(R|Q,D)}} \tag{2}$$

Based on the structure of XML documents as a tree of XML elements, we define a "document component" as an XML subtree that may include zero or more subordinate XML elements or subtrees with text as the leaf nodes of the tree. For example, in the XML Document Type Definition (DTD) for the INEX test collection defines an article (marked by XML tag <*article*>) that contains front matter (<*fm*>), a body (<*bdy*>) and optional back matter (<*bm*>). The front matter (<*fm*>), in turn, can contain a header <*hdr*> and may include editor information (<*edinfo*>), author information (<*au*>), a title group (<*tig*>), abstract (<*abs*>) and other elements. A title group can contain elements including article title (<*atl*>) the page range for the article (<*pn*>), and these in turn may contain other elements, down to the level of individual formatted words or characters. Thus, a component might be defined using any of these tagged elements. However, *not all possible components are likely to be useful* in content-oriented retrieval (e.g., tags indicating that a word in the title should be in italic type, or the page number range) therefore we defined the retrievable components selectively, including document sections and paragraphs from the article body, and bibliography entries from the back matter (see Table 3).

Naturally, a full XML document may also be considered a "document component". As discussed below, the indexing and retrieval methods used in this research take into account a selected set of document components for generating the statistics used in the search process and for extraction of the parts of a document to be returned in response to a query. Because we are dealing with not only full documents, but also document components (such as sections and paragraphs or similar structures) derived from the documents, we will use C to represent document components in place of D. Therefore, the full equation describing the LR algorithm used in these experiments is:

$$\begin{aligned}
\log O(R \mid Q, C) = & \\
& b_0 + \left(b_1 \cdot \left(\frac{1}{|Q_c|}\sum_{j=1}^{|Q_c|} \log qtf_j\right)\right) \\
& + \left(b_2 \cdot \sqrt{|Q|}\right) \\
& + \left(b_3 \cdot \left(\frac{1}{|Q_c|}\sum_{j=1}^{|Q_c|} \log tf_j\right)\right) \\
& + \left(b_4 \cdot \sqrt{cl}\right) \\
& + \left(b_5 \cdot \left(\frac{1}{|Q_c|}\sum_{j=1}^{|Q_c|} \log \frac{N - n_{t_j}}{n_{t_j}}\right)\right)
\end{aligned} \tag{3}$$

$$+ (b_6 \cdot \log |Q_d|)$$

Where:

Q is a query containing terms T,
$|Q|$ is the total number of terms in Q,
$|Q_c|$ is the number of terms in Q that also occur in the document component,
tf_j is the frequency of the jth term in a specific document component,
qtf_j is the frequency of the jth term in Q,
n_{t_j} is the number of components (of a given type) containing the jth term,
cl is the document component length measured in bytes.
N is the number of components of a given type in the collection.
b_i are the coefficients obtained though the regression analysis.

This equation, used in estimating the probability of relevance in this research, is essentially the same as that used in [5]. The b_i coefficients in the original version of this algorithm were estimated using relevance judgements and statistics from the TREC/TIPSTER test collection. In INEX 2005 we did not use the original or “Base” version, but instead used a version where the coeffients for each of the major document components were estimated separately and combined through component fusion. The coefficients for the Base version were $b_0 = -3.70$, $b_1 = 1.269$, $b_2 = -0.310$, $b_3 = 0.679$, $b_4 = -0.0674$, $b_5 = 0.223$ and $b_6 = 2.01$. The re-estimated coefficients were derived from the Logistic regression analysis using the INEX 2003 relevance assessments. In fact, separate formulae were derived for each of the major components of the INEX XML document structure, providing a different formula for each major component of the collection. These formulae were used in all the TREC3 LR runs submitted for the INEX 2005 adhoc tasks, The components and coefficients for each of b_i in formula 4 are shown in table 1

Index	b_0	b_1	b_2	b_3	b_4	b_5	b_6
Base	-3.70	1.269	-0.310	0.679	-0.0674	0.223	2.01
topic	-7.758	5.670	-3.427	1.787	-0.030	1.952	5.880
topicshort	-6.364	2.739	-1.443	1.228	-0.020	1.280	3.837
abstract	-5.892	2.318	-1.364	0.860	-0.013	1.052	3.600
alltitles	-5.243	2.319	-1.361	1.415	-0.037	1.180	3.696
sec_words	-6.392	2.125	-1.648	1.106	-0.075	1.174	3.632
para_words	-8.632	1.258	-1.654	1.485	-0.084	1.143	4.004

Table 1. Re-Estimated Coefficients for The TREC3 Logistic Regression Model

2.2 TREC2 Logistic Regression Algorithm

We also implemented a version of the LR algorithm that has been used very successfully in Cross-Language IR by Berkeley researchers for a number of years[3].

This algorithm, originally developed by Cooper et al. [4] for TREC2 is:

$$\begin{aligned}
\log O(R|C,Q) = log\frac{p(R|C,Q)}{1-p(R|C,Q)} &= log\frac{p(R|C,Q)}{p(\overline{R}|C,Q)} \\
&= c_0 + c_1 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\frac{qtf_i}{ql+35} \\
&+ c_2 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\log\frac{tf_i}{cl+80} \\
&- c_3 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\log\frac{ctf_i}{N_t} \\
&+ c_4 * |Q_c|
\end{aligned}$$

where C denotes a document component and Q a query, R is a relevance variable,

$p(R|C,Q)$ is the probability that document component C is relevant to query Q,
$p(\overline{R}|C,Q)$ the probability that document component C is not relevant to query Q, which is 1.0 - $p(R|C,Q)$
$|Q_c|$ is the number of matching terms between a document component and a query,
qtf_i is the within-query frequency of the ith matching term,
tf_i is the within-document frequency of the ith matching term,
ctf_i is the occurrence frequency in a collection of the ith matching term,
ql is query length (i.e., number of terms in a query like $|Q|$ for non-feedback situations),
cl is component length (i.e., number of terms in a component), and
N_t is collection length (i.e., number of terms in a test collection).
c_k are the k coefficients obtained though the regression analysis.

If stopwords are removed from indexing, then ql, cl, and N_t are the query length, document length, and collection length, respectively, after removing stopwords. If the query terms are re-weighted (in feedback, for example), then qtf_i is no longer the original term frequency, but the new weight, and ql is the sum of the new weight values for the query terms. Note that, unlike the document and collection lengths, query length is the "optimized" relative frequency without first taking the log over the matching terms.

The coefficients were determined by fitting the logistic regression model specified in $\log O(R|C,Q)$ to TREC training data using a statistical software package. The coefficients, c_k, used for our official runs are the same as those described by Chen[1]. These were: $c_0 = -3.51$, $c_1 = 37.4$, $c_2 = 0.330$, $c_3 = 0.1937$ and $c_4 = 0.0929$. Further details on the TREC2 version of the Logistic Regression algorithm may be found in Cooper et al. [4].

2.3 Blind Relevance feedback

It is well known that blind (also called pseudo) relevance feedback can substantially improve retrieval effectiveness in tasks such as TREC and CLEF. (See for example the papers of the groups who participated in the Ad Hoc tasks in TREC-7 (Voorhees and Harman 1998)[17] and TREC-8 (Voorhees and Harman 1999)[18].)

Blind relevance feedback is typically performed in two stages. First, an initial search using the original queries is performed, after which a number of terms are selected from the top-ranked documents (which are presumed to be relevant). The selected terms are weighted and then merged with the initial query to formulate a new query. Finally the reweighted and expanded query is run against the same collection to produce a final ranked list of documents. It was a simple extension to adapt these document-level algorithms to document components for INEX.

The TREC2 algorithm has been been combined with a blind feedback method developed by Aitao Chen for cross-language retrieval in CLEF. Chen[2] present a technique for incorporating blind relevance feedback into the logistic regression-based document ranking framework. Several factors are important in using blind relevance feedback. These are: determining the number of top ranked documents that will be presumed relevant and from which new terms will be extracted, how to rank the selected terms and determining the number of terms that should be selected, how to assign weights to the selected terms. Many techniques have been used for deciding the number of terms to be selected, the number of top-ranked documents from which to extract terms, and ranking the terms. Harman [7] provides a survey of relevance feedback techniques that have been used.

Lacking comparable data from previous years, we adopted some rather arbitrary parameters for these options for INEX 2005. We used top 10 ranked components for the initial search of each component type, and enhanced and reweighted the query terms using term relevance weights derived from well-known Robertson and Sparck Jones[15] relevance weights, as described by Chen and Gey[3]. The top 10 terms that occurred in the (presumed) relevant top 10 documents, that were not already in the query were added for the feedback search.

2.4 Result Combination Operators

As we have reported previously, the Cheshire II system used in this evaluation provides a number of operators to combine the intermediate results of a search from different components or indexes. With these operators we have available an entire spectrum of combination methods ranging from strict Boolean operations to fuzzy Boolean and normalized score combinations for probabilistic and Boolean results. These operators are the means available for performing fusion operations between the results for different retrieval algorithms and the search results from different different components of a document. We will only describe one of these operators here, because it was the only type used in the evaluation reported in this paper.

The MERGE_CMBZ operator is based on the "CombMNZ" fusion algorithm developed by Shaw and Fox [16] and used by Lee [13]. In our version we take the normalized scores, but then further enhance scores for components appearing in both lists (doubling them) and penalize normalized scores appearing low in a single result list, while using the unmodified normalized score for higher ranking items in a single list.

A new addition for this year was a merge/reweighting operator based on the "Pivot" method described by Mass and Mandelbrod[14]. In our case the new probability of relevance for a component is a weighted combination of the initial estimate probability of relevance for the component and the probability of relevance for the entire article for the same query terms. Formally this is:

$$P(R \mid Q, C_{new}) = (X * P(R \mid Q, C_{comp})) + ((1 - X) * P(R \mid Q, C_{art})) \qquad (4)$$

Where X is a pivot value between 0 and 1, and $P(R \mid Q, C_{new})$, $P(R \mid Q, C_{comp})$ and $P(R \mid Q, C_{art})$ are the new weight, the original component weight, and article weight for a given query. Although we found that a pivot value of 0.54 was most effective for INEX04 data and measures, we adopted the "neutral" pivot value of 0.5 for all of our 2005 adhoc runs, given the uncertainties of how this approach would fare with the new metrics and tasks.

3 INEX 2005 Adhoc Approach

Our approach for the INEX 2005 adhoc tasks was a bit different from the methods used in previous INEX 2003 and INEX 2004 This section will describe the indexing process and indexes used, and also discuss the scripts used for search processing. The basic database was the expanded IEEE collection. We will summarize the indexing process and the indexes used in the adhoc tasks for reference in the discussion.

3.1 Indexing the INEX 2005 Database

All indexing in the Cheshire II system is controlled by an XML/SGML Configuration file which describes the database to be created. This configuration file is subsequently used in search processing to control the mapping of search command index names (or Z39.50 numeric attributes representing particular types of bibliographic data) to the physical index files used and also to associated component indexes with particular components and documents. This configuration file also includes the index-specific definitions for the Logistic Regression coefficients (when not defined, these default to the "Base" coefficients shown in Table 1).

Table 2 lists the document-level (/article) indexes created for the INEX database and the document elements from which the contents of those indexes were

Name	Description	Contents	Vector?
docno	Digital Object ID	//doi	No
pauthor	Author Names	//fm/au/snm //fm/au/fnm	No
title	Article Title	//fm/tig/atl	No

topic	Content Words	//fm/tig/atl
		//abs
		//bdy
		//bibl/bb/atl
		//app
			Yes

topicshort	Content Words 2	//fm/tig/atl
		//abs
		//kwd
		//st
			Yes

date	Date of Publication	//hdr2/yr	No
journal	Journal Title	//hdr1/ti	No
kwd	Article Keywords	//kwd	No
abstract	Article Abstract	//abs	Yes
author_seq	Author Seq.	//fm/au @sequence	No
bib_author_fnm	Bib Author Forename	//bb/au/fnm	No
bib_author_snm	Bib Author Surname	//bb/au/snm	No
fig	Figure Contents	//fig	No
ack	Acknowledgements	//ack	No
alltitles	All Title Elements	//atl, //st	Yes
affil	Author Affiliations	//fm/aff	No
fno	IEEE Article ID	//fno	No

Table 2. Cheshire Article-Level Indexes for INEX

extracted. These indexes (with the addition of the are the same as those used last year. The *abstract, alltitles, keywords, title, topic* and *topicshort* indexes support proximity indexes (i.e., term location), supporting phrase searching.

As noted above the Cheshire system permits parts of the document subtree to be treated as separate documents with their own separate indexes. Tables 3 & 4 describe the XML components created for INEX and the component-level indexes that were created for them.

Table 3 shows the components and the path used to define them. The COMP_SECTION component consists of each identified section (<sec> ... </sec>) in all of the documents, permitting each individual section of a article to be retrieved separately. Similarly, each of the COMP_BIB, COMP_PARAS, and COMP_FIG components, respectively, treat each bibliographic reference (<bb> ... </bb>), paragraph (with all of the alternative paragraph elements shown in Table 3), and figure (<fig> ... </fig>) as individual documents that can be retrieved separately from the entire document.

<table>
<tr><th>Name</th><th>Description</th><th>Contents</th></tr>
<tr><td>COMP_SECTION</td><td>Sections</td><td>//sec</td></tr>
<tr><td>COMP_BIB</td><td>Bib Entries</td><td>//bib/bibl/bb</td></tr>
<tr><td>COMP_PARAS</td><td>Paragraphs</td><td>//ilrj|//ip1|//ip2|
//ip3|//ip4|//ip5|
//item-none|//p|
//p1|//p2|//p3|
//tmath|//tf</td></tr>
<tr><td>COMP_FIG</td><td>Figures</td><td>//fig</td></tr>
<tr><td>COMP_VITAE</td><td>Vitae</td><td>//vt</td></tr>
</table>

Table 3. Cheshire Components for INEX

Component or Index Name	Description	Contents	Vector?
COMP_SECTION			
sec_title	Section Title	//sec/st	Yes
sec_words	Section Words	//sec	Yes
COMP_BIB			
bib_author	Bib. Author	//au	No
bib_title	Bib. Title	//atl	Yes
bib_date	Bib. Date	//pdt/yr	No
COMP_PARAS			
para_words	Paragraph Words	*†	Yes
COMP_FIG			
fig_caption	Figure Caption	//fgc	No
COMP_VITAE			
vitae_words	Words from Vitae	//vt	No

Table 4. Cheshire Component Indexes for INEX †Includes all subelements of paragraph elements.

Table 4 describes the XML component indexes created for the components described in Table 3. These indexes make individual sections (COMP_SECTION) of the INEX documents retrievable by their titles, or by any terms occurring in the section. These are also proximity indexes, so phrase searching is supported within the indexes. Bibliographic references in the articles (COMP_BIB) are made accessible by the author names, titles, and publication date of the individual bibliographic entry, with proximity searching supported for bibliography titles. Individual paragraphs (COMP_PARAS) are searchable by any of the terms in the paragraph, also with proximity searching. Individual figures (COMP_FIG) are indexed by their captions, and vitae (COMP_VITAE) are indexed by keywords within the text, with proximity support.

Almost all of these indexes and components were used during Berkeley's search evaluation runs of the 2005 INEX topics. The official submitted runs and scripts used in INEX are described in the next section.

3.2 INEX '04 Official Adhoc Runs

Berkeley submitted a total of 20 retrieval runs for the INEX 2005 adhoc tasks, these included 3 for each of the CO and CO+S Focussed and Thorough tasks, two each for CO and COS FetchBrowse tasks and one run each for the VVCAS, VSCAS, SVCAS and SSCAS tasks. This section briefly describes the individual runs and general approach taken in creating the queries submitted against the INEX database and the scripts used to prepare the search results for submission. The paragraphs below briefly describe Berkeley's INEX 2005 runs.

3.3 CO and CO+S Runs

Essentially the same basic component retrieval runs were used with different post-retrieval processing for the Thorough, Focussed, and FetchBrowse tasks. Our primary focus was on the Thorough task, since that was most similiar to our most effective runs from previous INEX evaluations. The three runs for each of the CO and CO+S Thorough and Focussed tasks were:

LRPIV: Runs containing this term used the TREC3 algorithm as described above for all retrieval ranking. The basic results were the combination of searches on each of the component types described in Table 3 using the TREC3 algorithm with component scores scaled using document level scores using the Pivot method described above with a pivot value of 0.5.

T2: Runs containing this term used the TREC2 algorithm in place of the TREC3 algorithm, but were otherwise the same.

T2FB: Runs containing this term used the TREC2 algorithm with Blind Feedback as described above, but otherwise were the same as "T2" runs.

The primary task that we focussed on was the CO.Thorough task. For this task some automatic expansion of items in the XPath to the root of the document was used. The same data was used for the COS.Thorough task, but post-processing restricted results to (approximately) those matching the structural constraints of the "castitle".

For the CO and COS Focussed tasks, post-processing kept only the highest ranking non-overlapping elements from the unexpanded version of results. As the very poor results for the Focussed runs show, this trimming of the results was overly harsh, and eliminated many of the relevant items in the initial set. (In fact, the results for the focussed tasks were so bad that we plan, time permitting, to do a complete analysis of where the post-retrieval processing caused them to fail, and to do test runs using the corrected post-processing for comparison).

The summary average MAnxCG@10 results for the runs described above are shown in Table 5.

Given the large number of runs (and problems getting evalj to successfully complete and produce gnuplot data), we are not including figures showing nxCG plots for the runs, but plan to do so for the final version of the paper.

Unlike our attempt at INEX 2004 to use a simple form of "blind feedback" that used only the kwd element of the documents, use of the TREC2 algorithm

Run Name	Task	MAnxCG@10	MAnxCG@10
		Q=gen	Q=strict
CO_PIV50_LRPIV_FOC	CO.Focussed	0.0581	0.0077
CO_PIV50_T2_FOC	CO.Focussed	0.0924	0.0213
CO_T2FB_PIV50_NOV	CO.Focussed	0.0885	0.0255
CO_PIV50_LRPIV_FOC_COS	COS.Focussed	0.0612	0.0077
CO_PIV50_T2_FOC_COS	COS.Focussed	0.0881	0.0213
COS_T2FB_PIV50_NOV	COS.Focussed	0.0884	0.0318
CO_PIV50_LRPIV_EXP_THR	CO.Thorough	0.2242	0.0225
CO_PIV50_T2_EXP_THR	CO.Thorough	0.2432	0.0375
CO_T2FB_PIV50_THR	CO.Thorough	0.2907	0.0602
CO_PIV50_LRPIV_COSTHR	COS.Thorough	0.2228	0.0399
CO_PIV50_T2_COSTHR	COS.Thorough	0.2393	0.0491
COS_T2FB_PIV50_ALL_EXP	COS.Thorough	0.2652	0.0551
LRPIV_SSCAS	SSCAS	0.2409	0.1362
LRPIV_SVCAS	SVCAS	0.2409	0.1362
LRPIV_VSCAS	VSCAS	0.1733	0.0927
LRPIV_VVCAS	VVCAS	0.1733	0.0927
CO_PIV50_LRPIV_FETCHBROWSE	CO.FetchBrowse	0*	0*
CO_PIV50_T2_FETCHBROWSE	CO.FetchBrowse	0*	0*
CO_PIV50_LRPIV_FETCHBROWSE_COS	COS.FetchBrowse	0*	0*
CO_PIV50_T2_FETCHBROWSE_COS	COS.FetchBrowse	0*	0*

Table 5. Berkeley Adhoc Runs, Tasks, and Results

and Blind Feedback with terms selected by relevance values, showed a consistent improvement over the TREC2 algorithm alone, or the TREC3 algorithm alone. This was the case for Berkeley runs in CLEF and it was pleasing to see that it was equally applicable in INEX. We hope to do further analysis to attempt to determine the optimal number of records to use in feedback and the optimal number of additional terms to include in the reformulated query).

3.4 CAS Runs

Our approach to the 4 CAS tasks was to run them almost identically to the method used in INEX 2004, with a few additional constraints on the structural matching criteria. Only a single run for each of CAS tasks was submitted, and all of the runs used just the TREC3 ranking algorithm. (Given the effectiveness shown by the T2FB for the CO tasks, we now wish that we had submitted runs using that combination for consideration as well). Overall, the Berkeley CAS runs performed about average among the other submissions, with the SVCAS and VSCAS runs faring best among our CAS submissions.

3.5 FetchBrowse Runs

All of our FetchBrowse runs were rejected due to a sorting problem that incorrectly interleaved a few entries from separate documents in one topic.

4 Conclusions and Future Directions

Considerable further analysis needs to be done to digest the vast number of variables, metrics and tasks introduced in this year's INEX and to make sense of their implications for future adaptations of our system. Overall, however, we have been pleased to discover that the TREC2 with Blind Feedback method seems to work consistently better for INEX tasks than the TREC3 algorithm alone. We hope to further examine these results and to conduct further experiments to see whether combinations (fusion) of these algorithms will be more or less effective than the base algorithms alone. We also plan to do a more detailed analysis of the post-processing steps used this year to discover if some of the poorer results were simply the result of post-processing errors.

References

1. A. Chen. Multilingual information retrieval using english and chinese queries. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum, CLEF-2001, Darmstadt, Germany, September 2001*, pages 44–58. Springer Computer Scinece Series LNCS 2406, 2002.
2. A. Chen. *Cross-Language Retrieval Experiments at CLEF 2002*, pages 28–48. Springer (LNCS #2785), 2003.
3. A. Chen and F. C. Gey. Multilingual information retrieval using machine translation, relevance feedback and decompounding. *Information Retrieval*, 7:149–182, 2004.
4. W. S. Cooper, A. Chen, and F. C. Gey. Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression. In *Text REtrieval Conference (TREC-2)*, pages 57–66, 1994.
5. W. S. Cooper, F. C. Gey, and A. Chen. Full text retrieval based on a probabilistic equation with coefficients fitted by logistic regression. In D. K. Harman, editor, *The Second Text Retrieval Conference (TREC-2) (NIST Special Publication 500-215)*, pages 57–66, Gaithersburg, MD, 1994. National Institute of Standards and Technology.
6. W. S. Cooper, F. C. Gey, and D. P. Dabney. Probabilistic retrieval based on staged logistic regression. In *15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, June 21-24*, pages 198–210, New York, 1992. ACM.
7. D. Harman. Relevance feedback and other query modification techniques. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 241–263. Prentice Hall, 1992.
8. R. R. Larson. TREC interactive with cheshire II. *Information Processing and Management*, 37:485–505, 2001.
9. R. R. Larson. A logistic regression approach to distributed IR. In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pages 399–400. ACM, 2002.
10. R. R. Larson. Cheshire II at INEX: Using a hybrid logistic regression and boolean model for XML retrieval. In *Proceedings of the First Annual Workshop of*

the Initiative for the Evaluation of XML retrieval (INEX), pages 18–25. DELOS workshop series, 2003.

11. R. R. Larson. Cheshire ii at inex '04: Fusion and feedback for the adhoc and heterogeneous tracks. In *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX2004*, pages 322–336. Springer (LNCS #3493), 2005.
12. R. R. Larson. A fusion approach to XML structured document retrieval. *Information Retrieval*, 8:601–629, 2005.
13. J. H. Lee. Analyses of multiple evidence combination. In *SIGIR '97: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 27-31, 1997, Philadelphia*, pages 267–276. ACM, 1997.
14. Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement for xml retrieval. In *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX2004*, pages 73–84. Springer (LNCS #3493), 2005.
15. S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, pages 129–146, May–June 1976.
16. J. A. Shaw and E. A. Fox. Combination of multiple searches. In *Proceedings of the 2nd Text REtrieval Conference (TREC-2), National Institute of Standards and Technology Special Publication 500-215*, pages 243–252, 1994.
17. E. Voorhees and D. Harman, editors. *The Seventh Text Retrieval Conference (TREC-7).* NIST, 1998.
18. E. Voorhees and D. Harman, editors. *The Eighth Text Retrieval Conference (TREC-8).* NIST, 1999.

GPX - Gardens Point XML IR at INEX 2005

Shlomo Geva

Centre for Information Technology Innovation
Faculty of Information Technology
Queensland University of Technology
Queensland 4001 Australia

s.geva@qut.edu.au

Abstract *The INEX 2005 evaluation consisted of numerous tasks that required different approaches. In this paper we described the approach that we adopted to satisfy the requirements of all the tasks, CAS and CO, in Thorough, Focused, and Fetch Browse mode, using the same underlying system. The retrieval approach is based on the construction of a collection sub-tree, consisting of all nodes that contain one or more of the search terms. Nodes are then assigned a score using a TF_IDF variant, and finally results are ranked. We present results that demonstrate that the approach is versatile and produces relatively good performance across all INEX 2005 tasks.*

Keywords XML Information Retrieval, XML Search Engine, Inverted Files, XML-IR, Focused retrieval.

1. XML File Inversion

In our scheme each term posting in the collection consists of three path elements: the file name, the absolute XPath context, and the ordinal position within the XPath context. The entire collection is inverted and the indexing structure supporting access to the terms inverted lists is stored in a MS Access database. Details of the GPX database structure can be found in the 2004 proceedings ("GPX - Gardens Point XML Information Retrieval at INEX 2004". The most significant difference in the indexing structure from 2004 was that in 2005 we did not index stop words. Stop words were defined as words that occurred more than 150,000 times in the collection.

2. Processing NEXI queries

Processing of complex NEXI expressions is based on parsing of the expression and the incremental construction of a result-tree. The result-tree consists of all the elements in the collection that contains at least one of the keyword in the query (or a synonym or any other term deemed relevant). Each node in the result tree contains the necessary information to allow the computation of a score, using a TF-IDF variant (described in section 3). After the result-tree is constructed, a traversal of the result-tree generates the score for each node, from the leaves to the root node. These results are then organized as a list and sorted by score, with the top N results returned (N=1500 for the ad-hoc track).

When a NEXI expression contains multiple filters the system constructs a result-tree for each of the filters. After the score of each node in all trees is determined, the scores of support elements (i.e. elements that satisfy a support filter in the NEXI expression) are used to boost the score of result elements. In this manner, elements with support tend to be ranked higher than elements without support, everything else being equal. More specific details can be found in the paper describing our submission to the ad hoc track, in these proceedings.

3. Ranking Scheme

Elements are ranked according to a relevance score. In our scheme leaf and branch elements need to be treated differently. Data usually occurs at leaf elements, and thus, our inverted list mostly stores information about leaf elements. A leaf element is considered candidate for retrieval if it contains at least one query term. A branch node is candidate for retrieval if it contains a relevant child element. Once an element (either leaf or branch) is deemed to be a candidate for retrieval its relevance score is calculated. A heuristically derived formula (Equation 1) is used to calculate the relevance score of leaf elements. The same equation is used for both return and support elements. The score is determined from query terms contained in the element. It penalizes elements with frequently occurring query terms (frequent in the collection), and it rewards elements with more unique query terms within a result element.

Equation 1: Calculation of a Leaf Element's Relevance Judgment Score

$$L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i} \tag{1}$$

Here n is the number of unique query terms contained within the leaf element, N is a small integer (we used K=5). The term K^{n-1} scales up the score of elements having multiple distinct query terms. The system is not sensitive to the value of K – we experimented with K=5 to 25 with little difference in results. The sum is over all terms where t_i is the frequency of the i^th^ query term in the leaf element and f_i is the frequency of the i^{th} query term in the collection. This sum rewards the repeat occurrence of query terms, but uncommon terms contribute more than common terms.

Once the relevance scores of leaf elements have been calculated, they can be used to calculate the relevance judgment score of branch elements. A naïve solution would be to just sum the relevance judgment score of each branch relevant children.

However, this would ultimately result in root (i.e. article) elements accumulating at the top of the ranked list, a scenario that offers no advantage over document-level retrieval. Therefore, the relevance score of children elements should be somehow decreased while being propagated up the XML tree. A heuristically derived formula (Equation 2) is used to calculate the scores of intermediate branch elements.

Equation 2: Calculation of a Branch Element's Relevance Judgment Score

$$R = D(n)\sum_{i=1}^{n} L_i \qquad \textbf{(2)}$$

Where:

n = the number of children elements
$D(n)$ = 0.49 if n = 1
0.99 Otherwise
L_i = the i$^{\text{th}}$ return child element

The value of the decay factor D depends on the number of relevant children that the branch has. If the branch has one relevant child then the decay constant is 0.49. A branch with only one relevant child will be ranked lower than its child. If the branch has multiple relevant children the decay factor is 0.99. A branch with many relevant children will be ranked higher than its descendants. Thus, a section with a single relevant paragraph would be judged less relevant than the paragraph itself, but a section with several relevant paragraphs will be ranked higher than any of the paragraphs.

Having computed scores for all result and support elements, the scores of support elements are added to the scores of the corresponding result elements that they support. For instance, consider the query:

//A[about(.//B,C)]//X[about(.//Y,Z)]

The score of a support element //**A//B** will be added to all result elements **//A//X//Y** where the element **A** is the ancestor of both **X** and **Y**.

Finally, the results consist of an entire recall tree for the query where each node is individually scored. The results are sorted by score and the top N results returned.

4. Treatment of CAS variants

The INEX ad-hoc track aimed to answer some questions with respect to the utility of the various filters of a NEXI expression. Four different sub-tasks were defined: ***VVCAS, VSCAS, SVCAS, SSCAS.*** The GPX search engine starts with construction of a collection sub-tree using inverted lists of term posting. The *SVCAS, VSCAS,* and

SSCAS variants require strict structural interpretation of a result filter, the support filter/s, or both, respectively. The *VVCAS* variant requires loose structural interpretation of all filters. To enforce structural constraints filtering of result or support elements is performed when the term posting lists are processed. Following this initial filtering all variants are processed identically. All CAS variants at INEX2005 were processed as Thorough runs meaning that all relevant elements were returned, ignoring overlap. This is supported naturally because the collection sub-tree that the search engine constructs contains the entire recall base – as identified by the system. Results are extracted, sorted, and the top N results returned.

4. Treatment of CO variants

The *CO* and *COS* tasks were designed to study queries having structural constraints, in comparison with and the same queries without such constraints. The CO and COS queries were used to generate result for three different user models – Thorough, Focused, and Fetch Browse. The Thorough retrieval requires the complete recall base (or N top ranked results). The Focused retrieval requires the return of elements of just the right granularity, and without overlap. The Fetch Browse retrieval requires the return of ranked documents and the ranked complete recall base within those documents (supporting document browsing in document rank order with identified relevant components).

The *CO* and *COS* queries only differ in the complexity of the NEXI expression, where a *CO* query can be expressed as a search over the entire article element. Therefore our system did not treat *CO* topic any differently to *COS* topics.

Thorough retrieval was performed as for the *CAS* tasks. Focused retrieval and Fetch Browse retrieval also started with the construction of the collection sub-tree, as done for Thorough retrieval. The Fetch Browse retrieval then proceeded to sort the result by file node score and then by element score within, as required for this task. For Fetch Browse retrieval the system extracted from each path – from article to leaf – the highest ranking node. In this manner the resulting set of candidate results contained no overlap. The results were then sorted by element score.

5. Results

An Implementation of High-Speed and High-Precision XML Information Retrieval System on Relational Databases

Kei Fujimoto[1], Toshiyuki Shimizu[1], Kenji Hatano[2], Yu Suzuki[3], Toshiyuki Amagasa[4], Hiroko Kinutani[5], and Masatoshi Yoshikawa[1]

[1] Graduate School of Information Science, Nagoya University,
Furocho, Chikusa, Nagoya, Aichi, Japan
{fujimoto, shimizu}@dl.itc.nagoya-u.ac.jp, yosikawa@is.nagoya-u.ac.jp

[2] Graduate School of Information Science, Nara Institute of Science and Technology,
8916-5 Takayama, Ikoma 630-0192, Nara, Japan
hatano@is.naist.jp

[3] College of Information Science and Technology, Ritsumeikan University,
1-1-1, Noji-Higashi, Kusatsu 525-8577, Shiga, Japan
suzuki@ics.ritsumei.ac.jp

[4] Graduate School of Systems and Information Engineering, University of Tsukuba,
1-1-1 Tennodai, Tsukuba 305-8573, Japan
amagasa@cs.tsukuba.ac.jp

[5] Information Media and Education Square, Ochanomizu University,
2-1-1, Ohtsuka, Bunkyo 112-8610, Tokyo, Japan
kinutani@edu.cc.ocha.ac.jp

Abstract. We are developing an XML infromation retrieval system by using XRel, an XML database system on relational databases. In XRel, XML documents are stored into four relational tables that have fixed relational schemas. Relational schemas are independent of the logical structure of XML documents. Therefore, changes in logical structure do not affect relational schemas. Each node in XML documents is represented by labels that express the positions in XML tree and paths from the root to the node.

In addition, XRel has token table to enable us to retrieve XML fragments based on their content vectors. Token table has tokens, XML fragment ID that the token belongs to, and the weights of the tokens as attributes. The weights of the tokens are calculated by taking element or path information into consideration. Acceleration of XML documents retrieval is achived by filtering out irrelevant tokens that do not affect the final top-k result.

We have developed Kikori, an application system of XML information retrieval. One of the main features of Kikori is that it enables us to retrieve XML fragments such as chapter, section and paragraph of articles in INEX document set. In Kikori, search results are displayed in a manner reflecting document structures and relevance.

The Dynamic Retrieval of XML Elements

Carolyn J. Crouch, Sudip Khanna, Poorva Potnis, Nagendra Doddapaneni

Department of Computer Science
University of Minnesota Duluth
Duluth, MN 55812
(218) 726-7607
ccrouch@d.umn.edu

Extended Abstract

Our goal when we began our work with INEX in 2002 was to assess the utility of Salton's vector space model in its extended form for XML retrieval. Familiarity with Smart and faith in its capabilities led us to believe that this approach was promising if particular problems such as flexible retrieval (i.e., retrieval of elements at the desired degree of granularity) and ranking issues could be resolved. During the past year, our research has centered on an approach for the dynamic retrieval of elements which we believe provides a viable solution to both these problems.

For those interested in the background and evolution of our system, discussions are available in terms of earlier workshop papers [1-3]. This paper focuses on our method of flexible retrieval, which is performed dynamically at retrieval time. It returns a rank-ordered list of elements to the user. An overview of results with respect to INEX 2005 tasks is presented, comparing dynamic element retrieval with retrieval against all all-element index of the collection as well as results produced by other INEX participants. We note in particular the exceptional results produced when our method is applied to the INEX Fetch-and-Browse task.

Our method of flexible retrieval is based on a single indexing of the collection at the paragraph level. (Collection statistics as required for *Lnu-ltu* term weighting are also used, but these are available from an examination of the collection as a whole and once calculated can be applied to any collection with similar characteristics.) It uses an extension of the basic vector space model proposed by Fox to represent the various components of the structured document. This extension allows the incorporation of objective identifiers such as author name and date of publication with content identifiers in the representation of the document. Similarity between extended vectors is calculated as a linear continuation of the similarities of the corresponding subvectors.

Flexible retrieval in this system takes place after an initial retrieval. Given a query and a paragraph indexing of the collection, we retrieve a rank-ordered list of paragraphs. Paragraphs that correlate highly with the query are used to identify documents of interest to the query (i.e., those containing potentially relevant elements). Once such a document is identified, a bottom-up representation of the document tree is generated. *Lnu* term weights are generated for the element vectors at each level in the tree. The element vectors are correlated with the *ltu*-weighted query vector used for the initial retrieval and a rank-ordered list of elements is produced.

Before formulating our 2005 experiments, we first experimented with 2004 data (where relevance judgments were available) by using both extended vector and body-only retrieval and comparing the results to those produced by corresponding retrievals against an all-element indexing of the collection. Subvector weighting was also examined. Experiments on the 2004 INEX data set indicated that extended vector retrieval substantially outperformed body-only retrieval, so our 2005 results are based on extended vector retrieval using the same subvector weights. Although we participated in a number of the 2005 tasks, our interest centers on CO processing (with respect to both the Thorough and Focused tasks) and Fetch-and-Browse. Another interest is how flexible retrieval performs with respect to what may reasonably be considered an upper bound on performance, i.e., retrieval against the all element index. 2005 results show that flexible retrieval competes successfully in this respect.

In INEX 2005 we are using, for the first time, a system which retrieves elements dynamically in an effective manner and returns a rank-ordered list of elements to the user. Experimental results demonstrate the successful utilization of this approach for structured retrieval.

TopX & XXL at INEX 2005

Martin Theobald, Ralf Schenkel, and Gerhard Weikum

Max-Planck Institute für Informatik, Saarbrücken, Germany
{mtb,schenkel,weikum}@mpi-inf.mpg.de

Abstract. We participated with two different and independent search engines in this year's INEX round: The XXL Search Engine and the TopX engine. As this is the first participation for TopX, this paper focuses on the design principles, scoring, query evaluation and results of TopX. We shortly discuss the results with XXL afterwards.

1 TopX – System overview

Our query processing methods are based on precomputed index lists that are sorted in descending order of appropriately defined scores for individual tag-term content conditions, and our algorithmic rationale for top-k queries follows that of the family of *threshold algorithms (TA)* [2, 4, 5]. In order to find the top-k matches for multidimensional queries (e.g., with multiple content and structure conditions), scoring, and ranking them, TopX scans all relevant index lists in an interleaved manner. In each scan step, when the engine sees the score for a data item in one list, it combines this score with scores for the same data item previously seen in other index lists into a *global score* using a monotonic aggregation function such as weighted summation. We perform in-memory structural joins for content-and-structure (CAS) queries using pre-/postorder labels between whole element blocks for each query condition grouped by their document ids.

1.1 Top-k Query Processing for Semistructured Data

The query processor decomposes the query into *content conditions*, each of which refers to exactly one tag-term pair, and into additional elementary tag conditions (e.g., for navigation of branching path queries), plus the path conditions that constrain the way how the matches for the tag-term pairs and elementary tag conditions have to be connected. For NEXI we concentrate on content conditions that refer to the child-or-descendant axis, i.e., the full-text contents of elements. This way, each term is connected to its last preceding tag in the location path, in order to merge each tag-term pair into a single query condition with a corresponding list in the precomputed inverted index. Note that sequential reads are performed for these content-related tag-term-pairs, only, whereas additional structural query conditions for element paths or branching path queries are performed through a few judiciously scheduled random lookups on a separate, more compact element table.

The rationale for these distinctions is that random accesses are often one or two orders of magnitude more expensive than sorted accesses. Note that one index list (e.g., for a single term) on a large data collection may be very long, in the order of megabytes (i.e., multiple disk tracks), and the total index size may easily exceed a terabyte so that only the "hottest" fragments (i.e., prefixes of frequently needed lists) can be kept in memory. Sorted access benefits from sequential disk I/O with asynchronous prefetching and high locality in the processor's cache hierarchy; so it has much lower amortized costs than random access. Threshold algorithms with eager random accesses look up the scores for a data item in all query-relevant index lists, when they first see the data item in one list. Thus, they can immediately compute the global score of the item, and need to keep only the current top-k items with their scores in memory. Algorithms with a focus on sorted access do not eagerly look up all candidates' global scores and therefore need to maintain a candidate pool in memory, where each candidate is a partially evaluated data item d that has been seen in at least one list and may qualify for the final top-k result based on the following information (we denote the score of data item d in the i-th index list by $s(t_i, d)$, and we assume for simplicity that the score aggregation is summation):

- the set $E(d)$ of evaluated lists where d has already been seen,
- the $worstscore(d) := \sum_{i \in E(d)} s(t_i, d)$ based on the known scores $s(t_i, d)$, and
- the $bestscore(d) := worstscore(d) + \sum_{i \notin E(d)} high_i$ that d could possibly still achieve based on $worstscore(d)$ and the upper bounds $high_i$ for the scores in the yet unvisited parts of the index lists.

The algorithm terminates when the $worstscore(d)$ of the rank-k in the current top-k result, coined *min-k*, is at least as high as the highest $bestscore(d)$ among all remaining candidates.

1.2 Periodic Queue Maintenance and Early Candidate Pruning

All intermediate candidates that are of potential relevance for the final top-k results are collected in a hash structure (the *cache*) in main memory; this data structure has the full information about elements, *worstscores*, *bestscores*, etc. In addition, two priority queues merely containing pointers to these cache entries are maintained in memory and periodically updated. The top-k queue uses *worstscores* as priorities to organize the current top-k documents, and the candidate queue uses *bestscores* as priorities to maintain the stopping condition for threshold termination.

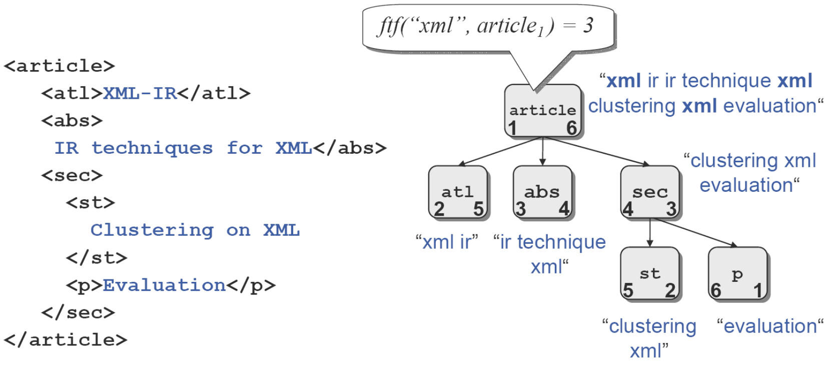
Results from [11] show that only a small fraction of the top candidates actually has to be kept in the candidate queue to provide a proper threshold for algorithm termination. Since TopX typically stops before having scanned all the relevant index lists completely, much less candidates than the ones that occur in the inverted lists for a query have to be kept in the cache. Both queues contain disjoint subsets of items currently in the cache. If an item's $bestscore(d)$ drops below the current *min-k* threshold, it is dropped from the candidate queue as well as from the cache. The queue is implemented using a Fibonacci heap, with efficient amortized lookups and maintenance.

Optionally, TopX also support various tunable probabilistic extensions to schedule random accesses for testing both content-related and structural query conditions as well as a probabilistic form of candidate pruning, thus yielding approximate top-k results with great run time gains compared to the conservative top-k baseline and probabilistic guarantees for the result quality [11]. However, for the current INEX experiments these probabilistic extensions were not employed, because here the focus is clearly on retrieval robustness rather than cutting edge performance.

2 Data & Scoring Model

2.1 Full-Content Indexing

We consider a simplified XML data model, where idref/XLink/XPointer links are disregarded. Thus every document forms a tree of nodes, each with a *tag* and a related *content*. We treat attributes nodes as children of the corresponding element node. The content of a node is either a text string or it is empty; typically (but not necessarily) non-leaf nodes have empty content. With each node, we can additionally associate its *full-content* which is defined as the concatenation of the contents of all the node's descendants. Optionally, we may apply standard IR techniques such as stemming and stop word removal to those text contents.



```
<article>
   <atl>XML-IR</atl>
   <abs>
    IR techniques for XML</abs>
   <sec>
     <st>
       Clustering on XML
     </st>
     <p>Evaluation</p>
   </sec>
</article>
```

ftf("xml", $article_1$) = 3
article 1 6
"xml ir ir technique xml clustering xml evaluation"
atl 2 5
abs 3 4
sec 4 3
"clustering xml evaluation"
"xml ir"
"ir technique xml"
st 5 2
p 6 1
"clustering xml"
"evaluation"

Fig. 1. Redundant full-text contents for elements.

This way, we conceptually treat each element as an eligible retrieval unit (i.e., in the classic IR notion of a document) with its expanded full-content text nodes as content, with no benchmark-specific tuning or preselection of commonly retrieved tags or the use of predefined retrieval units being necessary. In the following we focus on the child-or-descendant axis (i.e., the full-content case) as the much more important case for XML IR with vague search, thus following the NEXI specification; the case for the child-axis follows analogously. Figure 1 shows the full-content term frequency (*ftf*) of the term `xml` for a fictitious article element having a value of 3. So the whole article element is definitely relevant for a query containing the term xml, however it might be less compact than a more specific section or paragraph which should be taken into account in the scoring model.

2.2 Content Scores

TopX provides the option to evaluate queries either in conjunctive mode or in "andish" mode. In the first case, all terms and structural conditions must be met by a result candidate, but still different matches yield different scores. In the second case, a node matches a content condition of the form //"t_1 $t_2\ldots$" if its content contains at least one occurrence of at least one of the terms t_1, t_2, etc. It matches the full-content condition .//t_1 t_2 ..." if its full-content contains at least one occurrence of at least one of the search terms. In the first case, the significance (e.g., derived from frequencies and element-specific corpus statistics) of a matched term influences the score and the final ranking, but – similarly to boolean XPath – documents (or subtrees) that do not contain a specified term at all or that do not strictly match all structural query conditions are dismissed.

For content scores we make use of element-specific statistics that view the content or full-content of each element node n with tag A as a bag of words:

1) the *term frequency*, $tf(t, n)$, of term t in node n, which is the number of occurrences of t in the content of n;
2) the *full-content term frequency*, $ftf(t, n)$, of term t in node n, which is the number of occurrences of t in the full-content of n;
3) the *tag frequency*, N_A, of tag A, which is the number of nodes with tag A in the entire corpus;
4) the *element frequency*, $ef_A(t)$, of term t with regard to tag A, which is the number of nodes with tag A that contain t in their full-contents in the entire corpus.

Now consider a content condition of the form A//"$t_1 \ldots t_m$", where A is a tag name and t_1 through t_m are terms that should occur in the full-contents of a subtree. Our scoring of node n with regard to condition A//"t_1 $\ldots t_m$" uses formulas of the following type:

$$score(n, A//"t_1 \ldots t_m") := \frac{\sum_{i=1}^{m} relevance_i \cdot specificity_i}{compactness(n)},$$

where $relevance_i$ reflects ftf values, $specificity_i$ is derived from N_A and $ef_A(t_i)$ values, and $compactness(n)$ considers the subtree or element size for length normalization. Note that specificity is made XML-specific by considering combined tag-term frequency statistics rather than global term statistics only. It serves to assign different weights to the individual tag-term pairs which is a common technique from probabilistic IR.

Tag	N	Avg(dl)	k_1	b
`article`	16,808	2,903	10.5	0.75
`sec`	96,481	413	10.5	0.75
`p`	1,022,679	32	10.5	0.75
`fig`	109,230	13	10.5	0.75

Table 1. Element-specific parameterization of the extended BM25 model.

An important lesson from text IR is that the influence of the term frequency and element frequency values should be sublinearly dampened to avoid a bias for short elements with a high term frequency of a few rare terms. Likewise, the instantiation of compactness in the above formula should also use a dampened form of element size. Highly skewed score distributions would be beneficial for candidate pruning (and fast algorithm termination), but typically at a high expense in retrieval quality. To address these considerations, we have adopted the popular and empirically usually much superior Okapi BM25 scoring model (originating in probabilistic IR for text documents [6]) to our XML setting, leading to the following scoring function:

$$sore(n, A//"t_1 \ldots t_m") :=$$

$$\sum_{i=1}^{m} \frac{(k_1+1) \cdot ftf(t_i, n)}{K + ftf(t_i, n)} \cdot \log\left(\frac{N_A - ef_A(t_i) + 0.5}{ef_A(t_i) + 0.5}\right)$$

with

$$K = k_1 \left((1-b) + b \frac{length(n)}{avg\{length(n') \mid n' \; with \; tag \; A\}}\right).$$

The BM25 formula provides a dampened influence of the ftf and ef parts, as well as a compactness normalization that takes the average compactness of each element type into account. A simple hill-climbing-style parameter optimization using the 2004 INEX collection and relevance assessments yields a maximum in the MAP value for k_1 being set to 10.5, whereas the b parameter is confirmed to perform best at the default value of 0.75 provided in the literature. With regard to individual (element-specific) retrieval robustness, the above formula would also allow for a more elaborated parameter optimization for individual element types which was not considered for the current setup.

2.3 Structural Scores

For efficient testing of structural conditions we transitively expand all structural query dependencies. For example, in the query `//A//B//C[.// "t"]` an element with tag `C` (and content term `"t"`) has to be a descendant of both `A` and `B` elements. Branching path expressions can be expressed analogously. This way, the query forms a *directed acyclic graph* (DAG) with tag-term conditions as leafs, elementary tag conditions as interconnecting nodes between elements of a CAS query, and all transitively expanded descendant relations as edges. This transitive expansion of structural constraints is a key for efficient path validation and allows an *incremental testing* of path satisfiability. If `C` in the above example is not a valid descendant of `A`, we may safely prune the candidate document from the priority queue, if its $bestscore(d)$ falls below the current *min-k* threshold without ever looking up the `B` condition.

In non-conjunctive (aka. “andish”) retrieval, a result document (or subtree) should still satisfy most structural constraints, but we may tolerate that some tag names or path conditions are not matched. This is useful when queries are posed without much information about the possible and typical tags and paths or for vague content and structure (VCAS) search, where the structural constraints merely provide a hint on how the actual text contents should be connected. Our scoring model essentially counts the number of structural conditions (or connected tags) that are still to be satisfied by a result candidate d and assigns a small and constant score mass c for every condition that is matched. This structural score mass is combined with the content scores and aggregated with each candidate's $[worstscore(d),\ bestscore(d)]$ interval. In our setup we have set $c = 1$, whereas content scores were normalized to $[0, 1]$, i.e., we emphasize the structural query conditions. Note that it is still important to identify non-satisfiable structural conditions as early and efficiently as possible, because this can reduce the $bestscore(d)$ of a result candidate and make it eligible for pruning.

The overall score of a document or subtree for a content-and-structure (CAS) query is the sum of its content and structural scores. For content-only (CO) queries, i.e., mere keyword queries, the document score is the sum, over all terms, of the maximum per-term element scores within the same target element. If TopX is configured to return entire documents as query results (e.g., for the CO/S-Fetch&Browse task), the score of a document is the maximal score of any subgraph matching a target element in the document; if otherwise the result granularity is set to elements, we may obtain multiple results according to the differently scored target elements in a document. The internal TopX query processor completely abstracts from the original query syntax (NEXI or XPath) and uses a full-fletched graph traversal to evaluate arbitrary query DAGs. Furthermore, the top-k-style nature of the engine does not require candidates to be fully evaluated at all query conditions, but merely relies on $[worstscore(d),$ $bestscore(d)]$ bounds to determine the current top-k results and the *min-k* threshold for algorithm termination.

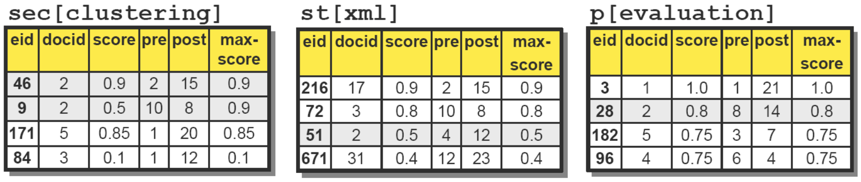
3 Database Schema & Indexing

3.1 Schema

Inverted index lists are stored as database tables; Figure 2 shows the corresponding schema definitions with some example data for three tag-term pairs. The current implementation uses Oracle 10g as a backbone, mainly for easy maintenance of the required index structures, whereas the actual query processing takes place outside the database exclusively in the TopX query engine, such that the DBMS itself remains easily exchangeable. Nodes in XML documents are identified by the combination of document id (`did`) and preorder (`pre`). Navigation along all XPath axes is supported by both the `pre` and `post` attributes using the XPath accelerator technique of [3]. Additionally, the `level` information may stored to support the child-axis as well, but may be omitted for the NEXI-style child-or-descendant constraints. The actual index lists are processed by the top-k algorithm using two B$^+$-tree indexes that are created on this base table: one index for sorted access support in descending order of the (maxscore, did) attributes for each tag-term pair and another index for random access support using (did, tag, term) as key.

3.2 Inverted Block-Index

The base table contains the actual node contents indexed as one row per tag-term pair per document, together with their local scores (referring either to the simple content or the full-content scores) and their pre- and postorder numbers. For each tag-term pair, we also provide the *maximum score* among all the rows grouped by tag, term, and document id to extend the previous notion of single-line sorted accesses to a notion of *sorted block-scans*. TopX scans each list corresponding to the key (tag, term) in descending order of (maxscore, did, score). Each sequential block scan prefetches all tag-term pairs for the same document id in one shot and keeps them in memory for further processing which we refer to as *sorted block-scans*. Random accesses to content-related scores for a given document, tag, and term are performed through small range scans on the respective B$^+$ tree index using the triplet (did, tag, term) as key. Note that grouping tag-term pairs by their document ids keeps the range of the pre-/postorder-based in-memory structural joins small and efficient. All scores in the database tables are precomputed when the index tables are built.



sec[clustering]

eid	docid	score	pre	post	max-score
46	2	0.9	2	15	0.9
9	2	0.5	10	8	0.9
171	5	0.85	1	20	0.85
84	3	0.1	1	12	0.1

st[xml]

eid	docid	score	pre	post	max-score
216	17	0.9	2	15	0.9
72	3	0.8	10	8	0.8
51	2	0.5	4	12	0.5
671	31	0.4	12	23	0.4

p[evaluation]

eid	docid	score	pre	post	max-score
3	1	1.0	1	21	1.0
28	2	0.8	8	14	0.8
182	5	0.75	3	7	0.75
96	4	0.75	6	4	0.75

Fig. 2. Inverted block-index with precomputed full-content scores over tag-term pairs.

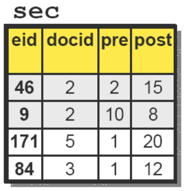
For search conditions of the form `A[.//"`t_1 t_2`"]` using the child-or-descendants axis, we refer to the full-contents scores, based on $ftf(t_1, A)$ and $ftf(t_2, A)$ values of entire document subtrees; these are read off the precomputed base tables in a single efficient sequential disk fetch for each document until the *min-k* threshold condition is reached and the algorithm terminates. We fully precompute and materialize this inverted block index to efficiently support the child-or-descendant axis. With this specialized setup, parsing and indexing times for the INEX collection are about 80 minutes on an average server machine including the modified BM25 scoring model and the materialization of the inverted block-index view.

We propagate, for every term t that occurs in a node n with tag A, its local tf value "upwards" to all ancestors of n and compute the ftf values of these nodes for t. Obviously, this may create a redundancy factor that can be as high as the length of the path from n to the root. The redundant full-content indexing introduces a factor of redundancy for the textual contents that approximately corresponds to the average nesting depth of text nodes of documents in the corpus; it is our intention to trade off a moderate increase in inexpensive disk space (factor of 4-5 for INEX) for faster query response times. Note that by using tag-term pairs for the inverted index lookups, we immediately benefit from more selective, combined tag-term features and shorter index lists for the actual textual contents, whereas the hypothetical combinatorial bound of $\#tags \cdot \#terms$ rows is by far not reached.

3.3 Navigational Index

To efficiently process more complex queries, where not all content-related query conditions can be directly connected to a single preceding tag, we need an additional element-only directory to test the structural matches for tag sequences or branching path queries.

Lookups to this additional, more compact and non-redundant navigational index yield the basis for the structural scores that a candidate may achieve for each matched tag-only condition in addition to the BM25-based content scores. As an illustration of the query processing, consider the example twig query `//A[.//B[.//"b"] and .//C[.//"c"]]`. A candidate that contains valid matches for the two extracted tag-term pairs `B:b` and `C:c` fetched through a series of block-scans on the inverted lists for `B:b` and `C:c`, may only obtain an additional static score mass c, if there is a common `A` ancestor that satisfies both the content-related conditions based on their already known pre-/postorder labels. Since all structural conditions are defined to yield this static score mass c, the navigational index is exclusively accessed through random lookups by an additional B$^+$ tree on this table. [10] provides different approaches to judiciously schedule these random accesses for the most promising candidates according to their already known content-related scores.



sec

eid	docid	pre	post
46	2	2	15
9	2	10	8
171	5	1	20
84	3	1	12

Fig. 3. Navigational index for branching path queries.

3.4 Random Access Scheduling

The rationale of TopX is to postpone expensive random accesses as much as possible and perform them only for the best top-k candidates. However, it can be beneficial to test path conditions earlier, namely, in order to eliminate candidates that might not satisfy the structural query conditions but have high *worstscores* from their textual contents. Moreover, in the query model where a violated path condition leads to a score penalty, positively testing a path condition increases the $worstscore(d)$ of a candidate, thus potentially improving the *min-k* threshold and leading to increased pruning subsequently. In TopX we consider random accesses at specific points only, namely, whenever the priority queue is rebuilt. At this point, we consider each candidate and decide whether we should make random accesses to test unresolved path conditions, or look up missing scores for content conditions. For this scheduling decision, we have developed two different strategies.

The first strategy, coined *MinProbe*, aims at a minimum number of random accesses by probing structural conditions for the most promising candidates, only. Since we do not perform any sorted scans for elementary tag conditions, we treat structural conditions as *expensive predicates* in the sense of [1]. We schedule random accesses only for those candidates d whose $worstscore(d)+o_j\cdot c > min\text{-}k$, where o_j is the number of untested structural query conditions for d and c is a static score mass that d earns with every satisfied structural condition.

This way, we schedule a whole batch of random lookups, if d has a sufficiently high $worstscore(d)$ to get promoted to the top-k when the structural conditions can be satisfied as well. If otherwise $bestscore(d)$ already drops below the current *min-k* threshold after a random lookup, we may safely prune the candidate from the queue. More sophisticated approaches may employ an analytic cost model, coined the *BenProbe* strategy in [10], in order to determine whether it is cost beneficial to explicitly lookup a candidate's remaining score in the structural and content-related query conditions.

4 Expensive Text Predicates

The use of auxiliary query hints in the form of expensive text predicates such as phrases (“”), mandatory terms (+), and negation (-) can significantly improve the retrieval results of an IR system. The challenge for a top-k based query processor lies in the *efficient* implementation of these additional query constraints and their adaptation into the sorted vs. random access scheduling paradigm.

4.1 Negation

The semantics of negations in a non-conjunctive, i.e. “andish”, query processor is not quite trivial. To cite the authors of the NEXI specification, “a user would be surprised if she encountered the negated term among the retrieval results”. This leaves some space for interpretation and most commonly leads to the conclusion that the negated term should not occur in any of the top-ranked results; yet we do not want to eliminate all elements containing one of the negated terms completely, if they also contain good matches to other content-related query conditions, and we would run into the danger of loosing substantial amount of recall. Therefore the scoring of negated terms is defined to be independent of the term's actual content score. Similarly to the structural query constraints introduced in the previous section, an element merely accumulates some additional static score mass if it does not match the negated term. This quickly leads us back to the notion of expensive predicates and the minimal probing approach. A random lookup onto this element's tag-term offsets is scheduled, if the document gets promoted into the top-k results after a successful negation test, i.e., if it does not contain the negated term among its full-content text nodes and obtains the static score for the unmatched negation. In the current setup, this static score mass was set to the same value $c = 1$ that was provided for structural query constraints.

4.2 Mandatory Terms

In contrast to term negations, the scores for mandatory query terms should still reflect the relevance of the term for a given element, i.e., our precomputed BM25-based content scores. Yet a too strict boolean interpretation of the +-operator would make us run into the danger of loosing recall at the lower ranks. We therefore introduce boosting factors and a slightly modified score aggregation of the form $score(n, A//"t_1 \ldots t_m") = \sum_{i=1}^{m} \beta_i + s(t_i, A)$, where $s(t_i, A)$ is the original content score, and β_i is set to 1 if the term is marked as mandatory (+) and 0 otherwise. Note that these β_i are constants at query evaluation time, and since the modified scores are taken into account for both the $worstscore(d)$ and $bestscore(d)$ bounds of all candidates, the boosting factors "naturally" enforce deeper sequential scans on the inverted index lists for the mandatory query conditions, typically until the final top-ranked results are discovered in those lists. Still weak matches for the remaining non-boosted query conditions may be compensated by a result candidate through high-scored matches in the mandatory query conditions.

4.3 Phrases & Phrase Negations

For phrase matching we store all term offsets in an auxiliary database table together with the pre-/postorder labels of each term's occurrence in a document. Again, phrases are interpreted as expensive predicates and tested by random accesses to the offset table using the minimal probing approach already described for the MinProbe scheduling. The only difference now is to determine whether a candidate element may aggregate the content-related score mass for the phrase-related conditions into it's overall $worstscore(d)$ that is then used to determine its position in the top-k results. In order to keep these score aggregations monotonous in the precomputed content scores, phrase lookups are treated as binary filters, only. Similarly to the single-term negations, phrase negations are defined to yield a static score mass c for each candidate element that does not contain the negated phrase. Single-term occurrences of the negated phrase terms are allowed, though, and do not contribute to the final element score unless they are also contained in the remaining query.

5 Experimental Results for TopX

5.1 CO-Thorough

For the CO-Thorough task, TopX ranks at position 22 for the nxCG@10 metric using a strict quantization with a value of 0.0379 and only at rank 37 of 55 submitted runs for MAP with a value of just 0.0008. As for all runs, we used the modified BM25 scoring model described above and also expensive text predicates to leverage phrases, negations, and mandatory terms. The very modest rank in the sensitive CO task attests that there is still some space for optimizations in our scoring model left for CO queries, when there is no explicit target element specified by the query ("//*"). Yet there was neither any restriction given on the result overlaps or granularities nor on the expected specificity or exhaustiveness of special element types such as sections or paragraphs, such that the engine was allowed to return any type of element (also list items or even whole articles) according to their aggregated content scores. An additional simple postprocessing step based on the element granularities and overlap removal would already be expected to achieve great performance gains here. However, for the old precision/recall metrics using INEX-eval with a strict quantization (INEX '04), the TopX run ranks at a significantly better position of rank 3 with an average precision of 0.058 (MAP), which actually corresponds to the particular metric and setup for which we had been tuning the system.

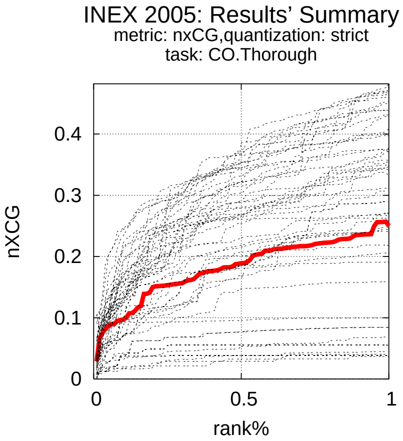


Fig. 4. nxCG results for the TopX CO-Thorough run

5.2 COS-Fetch&Browse

The situation improves for the COS-Fetch&Browse task where the TopX run ranks at position 4 out of 19 with a value of 0.0601 in the ep-gr metric with strict quantization. TopX was configured to first rank the result documents according to their highest-ranked target element and then return all target elements within the same result document with the same score according to a strict interpretation of the target element given by the query which exactly matches our full-content scoring model. Here the strict – XPath-like – interpretation of the query target element in combination with our full-content scoring model that treats each target element itself as a mini-document shows its benefits and naturally avoids overlap, since we return exactly the element type that is specified in the query and therefore seem to match the result granularity expected by a human user better.

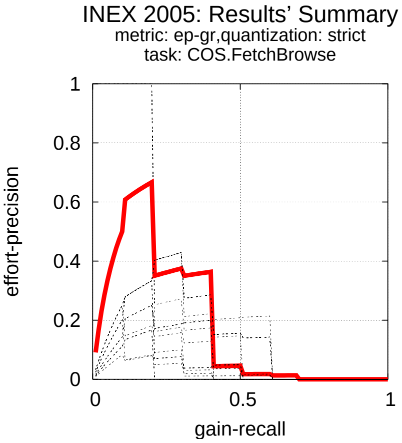


Fig. 5. ep-gr results for the TopX COS-Fetch&Browse run

5.3 SSCAS

Finally, the SSCAS task perfectly matches our strict interpretation of the target element with the precomputed full-content scores and no overlap allowed. The two submitted TopX runs rank at position 1 and 2 out of 25 submitted runs for the strict nxCG@10 metric with a value of 0.45 for both runs and still rank at position 1 and 6 for MAP with values of 0.0322 and 0.0272, respectively. Although this strict evaluation might be less challenging from an IR point-of-view, this task offers most opportunities to improve the efficiency of a structure-aware retrieval system, because the strict notion of all structural query components like target and support elements drastically reduces the amount of result candidates per document and, hence, across the corpus. Clever precomputation of the main query building blocks, namely tag-term pairs with their full-content scores, and index structures for efficient sorted and random access on whole element blocks grouped by document ids allows for decent run times of a true graph-based query engine that lies in the order of efficient text IR systems. Here TopX can greatly accelerate query run times and achieve interactive response times at a remarkable result quality. Similar experiments provided in [10] yield average response times for typical INEX (CO and CAS) queries in between 0.1 and 0.7 seconds for the top 10-20 and still an average run time of about 20 seconds for the top 1,500 results as demanded by INEX (which is of course not exactly nice to handle for a top-k engine).

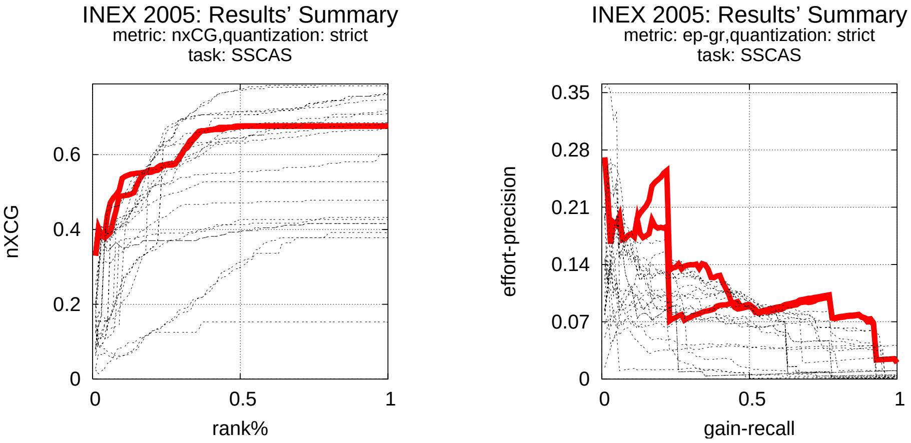


Fig. 6. TopX SSCAS runs

6 Experiments with XXL

The XXL Search Engine [7–9] was among the first XML search engines that supported content-and-structure queries with an IR-like scoring for content conditions. Focussing on aspects of semantic similarity conditions for tags and contents using ontologies, it applies an out-of-the-box text retrieval engine, namely Oracle's text engine, to evaluate content subqueries. Details of its architecture can be found in [8].

6.1 CO-Thorough

The CO.Thorough run basically represents the performance of the underlying text search engine. XXL automatically converted CO topics into corresponding Oracle text queries, using conjunctive combination of terms, enabling phrases, and applying some other simple heuristics that gave reasonable results with INEX 2003 and 2004. Surprisingly, this year's performance was not really convincing, with a rank 39 of 55 with inex_eval and the strict quantization (MAP 0.016), with similar results for the other metrics.

6.2 SSCAS

The results for the SSCAS run, where XXL has a higher influence on the outcome than with keyword-only topics, were much better. XXL is almost consistently among the top 10 for nxcg with the generalized quantization, with a peak rank of 2 for nxCG@25, and only slightly worse for strict quantization. For inex_eval, we achieved rank 11 with a MAP of 0.075. XXL has been especially built for this kind of strict structural match. The results are even better when taking the poor performance of the content-only run into account.

6.3 SVCAS and VVCAS

For the SSCAS run, XXL was configured to return a result only if it had a corresponding match (i.e., an element) for each subcondition of the query. For the SVCAS run, we relaxed this requirement and allowed results as soon as they had a match for the target subcondition, i.e., the subcondition whose result is returned as result of the query. This simple, 'andish'-like evaluation did surprisingly well, with top-10 ranks in several metrics.

For the VVCAS run, we additionally changed the tag of the target subcondition to the wildcard '*', accepting any element as result as long as it matches the associated content condition. However, this kind of relaxation turned out to be too coarse, so the results were quite poor with all metrics.

References

1. K.-C. Chang and S.-W. Hwang. Minimal probing: supporting expensive predicates for top-k queries. In *SIGMOD 2002*, pages 346–357, 2002.
2. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.
3. T. Grust. Accelerating XPath location steps. In *SIGMOD 2002*, pages 109–120, 2002.
4. U. Güntzer, W.-T. Balke, and W. Kießling. Optimizing multi-feature queries for image databases. In *VLDB 2000*, pages 419–428, 2000.
5. S. Nepal and M. V. Ramakrishna. Query processing issues in image (multimedia) databases. In *ICDE 1999*, pages 22–29, 1999.
6. S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, pages 232–241, 1994.
7. R. Schenkel, A. Theobald, and G. Weikum. XXL @ INEX 2003. In *INEX 2003 Workshop Proceedings*, pages 59–68, 2004.
8. R. Schenkel, A. Theobald, and G. Weikum. Semantic similarity search on semistructured data with the XXL search engine. *Information Retrieval*, 8(4):521–545, December 2005.
9. A. Theobald and G. Weikum. Adding Relevance to XML. In *WebDB 2000*, pages 105–124, 2000.
10. M. Theobald, R. Schenkel, and G. Weikum. An efficient and versatile query engine for TopX search. In *VLDB 2005*, pages 625–636, 2005.
11. M. Theobald, G. Weikum, and R. Schenkel. Top-k query evaluation with probabilistic guarantees. In *VLDB 2004*, pages 648–659, 2004.

When a few highly relevant answers are enough

Miro Lehtonen[1]

Department of Computer Science
P. O. Box 68 (Gustaf Hällströmin katu 2b)
FI–00014 University of Helsinki
Finland
Miro.Lehtonen@cs.Helsinki.FI

Abstract. Our XML retrieval system EXTIRP was slightly modified from the 2004 version for the INEX 2005 project. For the first time, the system is now completely independent of the document type of the XML documents in the collection, which justifies the use of the term "heterogeneous" when describing our methodology. Nevertheless, the 2005 version of EXTIRP is still an incomplete system that does not include query expansion or dynamic determination of the answer size. The latter is seen as a serious limitation because of the XCG-based metrics which favour systems that can adjust the size of the answer according to its relevance to the query. We put our main focus on the CO.Focussed task of the adhoc track although runs were submitted for other tasks, as well. Perhaps because of the incompleteness of our system, the initial results bring out the characteristics of our system better than in earlier years. Even when partially stripped, EXTIRP is capable of ranking the most obvious highly relevant answers at the top ranks better than many other systems. The relatively high precision at the top ranks is achieved at the cost of losing the sight of the marginally relevant content, which shows in some exceptionally steep curves, and the rankings among other systems that sink from the top ranks at low recall levels towards the bottom ranks at higher levels of recall. Another fact supporting our observation is that regardless of the metric, our runs are ranked higher with the strict quantisation than with any other quantisation function.

1 Introduction

The XML retrieval system at the University of Helsinki — EXTIRP — is completely independent of the XML document types as of 2005. In practice, the information coded in element names is ignored, which in turn lets us keep the doors open for collections of heterogeneous XML documents. The choice of ignoring the names of document structures also implies that our system specialises in the Content-Only type queries where only the content of the result elements has any significance.

2 Background

Before building two indices — one for words, one for phrases — EXTIRP divides the document collection into disjoint fragments. The root elements of the indexed fragments are chosen with an algorithm for *full-text fragment detection* which will be presented later in this paper in more detail. The disjoint fragments are then naturally treated as traditional documents which are independent of each other. The pros include that the traditional methods for information retrieval apply, so we use the vector space model with a weighting scheme based on the tfidf. The biggest of the cons is that the size of the indexed fragments is static, and if bigger or smaller answers are more appropriate for some query, the fragments have to be either divided further or combined into bigger fragments. Because of the previous challenges with fragment combination, we have left it for future research, and for now, the size of the answers that EXTIRP returns is determined when the fragments are chosen for indexing instead of dynamically adjusting the size according how relevant the fragment is to a query.

3 Detection of full-text fragments

4 Results

5 Conclusion

RMIT University at INEX 2005

Jovan Pehcevski, James A. Thom, and S. M. M. Tahaghoghi

School of CS and IT, RMIT University, Melbourne, Australia
{jovanp, jat, saied}@cs.rmit.edu.au

Abstract. Different scenarios of XML retrieval are analysed in INEX 2005, which reflect the different query interpretations and system behaviours that may be observed in the XML retrieval task. In this paper we report on the participation of the RMIT group in the INEX 2005 ad hoc track, where we design runs that investigate these XML retrieval scenarios. Our runs follow a hybrid XML retrieval approach that combines three information retrieval models with two ways of identifying the appropriate element granularity and two XML-specific heuristics to rank the final answers. We observe different behaviours when applying our hybrid approach to the different retrieval scenarios, suggesting that the optimal retrieval parameters are highly dependent on the nature of the XML retrieval task. Importantly, we show that using structural hints in content only topics is a useful feature which leads to a more precise search, irrespective of the XML retrieval scenario used.

1 Introduction

Most of the research activities in INEX 2005 focus on providing a detailed analysis of wide variety of aspects surrounding the XML retrieval task. There are seven tracks at INEX 2005, each exploring different applications of XML retrieval. Our group is actively involved in four of these tracks (ad hoc, interactive, multimedia [2], and heterogeneous), and we also contribute with additional activities related to the INEX evaluation methodology [9]. In this paper, we concentrate on the ad hoc track.

Two types of topics (queries) are explored in the ad hoc track: Content Only + Structure (`CO+S`) and Content And Structure (`CAS`). A `CO+S` topic is a request that typically ignores the document structure by only specifying plain query terms. However, there may be cases where adding structural hints to the query could result in a more precise search. Some INEX 2005 `CO+S` topics therefore express the same information need by either ignoring or including the structural hints (we call these `+S topics`). Figure 1 shows a snippet of the INEX 2005 `+S` topic 203 that was proposed by our group, where two topic fields – `title` and `castitle` – are used to represent the two possibilities. On the other hand, a `CAS` topic is a request that contains references to the document structure and explicitly specifies the type of the returned answer elements (the target element) and the structural constraints on the contained elements of the search context (the support elements).

```
<inex_topic topic_id="203" query_type="CO+S" ct_no="5">
   <title> code signing verification </title>
   <castitle> //sec[about(., code signing verification)] </castitle>
   <description> Find documents or document components, most probably sections,
   that describe the approach of code signing and verification. </description>
   <narrative> I am working in a company that authenticates a wide range of web
   database applications from different software vendors. [...] </narrative>
</inex_topic>
```

Fig. 1. A snippet of the INEX 2005 CO+S topic 203.

In accordance to the above topic types, three sub-tasks have been established within the INEX 2005 ad hoc track, and different retrieval strategies are explored in each sub-task. These are the CO, the +S, and the CAS sub-task (we denote the first two as CO+S), reflecting the three types of topics used. Three retrieval strategies are explored in the CO+S sub-task: Focused, Thorough, and FetchBrowse, which reflect different aspects of the XML retrieval task. On the other hand, four retrieval strategies are explored in the CAS sub-task: VV, VS, SV, and SS, which correspond to the way the structural constraints in the target and the support elements are interpreted [8].

The system we use in the INEX 2005 ad hoc track follows a *hybrid* XML retrieval approach, combining information retrieval features from Zettair[1] (a full-text search engine) with XML-specific retrieval features from eXist[2] (a native XML database). The hybrid approach can be seen as a “fetch and browse” [1] XML retrieval approach, since full articles estimated as likely to be relevant to a query are first retrieved by Zettair (the *fetch* phase), and then the most specific elements within these articles are extracted by eXist (the *browse* phase) [10].

To calculate the similarity score of an article to a query (represented by terms that appear in the title part of an INEX topic), a similarity measure – typically based on a theoretical model of information retrieval – is used by Zettair. Three similarity measures are currently implemented, each based on one of the following three information retrieval models: the vector-space model, the probabilistic model, and the language model. For the *fetch* phase of our hybrid system, we investigate which information retrieval model yields best effectiveness for full article retrieval.

To identify and rank the appropriate granularity of elements to return as answers, we use a retrieval module that utilises the structural information in the eXist list of extracted elements. Our retrieval module presents what we call *Coherent Retrieval Elements* (CREs) as final answers. For the *browse* phase of our hybrid system, we investigate which combining choice – among the two ways

[1] http://www.seg.rmit.edu.au/zettair/

[2] http://exist-db.org/

for identifying CREs and the two XML-specific heuristics for ranking the CREs – yields best effectiveness for element retrieval.

The remainder of this paper is organised as follows. In Section 2 we provide an explanation of the hybrid XML retrieval approach used by our INEX ad hoc runs; in particular, we describe the three information retrieval models used by Zettair to rank the whole documents, and the various XML-specific algorithms used by our hybrid system to identify the appropriate level of element granularity and to rank the final answers. A detailed description of the runs we use for the CO+S and the CAS sub-tasks is provided in Section 3. In Section 4, we present results of our runs for each retrieval strategy in the two ad hoc sub-tasks, by evaluating their retrieval effectiveness with using the official INEX 2005 metrics. We conclude in Section 5 with a brief discussion of our findings.

2 Hybrid XML Retrieval

In this section, we present a brief description of the three information retrieval models implemented in Zettair, the two algorithms for identifying the CREs, and the two heuristics for ranking the CREs, all of which are used by our hybrid system.

2.1 Information Retrieval Models

Many similarity measures for document retrieval have been proposed, and almost all follow one of the three major information retrieval models: the *vector-space model*, the *probabilistic model* and the *language model*. In this section, we describe the three similarity measures that are currently used in Zettair, which respectively implement each of these three retrieval models.

Term statistics for ranked retrieval The *similarity* of a document to a query, denoted as $S_{q,d}$, indicates how closely the content of the document matches that of the query. To calculate the query-document similarity, statistical information about the distribution of the query terms – within both the document and the collection as a whole – is often necessary. These term statistics are then subsequently utilised by the similarity measure. Following the notation and definitions of Zobel and Moffat [13], we define the basic term statistics as:

- q, a query;
- t, a query term;
- d, a document;
- $N_{\mathcal{D}}$, the number of all the documents in the collection;
- For each term t:
  - $f_{d,t}$, the frequency of t in the document d;
  - $N_{\mathcal{D}_t}$, the number of documents containing the term t (irrespective of the term frequency in each document); and
  - $f_{q,t}$, the frequency of t in query q.

- For each document d:
    - $f_d = |d|$, the document length approximation (sum of all the term frequencies in d).
- For the query q:
    - $f_q = |q|$, the query length.

We also denote the following sets:

- $\mathcal{D}$, the set of all the documents in the collection;
- $\mathcal{D}_t$, the set of documents containing term t;
- $\mathcal{T}$, the set of distinct terms in the collection;
- $\mathcal{T}_d$, the set of distinct terms in the document d;
- $\mathcal{T}_q$, the set of distinct terms in the query, and $\mathcal{T}_{q,d} = \mathcal{T}_q \cap \mathcal{T}_d$.

Vector-space model In the vector-space model, both the document and the query are representations of n-dimensional vectors, where n is the number of distinct terms observed in the document collection. The best-known technique for computing similarity under the vector-space model is the cosine measure, where the similarity between a document and the query is computed as the cosine of the angle between their vectors.

Zettair uses pivoted cosine document length normalisation [11] to compute the query-document similarity under the vector-space model (we call it `PCosine`). The formulation is described as follows.

$$S_{q,d} = \frac{1}{W_D \times W_q} \times \sum_{t \in \mathcal{T}_{q,d}} (1 + \log_e f_{d,t}) \times \log_e \left(1 + \frac{N_{\mathcal{D}}}{N_{\mathcal{D}_t}}\right)$$

where $W_D = \left((1.0 - s) + s \times \frac{W_d}{W_{AL}}\right)$ represents the pivoted document length normalisation, and $W_q = \sqrt{\sum_{t \in \mathcal{T}_q} \left[\log_e \left(1 + \frac{N_{\mathcal{D}}}{N_{\mathcal{D}_t}}\right)\right]^2}$ is the query length representation. The parameter s represents the *slope* (we use the value of 0.25), whereas W_d and W_{AL} represent the document length (usually taken as f_d) and the average document length (over all documents in $\mathcal{D}$), respectively.

Probabilistic model Probabilistic models of information retrieval are based on the principle that documents should be ranked according to the decreasing probability of their relevance to the user information need. Zettair uses the Okapi BM25 probabilistic model developed by Sparck Jones, Walker, and Robertson [4], which has proved highly successful in a wide range of experiments (we call it `Okapi`). The formulation is described as follows.

$$S_{q,d} = \sum_{t \in \mathcal{T}_{q,d}} w_t \times \frac{(k_1 + 1)\, f_{d,t}}{K + f_{d,t}} \times \frac{(k_3 + 1)\, f_{q,t}}{k_3 + f_{q,t}}$$

Article	Matching element
co/2000/r7108	/article[1]/bdy[1]/sec[1]/ip1[1]
co/2000/r7108	/article[1]/bdy[1]/sec[1]/p[1]
co/2000/r7108	/article[1]/bdy[1]/sec[2]/st[1]
co/2000/r7108	/article[1]/bdy[1]/sec[2]/p[2]
co/2000/r7108	/article[1]/bdy[1]/sec[2]/p[3]
co/2000/r7108	/article[1]/bdy[1]/sec[2]/p[4]
co/2000/r7108	/article[1]/bdy[1]/sec[4]/p[1]
co/2000/r7108	/article[1]/bdy[1]/sec[6]/ip1[1]
co/2000/r7108	/article[1]/bm[1]/app[1]/p[2]
co/2000/r7108	/article[1]/bm[1]/app[1]/p[3]
co/2000/r7108	/article[1]/bm[1]/app[1]/p[4]

Table 1. eXist list of matching elements for INEX 2005 CO topic 203 and article `co/2000/r7108`. The elements in the list are generated by using an eXist OR query.

where $w_t = \log_e \left(\frac{N_{\mathcal{D}} - N_{\mathcal{D}_t} + 0.5}{N_{\mathcal{D}_t} + 0.5} \right)$ is a representation of inverse document frequency, $K = k_1 \times \left[(1 - b) + \frac{b \cdot W_d}{W_{AL}} \right]$, and k_1, b and k_3 are constants, in the range 1.2 to 1.5 (we use 1.2), 0.6 to 0.75 (we use 0.75), and 1000 (effectively infinite), respectively. W_d and W_{AL} represent the document length and the average document length.

Language model Language models are probability distributions that aim to capture the statistical regularities of natural language use. In information retrieval, language modelling involves estimating the likelihood that both the document and the query could have been generated by the same language model. Zettair uses a query likelihood approach with Dirichlet smoothing [12] (we call it `Dirichlet`). The formulation is described as follows.

$$S_{q,d} = f_q \times \log \lambda_d + \sum_{t \in \mathcal{T}_{q,d}} \log \left(\frac{N_{\mathcal{D}} \times f_{d,t}}{\mu \times N_{\mathcal{D}_t}} + 1 \right)$$

where μ is a smoothing parameter (we use the value of 1000), and $\lambda_d = \mu / (\mu + f_d)$.

2.2 Identifying the appropriate element granularity

For each INEX topic (`CO`, `+S`, or `CAS`), a topic translation module is first used to automatically translate the underlying information need into a Zettair query. Terms that appear in the `title` part of the topic (with all structural query constraints completely removed) are used to formulate the Zettair query. A list of (up to) 500 `article` elements – presented in a descending order according to their estimated likelihood of relevance – is then returned as a resulting answer list for the INEX topic[3].

[3] We retrieve (up to) 500 rather than 1500 articles because roughly that number of articles is used to generate the pool of retrieved articles for relevance judgements.

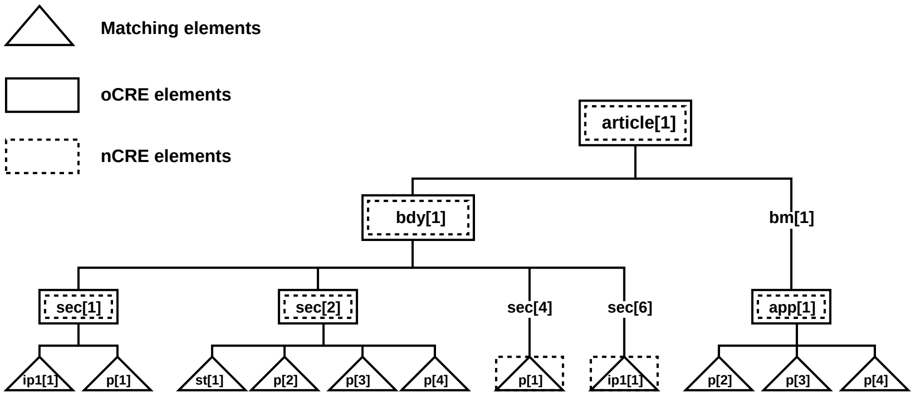


Fig. 2. Identifying appropriate element granularity: Matching, oCRE, and nCRE elements for INEX 2005 topic 203 and article co/2000/r7108.

To retrieve *elements* rather than full articles, a second topic translation module is used to formulate the eXist query. Depending on the topic type, either terms only, or both terms and structural query constraints from the INEX topic are used to formulate the eXist query. We use the OR eXist query operator to generate the element answer list for a given topic. The answer list contains (up to) 1500 matching elements, which are taken from articles that appear *highest* in the ranked list of articles previously returned by Zettair.

Consider the eXist answer list example shown in Table 1. The list shows matching elements for the INEX 2005 CO topic 203 after the OR eXist query operator is used (which means that each matching element in the list contains *one* or *more* query terms). The matching elements in the eXist answer list represent most specific (leaf) elements, and eXist correctly presents these elements in document order.

To effectively utilise the information contained in the resulting list of matching elements, we use a retrieval module capable of identifying the appropriate *granularity* of elements to return as answers. We call these answer elements as *Coherent Retrieval Elements* (CREs) [10]. To identify the CREs, our module first sequentially processes the list of matching elements, starting from the first element down to the last. For each pair of matching elements, their *most specific ancestor* is chosen to represent an answer element (a CRE). We denote these answer elements as oCRE elements.

The rationale behind choosing only oCRE elements as answers stems from the expectation that these elements are likely to provide better context for the contained textual information than that provided by each of their descendent leaf elements. However, different topics typically express information needs that are quite diverse in nature, resulting in relevant answers that often represent very specific elements [3, 10]. Therefore, the problem of only presenting the oCRE elements as answers is that in most cases the matching (and thus very specific) elements are *not included* in the final answer list. To cater for this, our retrieval module supports a second, alternative algorithm for identifying the CREs. The difference from the original `oCRE` algorithm is that, after sequentially processing all the pairs of matching elements, those matching elements whose *immediate parents* are not identified as CREs are also included in the final list of answers. The rationale behind this choice is that we expect these newly included matching elements to allow for more focused retrieval. We denote these answer elements as `nCRE` elements.

When the eXist list of matching elements contains only one element, both the `oCRE` and `nCRE` algorithms produce the same result: the matching element. In this case there is no supporting evidence for the ancestors of the matching element to be identified as CREs.

Figure 2 shows a tree representation of the eXist list of matching elements previously shown in Table 1. The matching elements appear within the triangle boxes, the `oCRE` elements appear within the solid square boxes, while the `nCRE` elements appear within dashed square boxes. The figure also shows elements that are neither matching elements nor CREs.

Once the CREs are identified, we still need to find a way to *rank* and present them according to their *estimated likelihood of relevance.* Next, we describe a range of heuristics used by our retrieval module to rank the resulting list of CREs.

2.3 Ranking the answer elements

To determine the ranks of CREs in the final answer list, our module uses a combination of the following XML-specific heuristics:

1. The number of distinct query terms that appear in a CRE — more distinct query term appearances (`T`) or less distinct query term appearances (`t`);
2. The length of the absolute path of the CRE, taken from the root element — longer path (`P`) or shorter path (`p`); and
3. The frequency of all the query terms in a CRE — more frequent (`F`) or less frequent (`f`).

There are eight distinct heuristic combinations (such as `TPF` or `TpF`) that can be explored in order to determine the final rank of a CRE, provided the ordering of the heuristics is preserved as above. However, we also expect that a reordering of the above heuristics could influence different CRE rankings. We therefore analyse all possible CRE heuristic combinations (16 in total, since we take the third heuristic as complementary to the other two and therefore always apply at the end). Preliminary experiments using the INEX 2004 test collection showed that two heuristic combinations – `TPF` and `PTF` – perform better than the others in a case where more specific elements are target of retrieval (the `nCRE` algorithm was used by the retrieval module to identify the CREs).

The two heuristic combinations can be interpreted as follows. With `TPF`, first the CREs are sorted in a descending order according to the number of

Article	nCRE answer element	T-matches	P-length	F-frequency
co/2000/r7108	/article[1]/bdy[1]/sec[2]	3	3	9
co/2000/r7108	/article[1]/bdy[1]	3	2	31
co/2000/r7108	/article[1]	3	1	39
co/2000/r7108	/article[1]/bdy[1]/sec[6]/ip1[1]	2	4	2
co/2000/r7108	/article[1]/bm[1]/app[1]	2	3	8
co/2000/r7108	/article[1]/bdy[1]/sec[1]	2	3	5
co/2000/r7108	/article[1]/bdy[1]/sec[4]/p[1]	1	4	2

Table 2. Ranked list of nCRE elements using the TPF heuristic combination for article co/2000/r7108. The query used is "code signing verification", which represents the title part of the INEX 2005 topic 203.

distinct query terms a CRE contains (the more distinct query terms it contains, the higher its rank). Next, if two CREs contain the same number of distinct query terms, the one with the longer length of its absolute path is ranked higher (which ensures that more specific elements are preferred over less specific ones). Last, if the lengths of the two absolute paths are the same, the CRE with more frequent query term appearances is ranked higher than the CRE where query terms appear less frequently. For example, after applying the TPF heuristic on the nCRE answer elements shown in Figure 2, we produce the final ranked list of CREs as shown in Table 2.

The table shows that when TPF heuristic is used, less specific and more general CREs tend to be preferred over more specific and less general CREs. However, we expect other heuristic combinations to be more suitable for different XML retrieval tasks. For example, to produce more specific and less general CREs early in the ranking, we could easily switch the heuristic combination and use PTF instead. With PTF, the CREs are first sorted in a descending order according to the length of the absolute path of a CRE (where the longer CRE path results in a higher rank). Next, if the lengths of the two absolute paths are the same, the CRE that contains more number of distinct query terms is ranked higher. Last, if it also happens that the two CREs contain same number of distinct query terms, the CRE with more frequent query term appearances is ranked higher.

Preliminary experiments using the INEX 2004 test collection have shown that the system performance quickly degrades when using the PTF ranking heuristic, merely because a large number of highly ranked elements typically contain only one query term. We therefore use a modification of this heuristic in our retrieval module, which ensures that all the CREs that contain exactly one query term are moved at the end of the ranked list (where ties are broken by the F heuristic). We denote this modified heuristic combination as PTF2.

The final ranked list of answers when the PTF2 heuristic combination is used on the nCRE answer elements is shown in Table 3.

Article	nCRE answer element	P-length	T-matches	F-frequency
co/2000/r7108	/article[1]/bdy[1]/sec[6]/ip1[1]	4	2	2
co/2000/r7108	/article[1]/bdy[1]/sec[2]	3	3	9
co/2000/r7108	/article[1]/bm[1]/app[1]	3	2	8
co/2000/r7108	/article[1]/bdy[1]/sec[1]	3	2	5
co/2000/r7108	/article[1]/bdy[1]	2	3	31
co/2000/r7108	/article[1]	1	3	39
co/2000/r7108	/article[1]/bdy[1]/sec[4]/p[1]	4	1	2

Table 3. Ranked list of nCRE elements using the PTF2 heuristic combination for article co/2000/r7108. The query used is "code signing verification", which represents the title part of the INEX 2005 topic 203.

3 Runs description

The following sections provide a detailed description of our runs for each retrieval strategy in both (CO+S and CAS) sub-tasks.

3.1 CO+S sub-task

We submitted six runs for each of the Thorough, Focused, and FetchBrowse retrieval strategies, resulting in 18 runs in total for the CO+S sub-task. All the runs in Thorough and Focused strategies use the Okapi BM25 similarity measure in Zettair to generate the initial list of ranked articles. Next, we provide a detailed explanation of our runs for each of the three CO+S retrieval strategies.

Thorough retrieval strategy

- nCRE-PTF2 and nCRE-S-PTF2 – using the hybrid system with nCRE answer elements and the PTF2 ranking heuristic in the retrieval module. For each +S topic, the structural query constraints are either completely removed (nCRE-PTF2) or strictly followed (nCRE-S-PTF2).
- oCRE-PTF2 and oCRE-S-PTF2 – same as with the two previous runs, except that oCRE answer elements are used by the hybrid system.
- nCRE-TPF and nCRE-S-TPF – using the hybrid system with nCRE answer elements and the TPF ranking heuristic in the retrieval module. The structural query constraints in each +S topic are either completely removed (nCRE-TPF) or strictly followed (nCRE-S-TPF).

Our goals with the Thorough retrieval strategy are threefold. First, we aim to explore which choice of identifying answer elements (oCRE or nCRE) yields best performance for the hybrid system. Second, for a particular choice of answer elements, we also aim to investigate the impact of the two ranking heuristics (PTF2 and TPF) on the system performance. Last, for a particular choice of answer elements and ranking heuristic, we aim to investigate the usefulness of retaining the structural constraints in the +S topics.

`Focused` retrieval strategy

- `nCRE-PTF2-NO` and `nCRE-S-PTF2-NO` – using the hybrid system with `nCRE` answer elements and the `PTF2` ranking heuristic, with overlap among the answer elements completely removed (using the top-down filtering approach). As in the Thorough strategy, the structural query constraints in each `+S` topic are either completely removed (`nCRE-PTF2-NO`) or strictly followed (`nCRE-S-PTF2-NO`).
- `oCRE-PTF2-NO` and `oCRE-S-PTF2-NO` – same as with the two previous non-overlapping runs, except that `oCRE` answer elements are used by the hybrid system.
- `nCRE-TPF-NO` and `nCRE-S-TPF-NO` – using the hybrid system with `nCRE` answer elements and the `TPF` ranking heuristic in the retrieval module, with overlap among the answer elements completely removed. The structural query constraints in each `+S` topic are either completely removed (`nCRE-TPF-NO`) or strictly followed (`nCRE-S-TPF-NO`).

By using the six non-overlapping runs above, we want to check whether the relative difference in the runs behaviour observed with the `Thorough` strategy would also be observed with the `Focused` strategy. That way, it may be possible to determine whether the system performance is also dependent on the nature of the retrieval task.

`FetchBrowse` retrieval strategy

- `nCRE-Okapi-PTF2` and `nCRE-S-Okapi-PTF2` – using the hybrid system with `nCRE` answer elements and the `PTF2` ranking heuristic in the retrieval module. For each `+S` topic, the structural query constraints are either completely removed (`nCRE-PTF2`) or strictly followed (`nCRE-S-PTF2`). Okapi BM25 is used in Zettair to generate the initial list of ranked articles.
- `nCRE-PCosine-PTF2` and `nCRE-S-PCosine-PTF2` – using the hybrid system with `nCRE` answer elements and the `PTF2` ranking heuristic in the retrieval module. For each `+S` topic, the structural query constraints are either completely removed (`nCRE-PTF2`) or strictly followed (`nCRE-S-PTF2`). Pivoted cosine document length normalisation is used in Zettair to generate the initial list of ranked articles.
- `nCRE-Dirichlet-PTF2` and `nCRE-S-Dirichlet-PTF2` – using the hybrid system with `nCRE` answer elements and the `PTF2` ranking heuristic in the retrieval module. For each `+S` topic, the structural query constraints are either completely removed (`nCRE-PTF2`) or strictly followed (`nCRE-S-PTF2`). Dirichlet language modelling representation is used in Zettair to generate the initial list of ranked articles.

Our goals with the `FetchBrowse` retrieval strategy are threefold. First, we aim to explore which of the three implemented information retrieval models yields best system performance for full article retrieval. Second, we aim to investigate the extent to which each of the three information retrieval models influences the system performance for element retrieval. Last, for a particular information retrieval model, we also check the usefulness of retaining the structural constraints in the +S topics.

3.2 CAS sub-task

We submitted two runs for each of the VV, SV, VS, and SS retrieval strategies, resulting in eight runs in total for the CAS sub-task. All the runs use the Okapi BM25 similarity measure in Zettair to generate the initial list of ranked articles. By evaluating each of these CAS runs against each of the four retrieval strategies, we aim to determine whether the way structural constraints are interpreted – in support elements, in the target element, or in both – has an effect on the overall system performance. Further, our goal with separately using each of the four CAS retrieval strategies is to investigate which of the two ranking heuristics (PTF2 or TPF) yields better retrieval effectiveness. The runs in each retrieval strategy are explained as follows.

VV retrieval strategy

- nCRE-VV-PTF2 – using the hybrid system where structural constraints in support elements and the target element of each CAS topic are interpreted as vague, leaving plain query terms only. The hybrid system uses nCRE answer elements and the PTF2 ranking heuristic in the retrieval module.
- nCRE-VV-TPF – same as with the previous run, except that TPF ranking heuristic is used in the retrieval module.

VS retrieval strategy

- nCRE-VS-PTF2 – using the hybrid system where structural constraints in support elements of each CAS topic are strictly followed, while the constraints in the target element are interpreted as vague (allowing any element granularity). The hybrid system uses nCRE answer elements and the PTF2 ranking heuristic in the retrieval module.
- nCRE-VS-TPF – same as with the previous run, except that TPF ranking heuristic is used in the retrieval module.

SV retrieval strategy

- SV-PTF2 – using the hybrid system where structural constraints in the target element of each CAS topic are strictly followed, while the constraints in support elements are interpreted as vague. The element answer granularity is already determined by the target element, so the retrieval module only uses the PTF2 heuristic combination to rank the final answers.
- SV-TPF – same as with the previous run, except that TPF ranking heuristic is used in the retrieval module.

SS **retrieval strategy**

- SS-PTF2 – using the hybrid system where structural constraints in support elements and the target element of each CAS topic are strictly followed. The retrieval module uses the PTF2 heuristic combination to rank the final answers.
- SS-TPF – same as with the previous run, except that TPF ranking heuristic is used in the retrieval module.

4 Experiments and results

In this section, we present results of our runs when their performance is evaluated for each retrieval strategy in both the CO+S and CAS sub-tasks. To evaluate the retrieval performance, a new set of metrics is adopted in INEX 2005, which belong to the eXtended Cumulated Gain (XCG) family of metrics [6, 7]. We use the strict quantisation function with the following three official INEX 2005 metrics [5] to measure the retrieval effectiveness of our runs:

1. nxCG – for a given rank r, nxCG[r] measures the relative gain a user has accumulated up to that rank, compared to the gain they could have accumulated if the system had produced the optimal ranking. We report values for MAnxCG[r], calculated as the average of nxCG[i] values for $i = 1$ to r.
2. ep/gr (effort-precision/gain-recall) – measures the amount of relative effort (as the number of visited ranks) a user is required to spend compared to the effort they could have spent when inspecting an optimal ranking for a particular cumulated gain level. We report values for MAep and iMAep, which represent values for mean average effort-precision calculated at natural or standard gain-recall points, respectively.
3. Q and R – modified normalised cumulated gain measures which employ bonus gain functions that directly incorporate the rank position of the cumulated gain level. We report values for both the Q and R measures.

4.1 CO+S sub-task

Three retrieval strategies are explored in the CO+S sub-task: Thorough, Focused, and FetchBrowse. In the following we present the performance results of our runs for each of these strategies.

Thorough **retrieval strategy** The evaluation results of our INEX 2005 CO+S runs for this strategy are shown in the upper part of Table 4. Several observations can be drawn from these results.

First, using the nCRE algorithm for identifying answer elements in our hybrid system yields better overall performance than when the oCRE algorithm is used. This finding validates our choice of retaining some specific matching elements as answers. However, the higher oCRE value observed with the iMAep measure of

Run	MAnxCG[r]							ep/gr		Q	R
	1	3	5	10	100	500	1500	MAep	iMAep		
Thorough											
nCRE-PTF2	**0.039**	0.043	0.040	0.032	0.048	**0.100**	0.188	0.001	0.007	0.001	**0.015**
nCRE-S-PTF2	**0.039**	**0.053**	**0.049**	0.040	0.048	0.085	0.159	0.001	0.007	0.001	0.012
nCRE-TPF	**0.039**	0.039	0.037	0.034	0.048	**0.100**	**0.189**	0.001	0.008	0.001	**0.015**
nCRE-S-TPF	**0.039**	0.049	0.047	**0.041**	**0.050**	0.085	0.159	0.001	0.007	0.001	0.012
oCRE-PTF2	0.000	0.011	0.010	0.015	**0.050**	0.088	0.166	0.001	**0.009**	0.001	0.012
oCRE-S-PTF2	0.000	0.021	0.020	0.018	0.047	0.084	0.147	0.001	0.008	0.001	0.010
Focused											
nCRE-PTF2-NO	**0.039**	0.043	0.041	0.043	0.052	**0.140**	**0.236**	0.012	0.011	0.016	**0.024**
nCRE-S-PTF2-NO	**0.039**	**0.053**	**0.051**	**0.048**	0.060	0.107	0.171	**0.014**	**0.012**	**0.018**	0.021
nCRE-TPF-NO	**0.039**	0.039	0.036	0.038	0.057	0.092	0.148	0.011	0.009	0.013	0.016
nCRE-S-TPF-NO	**0.039**	0.049	0.045	0.042	**0.062**	0.086	0.121	0.012	0.011	0.015	0.016
oCRE-PTF2-NO	0.000	0.011	0.018	0.023	0.054	0.114	0.159	0.011	0.010	0.015	0.016
oCRE-S-PTF2-NO	0.000	0.021	0.023	0.023	0.057	0.098	0.144	0.013	0.010	0.017	0.016

Table 4. Evaluation results of our INEX 2005 CO+S runs for the Thorough (upper part) and Focused (lower part) retrieval strategies. Strict quantisation function is used with each INEX metric to generate the above numbers. For each retrieval strategy, the best results obtained from each of the INEX 2005 evaluation measures are shown in bold.

the **ep/gr** metric suggests that less number of visited ranks may be required for achieving a particular level of cumulated gain when the oCRE algorithm is used.

Second, the relative gain a user has accumulated after any of the first ten elements are retrieved by the hybrid system is higher when using the PTF2 ranking heuristic than when using TPF, although when retrieving ten and more elements the gain seems to be higher when using the TPF ranking heuristic.

Last, using structural hints from the +S topics also increases the system performance when up to ten elements per topic are returned. As shown in the table, at any of the first ten elements returned, each of the three +S runs performs consistently better than its corresponding CO run.

Focused retrieval strategy The lower part of Table 4 shows evaluation results of our INEX 2005 CO+S runs for the Focused retrieval strategy. In this case, using the nCRE algorithm for identifying answer elements yields a consistent performance improvement than when using the oCRE algorithm. Also, when retrieving any number of elements (except 100), the PTF2 ranking heuristic in our hybrid system produces higher cumulated user gain than the TPF ranking heuristic.

For each of the three non-overlapping runs, using structural hints from the +S topics increases the system performance when up to ten elements per topic are returned. However, unlike for the Thorough retrieval strategy, in the Focused retrieval strategy values for both the **ep/gr** and **Q** measures also show that less

Run	inex_eval (strict)					inex_eval (SOG)				
	5	10	100	1500	iMAP	5	10	100	1500	iMAP
FetchBrowse-D										
nCRE-Okapi-PTF2	0.160	**0.120**	**0.028**	**0.017**	0.291	0.152	0.107	**0.028**	0.009	0.152
nCRE-S-Okapi-PTF2	**0.200**	**0.120**	0.026	0.005	**0.298**	0.137	0.090	0.024	0.005	0.133
nCRE-PCosine-PTF2	0.120	**0.120**	0.022	**0.017**	0.280	0.131	**0.119**	0.025	0.009	0.131
nCRE-S-PCosine-PTF2	0.160	**0.120**	0.022	0.004	0.290	0.120	0.106	0.023	0.005	0.117
nCRE-Dirichlet-PTF2	0.120	0.080	0.026	**0.017**	0.278	**0.155**	0.107	**0.028**	**0.010**	**0.157**
nCRE-S-Dirichlet-PTF2	0.160	0.080	0.026	0.005	0.284	0.136	0.088	0.024	0.005	0.155
FetchBrowse										
nCRE-Okapi-PTF2	0.052	0.044	**0.033**	**0.008**	**0.018**	0.155	0.152	0.107	**0.028**	**0.028**
nCRE-S-Okapi-PTF2	**0.074**	0.052	0.024	0.005	0.015	**0.210**	**0.173**	0.077	0.020	0.023
nCRE-PCosine-PTF2	0.052	0.052	0.030	0.007	0.017	0.112	0.126	0.088	0.027	0.021
nCRE-S-PCosine-PTF2	0.059	**0.056**	0.026	0.005	0.013	0.183	0.156	0.071	0.020	0.019
nCRE-Dirichlet-PTF2	0.067	0.048	**0.033**	**0.008**	0.017	0.181	0.161	**0.108**	**0.028**	**0.028**
nCRE-S-Dirichlet-PTF2	**0.074**	0.052	0.023	0.005	0.012	0.198	0.168	0.073	0.019	0.022

Table 5. Evaluation results of our INEX 2005 CO+S runs for the FetchBrowse-D (upper part) and FetchBrowse (lower part) retrieval strategies. Strict and SOG quantisation functions are used with the inex_eval metric to generate the above numbers. Values for iMAP represent interpolated mean average precision values calculated at standard recall points. For each retrieval strategy, the best results obtained from each of the two quantisation functions are shown in bold.

number of visited ranks are required for achieving a particular level of cumulated gain. This finding suggests that the nature of the retrieval task (such as Thorough or Focused) influences how structural constraints in the INEX +S topics should be interpreted. More precisely, using structural hints from the INEX +S topics seems to be more useful for Focused than for the Thorough retrieval strategy.

FetchBrowse retrieval strategy Table 5 shows evaluation results of our INEX 2005 CO+S runs for the FetchBrowse retrieval strategy. We use strict and SOG quantisation functions with inex_eval [6] to generate the above numbers.

The upper part of Table 5 shows results when only full articles represent units of retrieval. We observe that when *highly relevant* articles are target of retrieval (by using strict quantisation function), best overall system performance is achieved when using the Okapi similarity measure. Of the other two implemented measures, PCosine seems to work better than Dirichlet with ten or less articles returned. We also observe an increase in both the average precision (iMAP) and the precision at five articles returned when the structural constraints in the +S topics are strictly followed, irrespective of which similarity measure is used.

When *more specific* articles are target of retrieval (by using SOG), best overall system performance is achieved when using the Dirichlet similarity measure,

	MAnxCG[r]							**ep/gr**			
Run	1	3	5	10	100	500	1500	MAep	iMAep	**Q**	**R**
VV											
nCRE-VV-PTF2	0.000	**0.012**	**0.017**	0.016	0.040	0.090	0.127	0.001	0.007	0.002	0.026
nCRE-VV-TPF	0.000	0.000	0.013	**0.039**	**0.047**	**0.091**	**0.128**	**0.002**	0.007	0.002	0.026
VS											
nCRE-VS-PTF2	0.167	0.120	0.102	0.073	0.052	0.101	0.126	0.001	0.003	0.001	0.015
nCRE-VS-TPF	0.167	0.120	**0.109**	**0.087**	**0.054**	0.101	0.126	0.001	0.003	0.001	0.015
SV											
SV-PTF2	0.000	0.078	0.101	0.115	0.181	0.349	0.433	0.013	0.046	0.015	0.074
SV-TPF	0.000	0.078	**0.117**	**0.142**	**0.188**	**0.352**	**0.435**	0.013	**0.047**	0.015	0.074
SS											
SS-PTF2	0.000	0.125	0.165	0.167	0.312	0.355	0.388	0.011	0.036	0.012	0.068
SS-TPF	0.000	0.125	**0.185**	**0.201**	**0.320**	**0.356**	**0.389**	**0.012**	0.036	**0.013**	0.068

Table 6. Evaluation results of our INEX 2005 CAS runs for the VV, VS, SV, and SS retrieval strategies. Strict quantisation function is used with each INEX metric to generate the above numbers. For each retrieval strategy, the best results obtained from each of the INEX 2005 evaluation measures are shown in bold.

although with ten articles returned PCosine seems to work best. Interestingly, in this case there is no increase in system performance when the structural constraints in the +S topics are strictly followed.

The lower part of Table 5 shows results for the FetchBrowse retrieval strategy when elements are units of retrieval, where we investigate the extent to which each of the three similarity measures influences the system performance. We observe that for element retrieval the system performance is almost identical to the one observed in full article retrieval, except that, when structural constraints in the +S topics are strictly followed, there is a constant increase in precision with five and ten elements returned, irrespective of the similarity measure *and* the quantisation function used. This finding shows that using structural hints in the INEX +S topics is also a useful feature in the FetchBrowse retrieval strategy.

4.2 CAS sub-task

Four retrieval strategies are explored in the CAS sub-task: VV, VS, SV, and SS. Table 6 presents the performance results of our CAS runs for each of these strategies.

In the VV retrieval strategy all the the structural constraints in support elements and the target element are completely removed, leaving plain query terms only. From Table 6 we observe that, similarly as with the observed behaviour for the Thorough retrieval strategy of the CO+S task, the relative gain a user has accumulated after less than ten elements retrieved is higher when using the PTF2 ranking heuristic than when using TPF, although when retrieving ten and more elements the cumulated gain is higher when using the TPF ranking heuristic.

Table 6 also shows that the observed system performance is almost identical for any of the VS, SV, or SS retrieval strategies. Overall, the TPF ranking heuristic in our hybrid system produces equal and, in many cases, better performance than the PTF2 heuristic. This is especially evident for the latter two strategies, which allow for the target element to be strictly matched.

5 Conclusions

In this paper we have reported on our participation in the INEX 2005 ad-hoc track. We have tested three information retrieval models with two ways of identifying the appropriate element granularity and two XML-specific ranking heuristics in both the CO+S and CAS sub-tasks, mainly to investigate different aspects of XML retrieval.

For the CO sub-task, we have shown that using the nCRE elements as a choice for answer elements yields better overall performance for our hybrid system than when the oCRE answer elements are used. Also, the obtained cumulated gain for users is higher when using the PTF2 ranking heuristic than when using TPF, although for the Thorough retrieval strategy in cases where ten or more elements are retrieved, the gain seems to be higher when using the TPF ranking heuristic. Using structural hints in the +S topics increases the system performance when up to ten elements per topic are returned, although we observed that this is also dependent on the retrieval strategy. The Focused retrieval strategy seems to benefit more from the structural hints than the other two retrieval strategies.

For the CAS sub-task we observed that, regardless of the way the constraints in a CAS topic are interpreted, the TPF ranking heuristic in our hybrid system produces at least equal and in many cases better performance than the PTF2 ranking heuristic, which is particularly true for strategies that allow for the target element to be strictly matched. Only for the VV retrieval strategy where structural constraints in the CAS topics are completely removed is the PTF2 ranking heuristic better than TPF, and that is true only in cases where less than ten elements per topic are returned.

However, for the CAS sub-task we also plan to carry out additional experiments where each of the CAS runs will be evaluated against *each* of the four retrieval strategies (VV, VS, SV, and SS). Our goal is to determine whether the way structural constraints are interpreted – in support elements, in the target element, or in both – has the same effect on the overall system performance under different CAS retrieval strategies.

References

1. Y. Chiaramella, P. Mulhem, and F. Fourel. A model for multimedia information retrieval. Technical report, FERMI ESPRIT BRA 8134, University of Glasgow, April 1996.

2. D.N.F. Awang Iskandar, J. Pehcevski, J. A. Thom, and S. M. M. Tahaghoghi. Combining image and structured text retrieval. In *Pre-Proceedings of the Fourth INEX Workshop, Dagstuhl, Germany, November 28–30, 2005*, 2005.
3. K. Hatano, H. Kinutan, M. Watanabe, Y. Mori, M. Yoshikawa, and S. Uemura. Keyword-based XML fragment retrieval: Experimental evaluation based on INEX 2003 relevance assessments. In *Proceedings of the Second International Workshop of the INitiative of the Evaluation of XML Retrieval, INEX 2003, Dagstuhl Castle, Germany, December 15–17, 2003*, pages 81–88, 2004.
4. K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments. Parts 1 and 2. *Information Processing and Management*, 36(6):779–840, 2000.
5. G. Kazai and M. Lalmas. INEX 2005 evaluation metrics. 2005. Available at URL: `http://inex.is.informatik.uni-duisburg.de/2005/`.
6. G. Kazai and M. Lalmas. Notes on what to measure in INEX. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 22–38, Glasgow, UK, 2005.
7. G. Kazai, M. Lalmas, and A. P. de Vries. The overlap problem in content-oriented XML retrieval evaluation. In *Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 72–79, Sheffield, UK, 2004.
8. M. Lalmas. INEX 2005 retrieval task and result submission specification. 2005. Available at URL: `http://inex.is.informatik.uni-duisburg.de/2005/`.
9. J. Pehcevski and J. A. Thom. HiXEval: Highlighting XML retrieval evaluation. In *Pre-Proceedings of the Fourth INEX Workshop, Dagstuhl, Germany, November 28–30, 2005*, 2005.
10. J. Pehcevski, J. A. Thom, and A.-M. Vercoustre. Hybrid XML retrieval: Combining information retrieval and a native XML database. *Information Retrieval*, 8(4):571–600, 2005.
11. A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 21–29, Zurich, Switzerland, 1996.
12. C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004.
13. J. Zobel and A. Moffat. Exploring the similarity space. *ACM SIGIR Forum*, 32(1):18–34, 1998.

SIRIUS: A Lightweight XML Indexing and Approximate Search System at INEX 2005

Eugen Popovici, Gildas Ménier, Pierre-François Marteau

VALORIA Laboratory, University of South-Brittany
BP 573, 56017 Vannes Cedex, France
{Eugen.Popovici, Gildas.Menier Pierre-Francois.Marteau}@univ-ubs.fr

Abstract. This paper reports on SIRIUS, a lightweight indexing and search engine [6] for XML documents. The retrieval approach implemented is document oriented. It involves an approximate matching scheme of the structure and textual content. Instead of managing the matching of whole DOM trees, SIRIUS splits the documents object model in a set of paths. This set is indexed using optimized data structures. In this view, the request is a path-like expression with conditions on the attribute values. In this paper, we present the main functionalities and characteristics of this XML IR system and second we relate on our experience on adapting and using it for the INEX 2005 ad-hoc retrieval task. Finally, we present and analyze the SIRIUS retrieval performance obtained during the INEX 2005 evaluation campaign and show that despite the lightweight characteristics of SIRIUS we obtained quite good precision at low recall values.

1 Introduction

The widespread use of XML in digital libraries, product catalogues, scientific data repositories and across the Web prompted the development of appropriate searching and browsing methods for XML documents. Approximate matching in XML provides the possibility of querying the information acquired by a system having an incomplete or imprecise knowledge about both the structure and the content of the XML documents [14], [15]. In this context, we propose and experiment with a lightweight model for indexing and querying XML documents. We develop a simple querying algebra implemented using fast approximate searching mechanisms for structure and textual content retrieval. In order to evaluate the expected benefits and drawbacks of this new kind of search functionality we propose algorithms and data structures whose principles are detailed hereinafter.

We propose specific data structures dedicated to the indexing and retrieval of information elements embedded within heterogeneous XML data bases. The indexing scheme is well suited to the characterization of various contextual searches, expressed either at a structural level or at an information content level. Search mechanisms are based on context tree matching algorithms that involve a modified Levenshtein editing distance [12] and information fusion heuristics. The implementation that is finally described highlights the mixing of structured information presented as field/value instances and free text elements. Our approach is evaluated experimentally at the INEX 2005 workshop. The results are encouraging and give rise to a number of future enhancements.

The paper is organized as follows. In Section 2 we present the main functionalities and characteristics of the SIRIUS XML IR system. In Section 3 we relate on our experience on adapting and using the system in the INEX 2005 ad-hoc retrieval task. In Section 4 we present the evaluation of the SIRIUS retrieval while participating at INEX 2005 campaign on the VVCAS task. Finally, in Section 5 we summarize our conclusions and propose some futures work perspectives.

2 SIRIUS XML IR System

SIRIUS [6] is a lightweight indexing and search engine for XML documents developed at the VALORIA laboratory of the University of South-Brittany [6][7]. The retrieval approach implemented in SIRIUS is document oriented. It involves an approximate matching scheme of the structure and textual content. Instead of managing the matching of whole DOM trees, SIRIUS splits the documents object model in a set of paths. This set is indexed using optimized data structures. In this view, the request is a path-like expression with conditions on the attribute values. For instance */document(> date "1994")/chapter(= number 3)/John* is a request aiming to extract the documents (written after *94*) with the word John in the chapter number *3*. We designed a matching process that takes into account mismatched errors both on the attributes and on the xml elements. The matching process uses a weighted editing distance on XML paths: this provides an approximate matching scheme able to manage jointly the request on textual content and on document structure. The search scheme is extended with a set of Boolean and IR retrieval operators, and features a set of thesaurus rewriting rules. Recently the system was extended with a specialized set of operators for extracting, indexing and searching heterogeneous sequential and time series data embedded in heterogeneous XML documents [8].

2.1 Indexing Scheme

An XML document is composed of a set of elements with a possible nested structure. Each XML element may be composed of a set of possible nested XML elements, textual pieces of information (TEXT or CDATA), unordered <attribute, value> pairs, or a mixture of such items. XML documents are generally represented as rooted, ordered, and labeled trees in which each node corresponds to an element and each edge represent a parent-child relationship.

XML Context. According to the tree structure, every node n inherits a path $p(n)$ composed with the nodes that link the root to the node n. This path is an ordered sequence of XML elements potentially associated to unordered <attribute, value> pairs $A(n_i)$, that determines the XML context in which the node is occurring. A tree node n, containing textual/mixed information can be decomposed into textual sub-elements. Each string s (or word, lemma, …) of a textual sub-element is also linked to *p(n)*. This XML context characterizes the occurrence of s within the document and can be represented as follows:

$$p(n)=<n_0\,,\, A(n_0)> \; <n_1\,,\, A(n_1)> \;\ldots<n\,,\, A(n_n)> \qquad (1)$$

Index Model. The indexing process involves the creation of an enriched inverted list designed for the management of these XML contexts. For this model, the entries of the inverted lists are of two kinds: textual sub-element *s* of a tree node, and node *n* of the XML tree.
For a sub-element *s* of a node *n*, three pieces of information are attached:

- a link to the URI of the document to retrieve the original document,
- an index specifying the location of the sub-element within the document,
- a link toward its XML context *p(n)*.

For a node *n* of the DOM tree, only the link to the URI of the document and a link to its XML context *p(n)* are required.

2.2 Searching Scheme

Most of the time, for large heterogeneous databases, one cannot assume that the user knows all of the structures – even in the very optimistic case, when all of the structural properties are known. Some straightforward approaches (such as the XPath search scheme [16]) may not be efficient in these cases. As the user cannot be aware of the complete XML structure of the data base due to its heterogeneity, efficient searching should involved exact and approximate search mechanisms.

The main structure used in XML is a tree: It seems acceptable to express a search in term of tree-like requests and approximate matching. The matching tree process involves mainly elastic matching or editing distance [10], [11]. For [10], the complexity of matching two trees T_1 and T_2 is at least $O(|T_1|.|T_2|)$ where $|T_i|$ is the number of nodes in T_i. The complexity is much higher for common subtree search [11]. This complexity is far too high to let these approaches perform well for large heterogeneous database with documents with a high number of elements (as nodes).

We proposed [6], it seems at the same time than the IBM team at Haïfa, to focus on path matching rather than on tree matching – in a similar manner than the XML fragment approach [15]. The request should be expressed as a set of path *p(r)* that is matched with the set of sub-path *p(n)* in the document tree. This breaks the algorithmic complexity and seems to better correspond to the end-user needs: most of the data searches involve a node and its inherited sequence of elements rather than a full tree of elements. This 'low-level' matching only manage subpath similarity search with conditions on the element and attributes matching. This process is used to design a more higher-level request language: a full request is a tree of low-level matching goals (as leafs) with set operators as nodes. These operators are used to merge leaf results. The whole tree is evaluated to provide a set of ranked answers. The operators are classical set operators (intersection, union, difference) or dedicated fuzzy merging processors.

Approximate Path Search. Let *R* be a low-level request, expressed as a path goal p^R, with conditions or constraints to be fulfilled on the attributes and attributes values. We investigate the similarity between a p^R (coding a path with constraints) and a tree T^D of an indexed document *D* as follow:

$$\delta(p^R, T^D) = \underset{i}{Min}\ \delta_L(p^R, p_i^D) \tag{2}$$

where δ_L is a dedicated editing distance (see [12]) and $\{ p_i^D \}$ is the set of path in *D*, starting at the root and leading to the last element of the p^R request – terminal(r).

The complexity is $O(l(p^R).deep(T^D).|\{ p_i^D \}|)$ with $|\{ p_i^D \}|$ the size of the set $\{ p_i^D \}$ (i.e. the number of different path), $l(p)$ the length of the path *p* and $deep(T)$ the deepest level of *T.* This complexity remains acceptable for this application as 99% of the XML documents have fewer than 8 levels and their average depth is 4 [14].

Path Similarity Computation $\delta(p^R, T^D)$. Let p^R be the path for the structural request *R* and $\{p_i^D\}$ the set of root/../terminal(r) paths of the tree associated to an index document *D*.

We designed an editing pseudo-distance [15] using a customised cost matrix to compute the match between a path p_i^D and the request path p^R. This scheme, also known as modified Levenshtein distance, computes a minimal sequence of elementary transformation to get from p_i^D to p^R . The elementary transformations are:

- **Substitution**: a node *n* in p_i^D is replaced by a node *n'* for a cost $C_{subst}(n, n')$. Since a node *n* not only stands for an XML element, but also for attributes or attributes relations, we compute $C_{subst}(n, n')$ as follows:

```
1.n.element <> n'.element : Csubst(n, n') = •
        (full substitution)
2.n.element == n'.element :
    if n'.attCond(n.attributes) is true
       $C_{subst}$(n, n') = 0 (no substitution)
    else
       $C_{subst}$(n, n') = ½ ζ (attributes
substitution)
```

where `attCond` stands for a condition (stated in the request) that should apply to the attributes.

- **Deletion**: a node *n* in p_i^D is deleted for a cost $C_{del}(n)= \zeta$,
- **Insertion**: a node *n* is inserted in p_i^D for a cost $C_{ins}(n)= \zeta$.

For a sequence $Seq(p_i^D, p^R)$ of elementary operations, the global cost $GC(Seq(p_i^D, p^R))$ is computed as the sum of the costs of elementary operations. The Wagner&Fisher algorithm [15] computes the best $Seq(p_i^D, p^R)$ (i.e. minimizes $GC()$ *cost)* with a complexity of $O(length(p_i^D) * length(p^R))$ as stated earlier. Let

$$\delta_L(p^R, p_i^D,) = Min_k\ GC(Seq_k(p^R, p_i^D))\ . \tag{3}$$

The similarity between p^R and p_i^D is computed as follows:

$$\delta(p^R, p_i^D) = 1/(1+ \delta_L (p^R, p_i^D)) . \qquad (4)$$

Given p^R and p_i^D, the value for $\delta(p^R , p_i^D) \rightarrow 0$ when the number of mismatching nodes and attribute conditions between p^R and p_i^D increases. For a perfect match $\delta(p^R , p_i^D) = 1$ all the elements and the conditions on attributes from the request p^R match correspondent XML elements in p_i^D.

2.3 Query Language

Complex requests are built using the low-level request p^R described above and merging operators (boolean or specialized operators). Mainly a complex request is a tree of p^R requests as leafs. Each node supports an operator performing the merging of the descendant results. Currently, the following merging operators are implemented in the system for the low-levels management:

- **or**, **and** : n-booleans or n-set. (**or** p^R $p^{R'}$) provides the set merging the set of solution from p^R and from $p^{R'}$. (**and** p^R $p^{R'}$) provides only the answers belonging to both answer sets.
- **without**: this operator can be used to remove solutions from a set. For instance, (**without** p^R $p^{R'}$) delivers the set of solutions for p^R *minus* the solutions given by $p^{R'}$.
- **seq** is mainly an extension of the p^R request : it merges some of the inverted list to provides a simple sequence management. For instance, (**seq** warning * error) express the search of a sequence of texts items – it also applies to structure.
- **in**, **same** : express a contextual relation (same = the p^R have the same path, in = inside elements with the specified path),
- **+**, **same+** : should be related to the **or** operator. The **or** operator is a simple set merging operator, whereas '+' and 'same+' are dedicated operators that take into account the number of element answered. We used a dedicated TFIDF-like function for this purpose (TFIDF stands for *Term Frequency / Inverse Document Frequency*, see [18]).
- **in+** : add structural matching information to the set of solutions. It performs a weighted linear aggregation between the conditions on structure and the set of solutions.

Let $\arg_i$ be a complex request, or a simple (low level) p^R request. The similarity computation $\delta(arg_{root} , T^D)$ for a complex request arg_{root} and a tree T of document D is performed recursively starting at the leafs:

$$\delta((or \arg_1,...,\arg_n),T^D) = \underset{i}{Max}(\delta(\arg_i,T^D))$$

$$\delta((and \arg_1,...,\arg_n),T^D) = \underset{i}{Min}(\delta(\arg_i,T^D))$$

$$\delta((without \arg_1,\arg_2),T^D) = Max\{0, \delta(\arg_1,T^D) - \delta(\arg_2,T^D)\}$$

$\delta((seq\,\arg_1,...,\arg_n),T^D)=$ *1* if *arg*$_1$,*arg*$_2$, …, *arg*$_n$ occurs in sequence and belong to the same context/leaf, else *0*.

$$\delta((in\ ctx\ \arg_1,...,\arg_n),T^D)=\underset{i}{Min}(\delta(ctx/\arg_i,T^D))$$

where *ctx* is an XML context (a path) and *ctx/arg*$_i$ the path made in concatenating *ctx* and *arg*$_i$.

$$\delta((in+ctx\ \arg_1,...,\arg_n),T^D)=\beta\delta(ctx/\arg_i,T^D)+(1-\beta)\delta(\arg_i,T^D)$$

where *ctx* and *ctx/arg*$_i$ as above, and β in [0..1] used to emphasise the importance of the structural matching,.

$$\delta((same\ \arg_1,...,\arg_n),T^D)=\underset{ctx^D}{Max}(\delta(ctx^D/\arg_i,T^D))$$

with ctx^D a path of T^D.

$$\delta((+\arg_1,...,\arg_n),T^D)=\tau\cdot\sum_i\lambda_i(\delta(\arg_i,T^D))$$

where λ_i is a weighting factor specifying the discriminating power of *arg*$_i$*:* $\lambda_i = 1 - log(\ (1+N_D(arg_i)\ )\ /\ (1+N_D)\ )$, where $N_D(arg_i)$ is the number of documents in which *arg*$_i$ is occuring, N_D the total number of documents in the collection*,* and τ a normalization constant $\tau = 1\ /\ \Sigma_k(\ \lambda_k\ )$.

The *same+* operator is computed in a similar way:

$$\delta((same+\arg_1,...,\arg_n),T^D)=\underset{ctx^D}{Max}\left\{\tau\cdot\sum_i\lambda_i(\delta(ctx^D/\arg_i,T^D)\right\}$$

3 INEX 2005 Experience

The retrieval task we are addressing at INEX 2005 is defined as the ad-hoc retrieval of XML elements. This involves the searching of a static set of documents using a new set of topics [1]. We will further present several characteristics of the test collection and of the CAS topics used in INEX 2005 ad-hoc task. Next we will present how we tuned the SIRIUS retrieval approach for the VVCAS task.

3.1 INEX ad-hoc collection

The INEX 2005 document collection contains over 16,800 articles from 24 IEEE Computer Society journals, covering the period of 1995-2004 and totalling about 750 megabytes in size in its canonical form. The collection contains 141 different tag-names[1] composing 7948 unique XML contexts by ignoring the attributes and the attributes values. The maximum length of an index path is 20, while the average

[1] We calculate the statistics from the viewpoint of the retrieval system. That is, we use the XML tag equivalence classes (section 3.5). Also, the XML contexts associated to empty elements or containing only stop words do not count in our statistics.

length (calculated over the unique indexed contexts) is 8. The distribution of elements is heavily skewed towards short elements, such as italics [5].

3.2 INEX 2005 CAS Topics

CAS queries are topic statements that contain explicit references to the XML structure, and explicitly specify the contexts of the user's interest (e.g. target elements) and/or the context of certain search concepts (e.g. containment conditions).

A CAS topic has several parts expressing the same information need: <*narrative*>, <*description*>, <*title*>, <*castitle*> and <*parent*> [4]. An example of the <*castitle*> part of a CAS topic expresses in NEXI language [3] in shown in Fig. 1.

```
//article[ about(.//bb, Baeza-Yates) and
          about(.//sec , string matching)]//sec[about(., approximate algorithm)]
```

Fig. 1. CAS topic 280.

This year, a total of 47 CAS topics were selected of which 30 are subordinate topics of the form //A[B][2], 6 are independent/standalone topics of the form //A[B] and 11 are complex topics of the form //A[B]//C[D][3] constructed using the subordinate topics.

We make here some observations on the selected set of topics that are used for retrieval in the INEX 2005 CAS task.

First, no constraint on attributes or attributes values is made. This may be explained by the fact that the collection is considered to have no attribute or attribute value with practical interest for a real end user – see "A Note on Attributes" in [3].

Second, even if the collection contains paths with an average length ranging between 6 and 8 elements; a reduced number of elements (2, maximum 3) are employed by the users to express paths for the structural constraints.

Third, 21 of the total of 47 CAS topics, 9 of the 17 complex and standalone CAS topics, and 7 of the 10 assessed topics used phrase searching.

3.3 Translating INEX 2005 CAS topics to SIRIUS Query Language

We use automatic transformation of the INEX 2005 CAS topics expressed in NEXI [3] to SIRIUS [6] recursive query language.

We have two types of CAS queries: simple queries of the form //A[B] and complex queries of the form //A[B]//C[D].

For the simple type queries, the translation process is straight forward by using the SIRIUS operators: *and, or, in+, same+* and *seq* (Fig. 2).

//article[about(.//bb, Baeza-Yates)	(in+ [/article/bb/] (same+ (seq Baeza Yates)))

Fig. 2. Translating CAS topic 277 to SIRIUS query language.

[2] //A[B] = returns A tags about B [4].

[3] //A[B]//C[D] = returns C descendants of A where A is about B and C is about D [4].

For translating complex queries of the form //A[B]//C[D], we introduce a new *filter* operator in the SIRIUS query language (Fig. 3).

```
( filter  ( and  ( in+   [/article/bb/]   ( same+   ( seq  Baeza Yates ) ) )
                 ( in+   [/article/sec/]   ( same+   string  matching ) )
          )
          ( in+  [/article/sec/]  ( same+   approximate  algorithm ) )
)
```

Fig. 3. CAS topic 280 (Fig. 1) translated to SIRIUS query language.

The *filter* operator receives as arguments two sets arg_1, arg_2 of weighted matching results (r_i, w_i). It retains from the second set all the descendants (r_j, w'_j) of the results occurring in the first set. For this operator, the new weight w'_j associated with a selected result is calculated as the arithmetic average of the initial weight w_j of the corresponding result in the second set and the maximum weight $ArgMax_i(w_i)$ found in all it's ancestors and their descendants from the first set of results. This operator implies solving element containment relationships that were not supported by the SIRIUS index model. However, for all the *11* complex CAS topics, the ancestor *//A* specified in the structural path is the *article* element. Therefore, only document *D* level containment conditions are checked in the current implementation of the operator.

3.4 Indexing the INEX 2005 Ad-Hoc Collection

The collection is pre-processed by removing the *volume.xml* files and transforming all the XML files in their canonical form[4]. At indexing time, the most frequent words are eliminated using a stop list. The terms are stemmed using the Porter algorithm [17]. We index only *ALPHANUMERIC* words as defined in [3] (like *iso-8601*). We did not index numbers, the attributes, the attributes values, and empty XML elements. This allowed important performance gains both in indexing and querying time as well as disk space savings. The index model (section 2) enhanced with a *node labelling scheme* based on pre-ordered containment intervals was implemented using BTrees structures from Berkeley DB[5] library. The index size has about twice and a half the initial database size. The indexing time on a PIV 2.4GH processor with 1.5GB of RAM for the inex-1.6 ad-hoc collection in canonical form (~750MB) was about 50min with one more hour post-processing phase for disk optimizations.

3.5 Structure Approximate Match for INEX

Structural Equivalence Classes. We create structural equivalence classes for the tags defined as interchangeable in a request: *Paragraphs*, *Sections*, *Lists*, and *Headings* in conformance with [4]:

[4] Canonical XML Processor {http://www.elcel.com/products/xmlcanon.html}

[5] http://www.sleepycat.com/

Weighting Scheme for Modelling Ancestor-Descendant Relationships. NEXI language [3] specifies a path through the XML tree as a sequence of nodes. Furthermore, the only relationship allowed between nodes in a path is the descendant relation. Therefore the XML path expressed in the request is interpreted as a *subsequence* of an indexed path, where a subsequence need not consist of contiguous nodes.

This is not suited for the weighting scheme allowing (slight) mismatch errors between the structural query and the indexed XML paths implemented in SIRIUS [6]. The ancestor-descendant relationship is penalized by the SIRIUS weighting scheme relative to a parent-child relationship. Therefore we relax the weights of the path editing distance in order to allow node deletions in the indexed paths without any penalty $C_{del}(n)=0$. To illustrate this mechanism we show in Fig. 4 the distances between the path requested in topic 277 (Fig. 2) searching works citing Baeza-Yates and several indexed path retrieved by SIRIUS using the new weighting scheme..

δ (/article/bm/bib/bibl/bb/au/snm, //article//bb) = 0
δ (/article/bm/app/bib/bibl/bb/au/snm, //article//bb) = 0
δ (/article/fm/au/snm, //article//bb) = 1

Fig. 4. Example of distances between the indexed path p_i^D and the request path p^R.

In the first two cases the request path p^R is a subsequence of the retrieved paths p_i^D and therefore the editing distance is 0 independently of the length of the two index paths. In the last case, where Baeza-Yates is the author of the article, the editing distance is *1* highlighting the mismatch of the requested *bb* node from the indexed path.

The weighting scheme models an end user having precise but incomplete information about the xml tags of the indexed collection and about their ancestor-descendant relationships. It takes into account the number of matched XML nodes from the request and their order of occurrence. It heavily penalized any mismatch relatively to the information provided by the user but it is forgiving with mismatches/extra information extracted from the indexed paths.

3.6 SIRIUS VVCAS Runs.

CAS queries are topic statements that contain two kinds of structural constraints: where to look (support elements), and what elements to return (target elements). When implementing a VVCAS strategy, the structural constraints in both the target elements and the support elements are interpreted as vague [1]. We submitted 2 official and 4 additional runs within the CAS ad-hoc task using the VVCAS strategy and automatic query translation (Table 1).

Table 1. Sirius VVCAS runs (1,2 official runs, 3, 4, 5, 6 additional runs).

ID	Run
1.	VVCAS_contentWeight08_structureWeight02
2.	VVCAS_contentWeight05_structureWeight05
3.	VVCAS_contentWeight05_structureWeight05_unofficial
4.	VVCAS_contentWeight10_structureWeight00_ unofficial
5.	VVCAS_contentWeight05_structureWeight05_SAMEPLUS_unofficial
6.	VVCAS_contentWeight10_structureWeight00_SAMEPLUS_unofficial

In the official 1, 2 and additional 3, 4 runs we use the same retrieval approach, namely, strict sequence search, modified editing distance on XML paths for matching the structural constraints, weighted linear aggregation for content and structure matching scores. Additional run 3 corrects a bug found in one limit cases of our official runs. The difference observed on the evaluation curves between the official run 2 and the additional run 3 is minimal (see Fig. 5, Fig. 6, Tables 2 and 3). For additional runs 5 and 6 we use an approximate matching operator instead of a strict sequence search.

Strict Sequence Matching Runs: we used a strict *seq* operator – where *strict* stands for words appearing in sequence in the textual content (ignoring the stop list words) of the same XML node. Therefore searching for the sequence "*Ricardo Baeza-Yates*", in our implementation will point to the 'name' element of the first example (Fig. 2), but will completely ignore the sequence from the second one.

```
<reviewer>
  <name>Ricardo Baeza-Yates</name>
  …
</reviewer>
```

```
<au sequence="additional">
    <fnm>Ricardo</fnm>
    <snm>Baeza-Yates</snm>
    …
</au>
```

Fig. 5. Ex. of sequence allowed in the same XML element and ignored in adjacent elements.

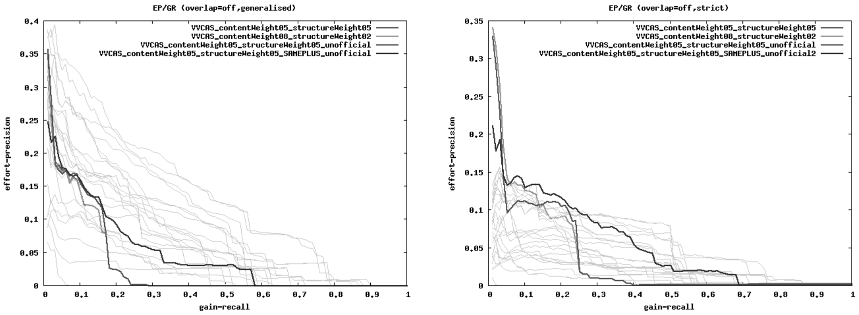
Flexible Matching Runs: the additional runs 4 and 6 implements a flexible sequence search based on the *same+* operator. These runs rank as best results the XML elements that contain all the researched terms without taking into account their order of occurrence. XML elements that contain part of the research terms are also retrieved and ranked based on the number of different researched terms contained, according to the weighting (IDF) of each occurring term.

4 SIRIUS Retrieval Performance Evaluation

XML IR systems relevance in XML retrieval tasks is evaluated along two dimensions: *exhaustivity* and *specificity*. An element is exhaustive if the topic of request is exhaustively discussed within that element, whereas an element is specific if the element is highly focussed on the topic [1].

4.1 INEX 2005 Evaluation Results

We report here the system-oriented and user-oriented official INEX 2005 evaluation measures: the effort-precision/gain-recall (*ep/gr*) metric (Fig. 6, Table 3), Extended Q and R Metric (Table 2.)**,** respectively the normalized extended cumulated gain (*nxCG*) metric (Fig. 7) for all the submitted VVCAS runs. Details of the evaluation metrics can be found in [2].



a) b)

Fig. 6. SIRIUS effort-precision/gain-recall (*ep/gr*) curves: a) EP/GR(overlap=off, , quant.=generalized), b) EP/GR (overlap=off, quant.=strict)

Effort-precision (EP) at a given gain-recall (GR) value measure the relative effort (number of visited ranks) that the user is required to spend when scanning a system's output compared to the effort an ideal ranking would take in order to reach a given level of cumulated gain. [2].

Table 2. Extended Q and R metrics.

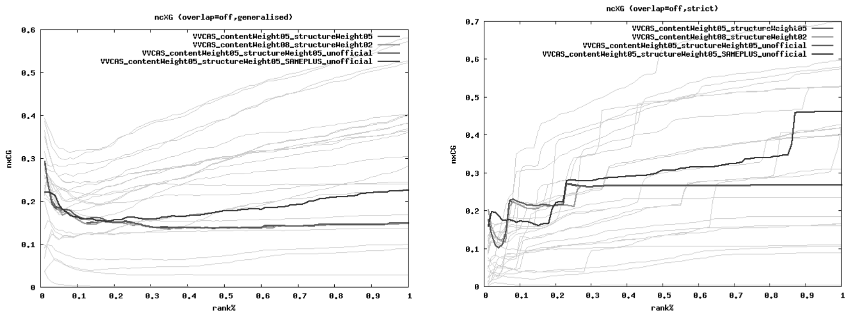
Run	**Q(overlap=off)**		**R(overlap=off)**	
ID	**Strict**	**Gen**	**Strict**	**Gen**
1.	0.009	0.025	0.039	0.084
2.	0.009	0.025	0.039	0.084
3.	0.009	0.026	0.040	0.085
4.	0.008	0.023	0.040	0.083
5.	0.014	0.045	0.063	0.125
6.	0.013	0.042	0.063	0.113

The Q-and-R-measures are modified normalised cumulated gain measures which employ a bonus gain function that directly incorporates the rank position in the measured gain [2].

Table 3. Effort-precision measures at different levels of gain-recall.

Run	EP v.s. GR (Overlap=off, Quant=strict)					
ID	**0.01/15**	**0.02/30**	**0.1/150**	**0.2/300**	**MAep**	**iMAep**
3.	0.3302	0.291	0.1123	0.1051	0.0086	0.0338
4.	0.3165	0.285	0.0971	0.092	0.0082	0.0314
5.	0.2114	0.1786	0.1298	0.1165	0.0125	0.0523
6.	0.2011	0.1673	0.125	0.1148	0.0122	0.0491
	EP v.s. GR (Overlap=off, Quant=gen)					
	0.01/15	**0.02/30**	**0.1/150**	**0.2/300**	**MAep**	**iMAep**
3.	0.3483	0.2813	0.1549	0.0253	0.0218	0.03
4.	0.275	0.228	0.1346	0.0245	0.0191	0.0253
5.	0.2474	0.217	0.1598	0.0925	0.0351	0.0465
6.	0.2306	0.213	0.145	0.093	0.0326	0.0428

For a given rank *i*, the value of *nxCG[i]* reflects the relative gain the user accumulated up to that rank, compared to the gain he/she could have attained if the system would have produced the optimum best ranking [2].



a) b)

Fig. 7. The user-oriented measure of normalized extended cumulated gain (*nxCG*): a) nxCG(overlap=off, quant.=generalized), b) nxCG(overlap=off, quant.=strict)

4.2 Analysis of SIRIUS Retrieval Performance

Using the structural information. The objective of our study was to determine to what extent the structural hints should be taken into account when implementing a VVCAS strategy. For the two official runs (1, 2) we assigned different weights to merge the content and structure relevance scores, i.e.(0.8, 0.2) and (0.5, 0.5): The coefficients used were not discriminate enough to impose a different ranking of the final results. Therefore, we conducted tests completely discarding the structural information from the CAS topics, i.e. using (1.0, 0.0) coefficients (runs 4, 6).

The runs with (0.5, 0.5) weight for content and structure (run 3, 5) outperform in average the ones based only on content matching (run 4, 6)(see Fig. 6, Fig. 7 and Table 3).This is true for all of the INEX 2005 evaluation measures (official and additional) and this independently of the quantization function used. This indicates (usual disclaimers apply) that the structural hints, and jointly, the modified editing distance on the XML paths increases the system retrieval performances (the gain varies from 2% up to 7% on EP measure at a fixed GR value)

Sequence Search Strategy. SIRIUS official VVCAS runs have a high effort/precision for low values of gain/recall (Fig. 6, Table 3). This behaviour is due to the fact that the runs implements strict constraints for phrase searching and the topic set was rich (7 among the 10 assessed topics used sequence search) in this kind of hints. The restrictive interpretation of the *seq* operator improved the system precision for the first ranked results (10% to 12%). The SIRIUS VVCAS official runs were ranked several times in the first top ten runs for the first 10-25 retrieved results. This fact is important because these are the most probably to be browsed by an end user. In the same time, this behaviour has penalized the system overall quality performance. One explanation could be that the system stops to return elements when running out of good answers – i.e. we implement a strict sequence search. This has important implications, as the system is not increasing its information gain until reaching the limit of 1500 returned answers by returning imprecise/less perfect results (Fig. 7 – see the lower curves). This hypothesis is also supported by the system behaviour relatively to the strict and generalized quantization functions. Our runs were better ranked by all the official evaluation measures when a strict quantization function was used (Fig. 6b, Fig. 7b). This indicates that the retrieved results are both fully specific and highly exhaustive.

We submitted and evaluated two additional runs (5, 6) allowing a flexible sequential search as described in section 3.6. The superior curve of the additional run 5 based on the *same+* operator in figures (Fig. 6, Fig. 7) shows an important improvement. We loose a small amount in the system precision for the first ranked results, but we obtain an obviously improved overall effort-precision/gain-recall curve (Fig. 7b). Table 3 shows a gain that varies from 0.5% up to 1.8% on the MAep and iMAep measures). We could further improve these results by defining a new operator combining *same+* ranking with *seq* ranking strategies.

4.3 Limits of Our Approach

Returning small XML elements. XML elements containing a very limited amount of text (less than 3-6 characters) either possibly highly relevant as *atl, tig* (article titles), *st* (section titles) or meaningless to an end user as *it* (italicized text) or *sub* (subscript), are frequent in the INEX ad-hoc collection [5].

```
<p align="left" ind="none">
    The  <it>query optimizer</it>, the query engine's central component, determines the
    best service execution plan based on QoWS, service ratings, and matching degrees.
</p>
```

Fig. 8. Mixed XML element content with italicized text (i.e. one of the SIRIUS ‘relevant‘ answer for topic 244)

Elements of such small size are unlikely to satisfy the information need of a topic. Indeed, in the INEX 2002 and 2003 longer elements were preferred by the assessors [5]. This is not beneficial to our approach as we consider the XML element as the basic and implicitly valid unit of retrieval, regardless of its size. With this assumption, we return the *<it>query optimizer</it>* element (Fig. 8) as a relevant answer for topic 244.

Returning overlapping elements: The highly mixed nature of the INEX ad-hoc collection may lead to cases where nested/overlapping XML elements could be returned as valid results by our approach. For instance, the *p* (paragraph) and the *it* (italicized text) elements of the excerpt from Fig. 8 will be retrieved by a request aiming to extract relevant elements for the "*query*" term. Our official runs (1, 2) have a *set-overlap*[6] ranging between 0.002 – 0.004 for the first 20 retrieved results and 0.014-0.015 at 1500 results.

5 Conclusions

We evaluated the retrieval performances of a lightweight XML indexing and approximate search scheme currently implemented in the SIRIUS XML IR system [6], [7], [8]. At INEX 2005, SIRIUS retrieves relevant XML elements by approximate matching both the content and the structure of the XML documents. A modified weighted editing distance on XML paths is used to approximately match the documents structure while strict and fuzzy searching based on the IDF of the researched terms are used for content matching.

Our experiments show that taking into account the structural constraints improves the retrieval performances of the system and jointly shows the effectiveness of the proposed weighted editing distance on XML paths for this task. They also show that the approximate search inside XML elements implemented using our *same+* operator improve greatly the overall performance of the ranking, compared to a strict sequence search (*seq* operator), except for low recall values. The complementarities of the two operators call for the design of a new matching operator based on their combination to further improve the retrieval performance of our system.

While designing our lightweight indexing and XML approximate search system we have put forward the performance and the implementation simplicity. SIRIUS structural match is well adapted for managing mismatches in writing constraints on XML paths involving complex conditions on attributes and attributes values [6]. Unfortunately, this was not experimented in INEX 2005 campaign. SIRIUS was designed to retrieve relevant XML documents by highlighting and maintaining the relevant fragments in the document order (approach explored by the INEX CO.FetchBrowse task). For this year, at the VVCAS task we evaluated only a subset of its functionalities and proved its ability of retrieving relevant XML elements. Still, we obtained average and good quality results in the range of the 25 first ranked

[6] Set overlap measures the percentage of elements that either contain or are contained by at least one or other element in the set.

answers, which is quite encouraging since first ranked elements are the ones end users will most probably browse.

Future research work will include: the semantic enrichment of the requests (at xml tag and query term levels) in order to improve the overall recall of the system, index models better suited for the approximate search scheme, and new matching operators.

Acknowledgements

This work was partially supported by the ACIMD – ReMIX French grant (Reconfigurable Memory for Indexing Huge Amount of Data).

References

1. Lalmas M., INEX 2005 Retrieval Task and Result Submission Specification, June 20, 2005.
2. Kazai G., Lalmas M., INEX 2005 Evaluation Metrics, November 2005, INEX 2005.
3. Trotman A., & Sigurbjörnsson B. (2004). Narrowed Extended XPath I (NEXI) In Proceedings of the INEX 2004 Workshop, (pp. 16-40).
4. Sigurbjörnsson B., Trotman A., Geva S.,Lalmas M., Larsen B., Malik S., INEX 2005 Guidelines for Topic Development, INEX 2005.
5. Kamps J., de Rijke M., and Sigurbjörnsson B., The Importance of Length Normalization for XML Retrieval, Information Retrieval. Volume: 8. Issue: 4. Pages: 631-654. 2005.
6. Ménier G., Marteau P.F., Information retrieval in heterogeneous XML knowledge bases, The 9th International Conference on Information Processing and Magement of Uncertainty in Knowledge-Based Systems, 1-5 July 2002, Annecy, France.
7. Ménier G., Marteau P.F., PARTAGE: Software prototype for dynamic management of documents and data, ICSSEA, 29 Nov.- 1 Dec. 2005, Paris.
8. Popovici E., Marteau P.F., Ménier G., "Information Retrieval of Sequential Data in Heterogeneous XML Databases", AMR 2005, 28-29 July 2005, Glasgow.
9. Tai, K.C, “The tree to tree correction problem”, J.ACM, 26(3):422-433, (1979)
10. Wang T.L.J, Shapiro B., Shasha D., Zhang K., Currey K.M., “An algorithm for finding the largest approximately common substructures of two trees”, In J. IEEE Pattern Analysis and Machine Intelligence, vol.20, N°8, August (1998)
11. Levenshtein A., “Binary Codes Capable of Correcting Deletions, Insertions and Reversals”, Sov.Phy. Dohl. Vol.10, P.707-710, (1966)
12. Wagner R., Fisher M., “The String-to-String Correction Problem”, Journal of the Association for Computing Machinery, Vol.12, No.1, p.168-173, (1974)
13. Mignet L.,Barbosa D.,Veltri P., The XML Web: A First Study (2003),WWW2003, May 20-24, Budapest, Hungary
14. Carmel D., Maarek Y. S., Mandelbrod M., Mass Y. and Soffer A., Searching XML documents via XML fragments, SIGIR 2003, Toronto, Canada pp. 151-158.
15. Fuhr N., Groβjohann K., XIRQL: An XML query language based on information retrieval concepts, (TOIS), v.22 n.2, p.313-356, April 2004
16. Clark J., DeRose S., XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999, http://www.w3.org/TR/xpath.html, (1999)
17. Porter, M.F., 1980, An algorithm for suffix stripping, Program, 14(3):130-137
18. Salton G. and Buckeley C., Term-weighting approaches in automatic text retrieval, Information Processing and Management, 24, 513-523, 1988.

An Evaluation of Relevance Ranking Methods for XML Using Both Document and Query Structures

Sihem Amer-Yahia[1], Kenji Hatano[2], Jayavel Shanmugasundaram[3], and Divesh Srivastava[1]

[1] AT&T Labs–Research, Florham Park, NJ 07932, USA
(sihem,divesh)@research.att.com
[2] Nara Institute of Science and Technology (NAIST), Nara 630-0192, Japan
hatano@is.naist.jp
[3] Cornell University, Ithaca, NY 14853, USA
jai@cs.cornell.edu

Abstract. XML queries, in particular, NEXI queries, combine conditions on both content and structure and return document fragments satisfying query conditions. Thus, several ranking methods have been proposed to extend the well-established content-based scoring in the vector space model in order to account for input document structure to score answers to XML queries. Such methods use document structure to propagate answer scores along the XML hierarchy [6], to apply length normalization between query paths and data paths [3], to compute term weights based on element tags [5] or, to account for overlapping elements [4]. In this paper, we show that accounting for query conditions on structure *in addition to* conditions on content and *combining them with* data-based scoring improves the relevance of answers to XML queries. Our ideas are implemented in the STRUX system that is used as a platform for experimenting with XML scoring on top of GALATEX, an open-source implementation of XQuery Full-Text [1, 8], an upcoming W3C standard for extending XPath/XQuery with full-text search primitives.

The score of an answer in STRUX combines a data score ($\mathtt{IPF_d}$) and a query score ($\mathtt{IPF_q}$). $\mathtt{IPF_d}$ is inspired from [7] and $\mathtt{IPF_q}$ is inspired from the *path scoring* method proposed in [2]. In order to compute $\mathtt{IPF_d}$ (resp., $\mathtt{IPF_q}$), path scoring assumes independence between paths in the data (resp., in the query), and combines the scores of individual paths. For example, given the query:
//article//sec[about(., intelligent transport systems)], the score of an answer **s** is defined as:
$\mathtt{score(s)} = \Sigma_{\mathtt{t}\in\mathtt{T}}\ (\mathtt{tf_d(s,t)} * \mathtt{ipf_d(s,t)} * \mathtt{tf_q(t)} * \mathtt{ipf_q(t)})/\mathtt{length(s)}$ where:
$\mathtt{T}$ is the set of keyword terms:: intelligent, transport and systems.
$\mathtt{tf_d(s,t)}$ is the number of occurrences of term $\mathtt{t}$ in $\mathtt{s}$;
$\mathtt{tf_q(t)}$ is the number of occurrences of term $\mathtt{t}$ in the query;
$\mathtt{length(s)}$ is the total number of words in $\mathtt{s}$ and,
$\mathtt{ipf_d}$ and $\mathtt{ipf_q}$ are defined as follows:
$\mathtt{ipf_d(s,t)} = 1 + \mathtt{log_2}$*(number of answers satisfying s's data path/number of such answers containing t).*
Elements sharing the same data path in input documents have the same value for $\mathtt{ipf_d}$ for a given term.
$\mathtt{ipf_q(t)} = 1 + \mathtt{log_2}$*(number of answers satisfying the query path* (**//article//sec**)*/number of such answers containing t).*

All query answers share the same $\mathtt{ipf_q(t)}$.
In summary, our technical contributions are:

- We define the score of an answer to a NEXI query as a combination of $\mathtt{IPF_d}$ (as in [7]) and $\mathtt{IPF_q}$ (as in [2]), two scoring methods that account for paths in input document and for query conditions on structure to rank query answers.
- We implement our scoring method in STRUX and run extensive experiments on INEX 2005 datasets and queries, that show that combining $\mathtt{IPF_d}$ and $\mathtt{IPF_q}$ to compute the score of answers to INEX queries yields more effective results than just using $\mathtt{IPF_d}$ or $\mathtt{IPF_q}$ by itself.

Our scoring methods are a basis for considering query relaxation on structure which accounts for XML heterogeneity and enables to return answers that do not match exactly query conditions on structure, suitably ranked by their relevance to the input query. In this case, $\mathtt{ipf_q(t)}$ would need to be adapted to each query answer based on the relaxed query satisfied by the answer. We plan to explore this direction in the future.

References

1. S. Amer-Yahia, C. Botev, J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. WWW 2004.
2. S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, D. Toman. Content and Structure Scoring for XML. VLDB 2005.
3. D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, A. Soffer. Searching XML Documents via XML Fragments. SIGIR 2003.
4. C. L. A. Clarke. Controlling Overlap in Content-Oriented XML Retrieval. SIGIR 2005.
5. S. Cohen, J. Mamou. Y. Kanza, Y. Sagiv. XSEarch: A Semantic Search Engine for XML. VLDB 2003.
6. N. Fuhr, K. Grossjohann. XIRQL: An Extension of XQL for Information Retrieval. SIGIR Workshop on XML and Information Retrieval 2000.
7. T. Grabs, H. Schek ETH Zürich at INEX: Flexible Information Retrieval from XML with PowerDB-XML. INEX Workshop 2002.
8. The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Full-Text. Working draft. http://www.w3.org/TR/xquery-full-text/

Machine Learning Ranking and INEX'05

Jean-Noël Vittaut and Patrick Gallinari

Laboratoire d'Informatique de Paris 6
8, rue du Capitaine Scott, F-75015 Paris, France
{vittaut, gallinari}@poleia.lip6.fr

Abstract. We present a Machine Learning based ranking model which can automatically learn its parameters using a training set of annotated examples composed of queries and relevance judgments on a subset of the document elements. Our model improves the performance of a baseline Information Retrieval system by optimizing a ranking loss criterion and combining scores computed from doxels and from their local structural context. We analyze the performance of our algorithm on CO-Focussed and CO-Thourough tasks and compare it to the baseline model which is an adaptation of Okapi to Structured Information Retrieval.

1 Introduction

Different studies and developments have been recently carried out on ranking algorithms in the machine learning community. In the field of textual documents, they have been successfully used to combine features or preferences relations in tasks such as meta search [1] [2] [3], passage classification, automatic summarization [4] and recently for the combination of different sources of evidence in Information Retrieval (IR) [5]. One of the challenges of this paradigm is to reduce the complexity of the algorithms which is in the general case quadratic in the number of samples. This is why most real data applications of ranking are based on two-classes problems. Nevertheless, under some conditions, fast rates of convergence are achieved with this class of methods [6].

Ranking algorithms work by combining features which characterize the data elements to be ranked. In our case, these features will depend on the doxel itself and on its structural context. Ranking algorithms will learn to combine these different features in an optimal way according to a specific loss function using a set of examples. It is hoped that ranking algorithms may help to improve the performance of existing techniques.

The paper is organized as follows, in section 2 we present the ranking model, in section 3 we show how we adapted it to CO-Focussed and CO-Thorough tasks. In section 4 we comment the results reached by our model and compare it to a baseline Okapi method adapted for SIR.

2 Ranking model

We present in this section a general model of ranking which can be adapted to IR or SIR. The idea of the ranking algorithms proposed in the machine learning community is to learn a total order on a set $\mathcal{X}$, which allows to compare any element pair in this set. Given this total order, we are able to order any subset of $\mathcal{X}$ in a ranking list. For instance in IR, $\mathcal{X}$ can be the set of couples (document, query), and the total order is the natural order on the document scores.

As for any machine learning technique, one needs a training set of labeled examples in order to learn how to rank. This training set will consist in ordered pairs of examples. This will provide a partial order on the elements of $\mathcal{X}$. The ranking algorithm will use this information to learn a total order on the elements of $\mathcal{X}$ and after that will allow to rank new elements. For plain IR, the partial ordering may be provided by human assessments on different documents for a given query.

2.1 Notations

Let $\mathcal{X}$ be a set of elements with a partial order $\prec$ defined on it. This means that some of the element pairs in $\mathcal{X}$ may be compared according to the $\prec$ relation. For Structured Information retrieval $\mathcal{X}$ will be the set of couples (doxel, query) for all doxels and queries in the document collection. This set is partially ordered according to the existing relevance judgments for each query.

2.2 Ranking

Let f be a function from $\mathcal{X}$ to the set of real numbers. We can associate a total order to f such that:

$$x \prec x' \Leftrightarrow f(x) < f(x') \ . \tag{1}$$

Clearly, learning the f function is the same as learning the total order.

An element of $\mathcal{X}$ will be represented by a real vector of features:

$$x = (x_1, x_2, ..., x_d).$$

In our case, the features will be local scores computed on different contextual elements of a doxel. In the following, f will be a linear combination of x's features:

$$f_\omega(x) = \sum_{j=1}^{d} \omega_j x_j \tag{2}$$

where $\omega = (\omega_1, \omega_2, ..., \omega_d)$ are the parameters of the combination to be learned.

Ranking loss. f_ω is said to respect $x \prec x'$ if $f_\omega(x) < f_\omega(x')$. In this case, couple (x, x') is said to be well ordered by f_ω. The ranking loss [3] measures how much f_ω respects $\prec$.

By definition, the ranking loss measures the number of mis-ordered couples in $\mathcal{X}^2$:

$$R(\mathcal{X}, \omega) = \sum_{\substack{(x,x') \in \mathcal{X}^2 \\ x \prec x'}} \chi(x, x') \tag{3}$$

where $\chi(x, x') = 1$ if $f_\omega(x) > f_\omega(x')$ and 0 otherwise.

Ranking aims at learning ω for minimizing (3).

Exponential loss. In practice, this expression is not very useful since χ is not differentiable, ranking algorithms use to optimize another loss criterion called the exponential loss:

$$R_e(\mathcal{X},\omega) = \sum_{\substack{(x,x')\in\mathcal{X}^2\\ x\prec x'}} e^{f_\omega(x)-f_\omega(x')}. \tag{4}$$

If is straightforward that $R(\mathcal{X},\omega) \leq R_e(\mathcal{X},\omega)$. (4) is differentiable and convex, and then can be minimized using standard optimization techniques. Minimizing (4) will allow to minimize $R(\mathcal{X},\omega)$.

We can compute a gradient descent. The components of the gradient of R_e are:

$$\frac{\partial R_e}{\partial \omega_k}(\mathcal{X},\omega) = \sum_{\substack{(x,x')\in\mathcal{X}^2\\ x\prec x'}} (x_k - x'_k)e^{f_\omega(x)-f_\omega(x')}. \tag{5}$$

With no more hypothesis, the computation of (5) is in $O(|\mathcal{X}|^2)$.

3 Application to CO tasks

3.1 Definitions

Let denote $\mathcal{D}$ is the set of doxels for all the documents in the collection and $\mathcal{Q}$ the set of CO queries. $\mathcal{X} = \mathcal{Q}\times\mathcal{D}$ is the set of elements we want to order.

We suppose that there exists a partial order $\prec$ on $\mathcal{X} = \mathcal{Q}\times\mathcal{D}$, this partial order will reflect for some queries, the evidence we have about preferences between doxels provided via manual assessments. Note that these relevance assessments are only needed for a few queries and doxels in the collection. We consider here the task which consists in producing a ranked list of doxels which answer the query $q \in \mathcal{Q}$. For that, we will train the ranking model to learn a total strict order on $\mathcal{X}$.

3.2 Vector Representation

Each element $x \in \mathcal{X}$ is represented by a vector $(x_1, x_2, ..., x_d)$ were x_i represents some feature which could be useful to order elements of $\mathcal{X}$. Let denote $\mathcal{L}$ the set of doxel types, which are defined according to the DTD of the document collection: article, abstract, sections, paragraphs, lists...

We used the following combination:

$$f_w(x) = \omega_1^l + \omega_2^l Okapi(x) + \omega_3^l Okapi(parent(x)) + \omega_4^l Okapi(document(x))$$

where l is the node type of x and *Okapi* is the SIR adapted Okapi model [7] described in [8]. This adaptation consists in using doxels rather than documents for computing the term frequencies, and using as normalization factor for each doxel, the mean size of the doxels with the same node type.

This combination take into account the information provided by the context of the doxel and the structural information given by the node type of the doxel.

This combination leads to the following vector representation:

$$x = \Big((x_1^{l_1}, x_2^{l_1}, x_3^{l_1}, x_4^{l_1}), (x_1^{l_2}, x_2^{l_2}, x_3^{l_2}, x_4^{l_2}), ..., (x_1^{l_{|\mathcal{L}|}}, x_2^{l_{|\mathcal{L}|}}, x_3^{l_{|\mathcal{L}|}}, x_4^{l_{|\mathcal{L}|}}) \Big)$$

where $|\mathcal{L}|$ is the number of different doxel types in the collection.

In the above expression all vector components of the form $(x_1^{l_i}, x_2^{l_i}, x_3^{l_i}, x_4^{l_i})$ are equal to $(0, 0, 0, 0)$ except for one where l_i is the doxel type of x which is equal to $(1, Okapi(x), Okapi(parent(x)), Okapi(document(x)))$.

3.3 Reduction of complexity

In this section, we use some properties of SIR in order to decrease the complexity of the computation of (4) and (5).

Queries. Comparing elements from different queries has no sense. We can define a partition $\mathcal{X} = \bigcup_{q \in \mathcal{Q}} \mathcal{X}_q$, where

$$\mathcal{X}_q = \{x = (d, q') \in \mathcal{X} / q' = q\}$$

and we can rewrite (4):

$$R_e(\mathcal{X}, \omega) = \sum_{q \in \mathcal{Q}} \left\{ \sum_{\substack{(x,x') \in \mathcal{X}_q \times \mathcal{X}_q \\ x \prec x'}} e^{f_\omega(x)} e^{-f_\omega(x')} \right\}. \qquad (6)$$

Assessments. For each subset $\mathcal{X}_q$, the preferences among doxels are expressed according to a several discrete dimensions. We have:

- an information of exhaustivity, which measures how much a doxel answers the totality of an information need (0 not exhaustive, ..., 3 fully exhaustive)
- an information of specificity, which measures how much a doxel answers only the information need (0 not specific, ..., 3 means fully specific)

There is no preference between doxels sharing the same value of exhaustivity and specificity.

An assessment is a couple (exhaustivity, specificity). Let denote $\mathcal{A}$ the set of assessments and $A(x)$ the assessment of element x. We can define a partition $\mathcal{X}_q = \bigcup_{a \in \mathcal{A}} \mathcal{X}_q^a$, where

$$\mathcal{X}_q^a = \{x \in \mathcal{X}_q / A(x) = a\}.$$

We can rewrite (6):

$$R_e(\mathcal{X},\omega) = \sum_{q\in\mathcal{Q}}\sum_{a\in\mathcal{A}} \left\{ \left(\sum_{x\in\mathcal{X}_q^a} e^{f_\omega(x)}\right)\left(\sum_{\substack{b\in\mathcal{A}\\ \mathcal{X}_q^b\prec\mathcal{X}_q^a}}\sum_{x\in\mathcal{X}_q^b} e^{-f_\omega(x)}\right)\right\}. \tag{7}$$

where $\mathcal{X}_q^b \prec \mathcal{X}_q^a$ means that the assessments of the elements of $\mathcal{X}_q^a$ are better than those of $\mathcal{X}_q^b$. An possible order between assessments is represented in figure 1.

The complexity for computing this expression is $O(|\mathcal{Q}|.|\mathcal{X}|)$ whereas it is $O(|\mathcal{X}|^2)$ for (4).

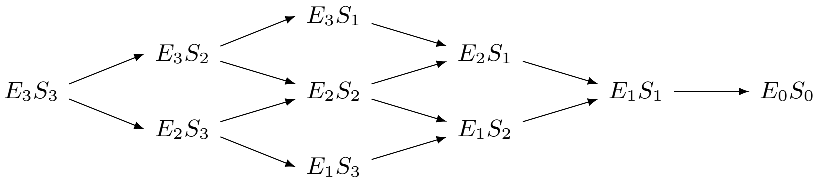


Fig. 1. Graph representing the order between elements for a given query, according to the two dimensional discrete scale of INEX. Doxels labeled E_3S_3 must be the highest ranked, and doxels labeled E_0S_0 the lowest ranked.

3.4 Gradient descent

Since (7) is convex, we can use a gradient descent technique to minimize it. The components of the gradient has the following form:

$$\begin{aligned}\frac{\partial R_e}{\partial \omega_k}(\mathcal{X},\omega) = \sum_{q\in\mathcal{Q}}\sum_{a\in\mathcal{A}} \Bigg\{ &\left(\sum_{x\in\mathcal{X}_q^a} x_k e^{f_\omega(x)}\right)\left(\sum_{\substack{b\in\mathcal{A}\\ \mathcal{X}_q^b\prec\mathcal{X}_q^a}}\sum_{x\in\mathcal{X}_q^b} e^{-f_\omega(x)}\right) \\ &+ \left(\sum_{x\in\mathcal{X}_q^a} e^{f_\omega(x)}\right)\left(\sum_{\substack{b\in\mathcal{A}\\ \mathcal{X}_q^b\prec\mathcal{X}_q^a}}\sum_{x\in\mathcal{X}_q^b} -x_k e^{-f_\omega(x)}\right)\Bigg\}\end{aligned} . \tag{8}$$

The complexity for computing the gradient is the same ($O(|\mathcal{Q}|.|\mathcal{X}|)$) as that of (7).

4 Experiments

4.1 Learning base

We used the series of topics and assessments from the INEX 2003 and 2004 collections as a learning base. We will comment the results on 2005 collection.

4.2 Filtering

In CO-Focussed task, overlapping doxels were not allowed. In order to suppress all overlapping elements from the lists computed by the ranking algorithm, we used a strategy which consists in removing all elements which are overlapping with an element ranked higher in the list.

As for Okapi model, we used the same strategy exept that biggest doxels like articles or bdy's were not allowed in the final ranking list to reach better performance.

4.3 Results

We comment the results obtained with the ncXG official metric with generalized quantization which is more related to the ranking loss criterion and the different levels of assessment we have used in our model.

CO-Focussed. We have plotted in figure 2 the evaluation of the lists produced by the ranking algorithm and by the modified Okapi when overlap is not authorized. We can see that the ranking algorithm performs better than Okapi. In some parts of the plot, the difference between the two models is not large: this is due to the post filtering of the lists. The ranked lists had not been optimized for non overlapping doxels since there is no notion of overlap in the exponential loss.

The table 1 shows that the ranking model is always significantly better than its baseline Okapi model, and that is quite good to retrieve the most informative doxels in the begining of the list.

Table 1. Rank of Okapi and ranking models among all participant submissions using MAncXG metric for CO-Focussed task

	@1	@2	@3	@4	@5	@10	@15	@25	@50	@100	@500	@1000	@1500
Okapi	21	20	19	19	18	18	19	19	19	18	20	20	20
Ranking	1	1	1	1	2	7	11	13	15	14	10	14	13

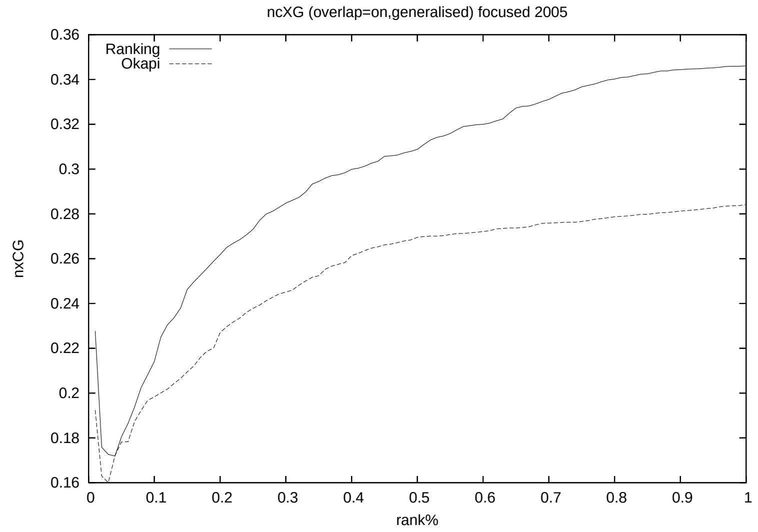


Fig. 2. Performance of ranking and Okapi models for CO-Focussed task evaluated with the cumulated gain based metric ncXG.

CO-Thorough. Figure 3 show the evaluation of the lists produced by the ranking algorithm and modified Okapi when overlap is authorized. We can see that the ranking algorithm performs clearly better than Okapi and the difference in performance is superior than in the Focussed task.

The table 2 shows that the ranking model is always significantly better than its baseline Okapi model, and that is quite good to retrieve the most informative doxels in the begining of the list. This can be explained by the expression of the ranking loss which penalize more a irrelevant doxel when it is located in the begining of the list.

Table 2. Rank of Okapi and ranking models among all participant submissions using MAncXG metric for CO-Thorough task

	@1	@2	@3	@4	@5	@10	@15	@25	@50	@100	@500	@1000	@1500
Okapi	26	22	26	26	26	31	34	37	38	38	35	32	32
Ranking	1	1	1	2	2	3	3	4	11	12	5	5	6

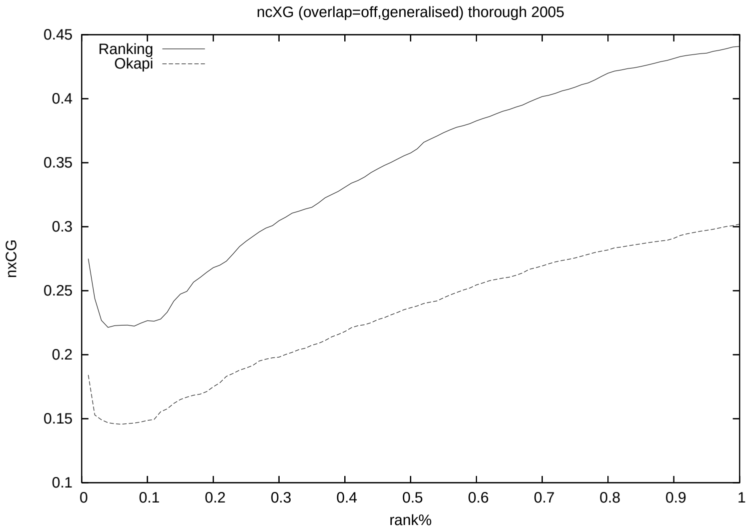


Fig. 3. Performance of ranking and Okapi models for CO-Thorough task evaluated with the cumulated gain based metric ncXG.

5 Conclusion

We have described a new model for CO tasks which relies on a combination of scores from the Okapi model and takes into account the document structure. This score combination is learned from a training set by a ranking algorithm.

For both tasks, the ranking algorithm has been able to increase by a significant amount the performance of the baseline Okapi. Ranking methods thus appear as a promising direction for improving SIR search engine performance. It remains to perform tests with additional features (for example the scores of additional IR systems).

References

1. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. In Jordan, M.I., Kearns, M.J., Solla, S.A., eds.: Advances in Neural Information Processing Systems. Volume 10., The MIT Press (1998)
2. Bartell, B.T., Cottrell, G.W., Belew, R.K.: Automatic combination of multiple ranked retrieval systems. In: Research and Development in Information Retrieval. (1994) 173–181
3. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. In Shavlik, J.W., ed.: Proceedings of ICML-98, 15th International Conference on Machine Learning, Madison, US, Morgan Kaufmann Publishers, San Francisco, US (1998) 170–178
4. Amini, M.R., Usunier, N., Gallinari, P.: Automatic text summarization based on word-clusters and ranking algorithms. In: ECIR. (2005) 142–156

5. Craswell, N., Robertson, S., Zaragoza, H., Taylor, M.: Relevance weighting for query independent evidence. In: SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (2005) 416–423
6. Auer, P., Meir, R., eds.: Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings. In Auer, P., Meir, R., eds.: COLT. Volume 3559 of Lecture Notes in Computer Science., Springer (2005)
7. Robertson, S.E., Walker, S., Hancock-Beaulieu, M., Gull, A., Lau, M.: Okapi at TREC. In: Text REtrieval Conference. (1992) 21–30
8. Vittaut, J.N., Piwowarski, B., Gallinari, P.: An algebra for structured queries in bayesian networks. In: Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004. Volume 3493 of Lecture Notes in Computer Science., Springer (2005)

Relevance Feedback for Structural Query Expansion

Ralf Schenkel and Martin Theobald

Max-Planck-Institut für Informatik, Saarbrücken, Germany
{schenkel,mtb}@mpi-inf.mpg.de

Abstract. Keyword-based queries are an important means to retrieve information from XML collections with unknown or complex schemas. Relevance Feedback integrates relevance information provided by a user to enhance retrieval quality. For keyword-based XML queries, feedback engines usually generate an expanded keyword query from the content of elements marked as relevant or nonrelevant. This approach that is inspired by text-based IR completely ignores the semistructured nature of XML. This paper makes the important step from pure content-based to structural feedback. It presents two independent approaches that include structural dimensions in a feedback-driven query evaluation: The first approach reranks the result list of a keyword-based search engine, using structural features derived from results with known relevance. The second approach expands a keyword query into a full-fledged content-and-structure query with weighted conditions.

1 Introduction

1.1 Motivation

XML has seen increasing importance recently to represent large amounts of semistructured or textual information in digital libraries, intranets, or the Web, so information retrieval on XML data is growing more and more important. XML search engines employ the ranked retrieval paradigm for producing relevance-ordered result lists rather than merely using XPath or XQuery for Boolean retrieval. An important subset of XML search engines uses keyword-based queries [2, 7, 26], which is especially important for collections of documents with unknown or highly heterogeneous schemas. However, simple keyword queries cannot exploit the often rich annotations available in XML, so the results of an initial query are often not very satisfying.

Relevance Feedback is an important way to enhance retrieval quality by integrating relevance information provided by a user. In XML retrieval, existing feedback engines usually generate an expanded keyword query from the content of elements marked as relevant or nonrelevant. This approach that is inspired by text-based IR completely ignores the semistructured nature of XML.

This paper makes the important step from content-based to structural feedback. We present two independent approaches to exploit the structure of XML with relevance feedback:

1. Using the feedback approach by Rocchio [18], we create new content and structural features that are used to rerank the results of a keyword-based engine, enabling structural feedback for engines that support only keyword-based queries.
2. We extend the feedback approach by Robertson and Sparck-Jones [17] to expand a keyword-based query into a possibly complex content-and-structure query that specifies new constraints on the structure of results, in addition to “standard” content-based query expansion. The resulting expanded query has weighted structural and content constraints and can be fed into a full-fledged XML search engine like our own TopX [22] Search Engine.

1.2 Related Work

Relevance feedback has already been considered for document retrieval for a long time, starting with Rocchio’s query expansion algorithm [18]. Ruthven and Lalmas [19] give an extensive overview about relevance feedback for unstructured data, including the assessment of relevance feedback algorithms.

Relevance feedback in XML IR is not yet that popular. Of the few papers that have considered it, most concentrate on query expansion based on the content of elements with known relevance [4, 12, 21, 25]. Some of these focus on blind (”pseudo”) feedback, others on feedback provided by users. Pan et al. [14] apply user feedback to recompute similarities in the ontology used for query evaluation.

Even fewer papers have considered structural query expansion [8, 9, 13]. Mihajlovič et al. [13] proposed deriving the relevance of an element from its tag name, but could not show any significant gain in retrieval effectiveness. Additionally, they considered hand-tuned structural features specific for the INEX benchmark (e.g., the name of the journal to which an element’s document belongs), but again without a significant positive effect. In a follow-up to this work, Ramírez et al. [15] could show significant improvements with journal names. In contrast, our general approach for exploiting feedback can be applied with the INEX data, but does not rely on any INEX-specific things.

Hlaoua and Boughanem [8] consider common prefixes of relevant element’s paths as additional query constraints, but don’t provide any experimental evaluation of their approach.

Gonçalvez et al. [5] use relevance feedback to construct a restricted class of structured queries (namely field-term pairs) on structured bibliographic data, using a Bayesian network for query evaluation. While they did not consider XML, their overall approach is somewhat similar to our reranking framework presented in Section 3.

The work of Hsu et al. [9] is closest to our approach. They use blind feedback to expand a keyword-based query with structural constraints derived from a neighborhood of elements that contain the keywords in the original query. Our approach considers the whole document instead of only a fragment, can generate constraints with negative weight, and integrates also content-based constraints.

2 Formal Model and Notation

2.1 Data Model

We consider a fixed corpus of D XML documents with their elements. For such an element e, $t(e)$ denotes its tag name and $d(e)$ the document to which it belongs.

The *content* $c(e)$ of an element e is the set of all terms (after stopword removal and optional stemming) in the textual content of the element itself and all its descendants. For each term t and element e, we maintain a weight $w_e(t)$. This can be a binary weight ($w_e(t) = 1$ if the term occurs in e's content and 0 otherwise), a tf-idf style [11] or a BM25-based [1, 24] weight that captures the importance of t in e's content. The *content* $c(d)$ of a document d is defined as the content $c(r)$ of its root element r.

We maintain a number of statistics about the occurrence of terms in documents and elements: The *document frequency* df_t of a term t is the number of documents in which the term appears in the content. Analogously, the *element frequency* ef_t of a term t is the number of elements in which the term appears in the content.

2.2 Queries and Relevance of Results

We use an extended version of INEX's query language NEXI [23]. NEXI basically corresponds to XPath restricted to the `descendants-or-self` and `self` axis and extended by an IR-style `about` predicate to specify conditions that relevant elements should fulfil. The wildcard symbol '*' matches any tag and can be used to formulate keyword queries in NEXI. We extend NEXI with additional weights for each content constraint. A typical extended NEXI query looks like the following:

```
//article[about(.,"0.8*XML")//*[about(.//p,"0.4*IR -0.2*index")]
```

The result granularity of such a query are elements. We currently assume that the relevance of an element with respect to a query is measured with the strict quantization, i.e., an element is either relevant or nonrelevant.

2.3 Feedback Model

We consider a keyword query $q = \{q_1, \ldots, q_p\}$ with a set $E = \{e_1, \ldots, e_l\}$ of results with known relevance. i.e., elements for which a user has assigned an exhaustivness value $e(e)$ and a specificity value $s(e)$. Using the strict quantization, we say that an element e is *relevant* for the query if both $e(e)$ and $s(e)$ are maximal, yielding a set $E^+ = \{e_1^+, \ldots, e_R^+\}$ of relevant elements and a set $E^- = \{e_1^-, \ldots, e_N^-\}$ of nonrelevant elements.

Note that even though we consider only binary relevance, it is possible to extend the mechanism presented here to approaches where relevance is measured with a probability-like number between 0 and 1, for example by representing E^+ and E^- as probabilistic sets.

3 Reranking Results of Keyword-only Runs

Our first approach aims at identifying documents that contain relevant elements and paths of relevant elements, in addition to standard content-based query expansion. We first compute the results for a keyword query with an existing keyword-based engine and ask a user for relevance feedback. Based on the user input, we compute certain classes of *features* from elements with relevance feedback and select those that best discriminate relevant from nonrelevant results. Using these features, we compute additional scores for element-feature-matches for all remaining elements and rerank them by their combined score. This approach allows to evaluate certain classes of structural constraints with engines that support only keyword-based queries.

For space restrictions we can only informally present our approach here; a more detailed and formal description can be found in [20].

3.1 Features Used for Reranking

We derive the following classes of candidates for query expansion from an element with known relevance:

- all terms of the element's content together with their score (C features),
- features derived from the path of the element (P features), and
- tag-term pairs within the element's document (D features).

The system can be extended with additional classes of features.

Content Features. Content-based feedback is widely used in standard IR and has also made its way into XML retrieval [12, 21]. It expands the original query with new, weighted keywords that are derived from the content of elements with known relevance. As an example, consider the keyword query `"multimedia information" retrieval` (this is topic 178 from the INEX topic collection). From the feedback of a user, we may derive that elements that contain the terms 'brightness', 'annotation', or 'rgb' are likely to be relevant, whereas elements with 'hypermedia' or 'authoring' are often irrelevant.

Document Features. Unlike standard text retrieval where the unit of retrieval are whole documents, XML retrieval focuses on retrieving parts of documents, namely elements. Information in other parts of a document with a relevant element can help to characterize documents in which relevant elements occur. A natural kind of such information is the content of other elements in such documents.

As an example, consider again INEX topic 178 (`"multimedia information" retrieval`). We may derive from user feedback that documents with the terms 'pattern, analysis, machine, intelligence' in the journal title (i.e., those from the 'IEEE Transactions on Pattern Analysis and Machine Learing') are likely to contain relevant elements. The same may hold for documents that cite papers by Gorkani and Huang (who are co-authors of the central paper about the QBIC system), whereas documents that cite papers with the term 'interface' in their title probably don't contain relevant elements (as they probably deal with interface issues in multimedia applications).

Other possible structural features include twigs, occurence of elements with certain names in a document, or combination of path fragments with terms. Further exploration of this diversity is subject to future work.

Path Features. Elements with certain tag names are more likely to be relevant than elements with other tag names. As an example, a keyword query may return entries from the index of a book or journal with high scores as they often contain exactly the requested keywords, but such elements are usually not relevant. Additionally, queries may prefer either large elements (such as whole articles) or small elements (such as single paragraphs), but rarely both. However, experiments show that tag names alone do not bear enough information to enhance retrieval quality, but the whole path of a result element plays an important role. As an example, the relevance of a paragraph may depend on whether it is in the body of an article (with a path like `/article/bdy/sec/p` from the root element), in the description of the vitae of the authors (with a path like `/article/bm/vt/p`), or in the copyright statement of the journal (with a path like `/article/fm/cr/p`).

As element tag names are too limited, but complete paths may be too strict, we consider the following six classes of *path fragments*, with complete paths and tag names being special cases:

- P_1: prefixes of paths, e.g., `article/#`,`/article/fm/#`
- P_2: infixes of paths, e.g., `#/fm/#`
- P_3: subpaths of length 2, e.g., `#/sec/p/#`
- P_4: paths with wildcards, e.g, `#/bm/#/p/#`
- P_5: suffixes of paths, e.g., `#/fig`, `#/article`
- P_6: full paths, e.g, `/article/bdy/sec`

Mihajlovič et al. [13] used a variant of P_5, namely tag names of result elements, but did not see any improvement. In fact, only a combination of fragments from several classes leads to enhancements in result quality. Experiments in [20] show that the best results are yielded with a combination of P1, P3 and P4, whereas using P5 or P6 alone actually reduced the quality of results below the baseline without feedback.

3.2 Feature Weights and Selection

We compute the weight for all features using the standard Rocchio weight [18]. We tried several variations, including binary and weighted Rocchio weights, and have yet to explore the whole space of solutions.

Among the (usually many) possible features, we choose the n_c content features and n_s document features with highest absolute weights. If there are too many with the same weight, we use the mutual information of the feature's score distribution among the elements with known relevance and the relevance distribution as a tie breaker. If there are no positive examples and mutual information is zero for all features, we use the feature's document frequency (the number of documents in which this feature occurs) for tie breaking then, preferring features that occur infrequently.

3.3 Reranking Results

The most promising approach for evaluating an expanded query is to evaluate as much of the expanded query with the existing search engine, evaluate the remaining part of the query separately, and combine the partial scores of each element. The result of the expanded query is then a ranked list of elements, sorted by combined score. Besides reusing the existing engine, this approach has the additional advantage that we can use different similarity measures for the different constraint dimensions and are not fixed to a single one (as with the extended vector space). On top of that, we can easily integrate new constraint dimensions.

For each element that occurs in the baseline run, we compute an additional score for each feature class. The score for each feature class is computed in a separate vector space where each dimension corresponds to a feature that occurs in at least one element. The score of the element for this class is then computed as the cosine of the vector with the selected features for this dimension and the element's feature vector. Each of the scores is normalized to the interval $[-1.0, 1.0]$. The overall score of the element is then the sum of its score from the baseline run and its additional scores.

This scoring model can easily integrate new dimensions for feedback beyond content, path and document features, even if they use a completely different model (like a probabilistic model). It only requires that the relevance of an element to a new feedback dimension can be measured with a score between -1 and 1. It is simple to map typical score functions to this interval by normalization and transformation. As an example, the transformation rule for a probability p, $0 \leq p \leq 1$, is $2 \cdot p - 1$.

4 Generating Structural Queries from Feedback

Keyword-based queries are the best way to pose queries without knowledge of the underlying schema of the data, but they cannot exploit the structure of documents. As an example, consider the keyword query (query 230 from the INEX benchmark [10]) `+brain research +"differential geometry"`, asking for applications of differential geometry in brain research. In relevant results, ”brain research” is usually the topic of the whole article, while ”differential geometry” is typically the topic of a section. A query with constraints on both content and structure would probably yield a lot more relevant results, but it is impossible to formulate a query like the following without knowledge of the underlying schema:

```
//article[about(.,brain research)]//sec[about(.,differential geometry)]
```

We studied the content-and-structure queries from INEX to find patterns that are regularly used in such queries to describe relevant elements, in addition to content conditions on the result element. A canonical example for such a query is the following:

```
//article[about(.,"RDF") and about(//bib,"W3C")]//sec[about(.,"query")
and about(//par,"performance")]
```

that is a content-and-structure version of the simpler keyword query ”RDF W3C query performane”. In contrast to the keyword query, the structured query specifies a tag (or, more generally, a set of tags) that relevant elements should have (”I am interested in sections about ’query’”). Additionally, this query contains constraints on the content of descendants of relevant elements (”sections with a paragraph about ’performance’”), the content of ancestors (”sections in articles about ’RDF’”), and the content of descendants of ancestors (”sections in articles that cite a paper from the ’W3C’”).

As such a content-and-structure query specifies much more precisely the conditions that relevant elements must satisfy, we can expect that a search engine will return more relevant results for a content-and-structure query than for the keyword query, provided that the content-and-structure query correctly captures the same information need as the keyword query. Our feedback framework aims at generating a content-and-structure query from a keyword query, exploiting relevance feedback provided by a user for some results of the keyword query.

4.1 Candidates for Query Expansion

Following the discussion in the beginning of this section, we derive the following classes of candidates for query expansion from an element with known relevance:

- all terms of the element’s content together with their score (C candidates),
- all tag-term pairs of descendants of the element in its document, together with their score (D candidates),
- all tag-term pairs of ancestors of the element in its document, together with their score (A candidates), and
- all tag-term pairs of descendants of ancestors of the element in its document, together with their score and the ancestor’s tag (AD candidates).

The system can be extended with additional classes of candidates like tags, twigs, or paths, which is subject to future work.

The candidate set of an element is the set of all possible candidates for this element. We extend the notion of frequencies from terms to candidates as follows: the element frequency of a candidate is the number of elements where the candidate appears in the candidate set, and its document frequency its the number of documents with at least one element with the candidate in their candidate set.

4.2 Candidate Weights and Selection

To weight the different candidates, we apply an extension of the well-known Robertson-Sparck-Jones weight [17] to element-level retrieval in XML, applying it for elements instead of documents:

$$w_{RSJ}(c) = \log \frac{r_c + 0.5}{R - r_c + 0.5} + \log \frac{E - ef_c - R + r_c + 0.5}{ef_c - r_c + 0.5}$$

Here, for a candidate c, r_c denotes the number of relevant elements which contain the candidate c in their candidate set, R denotes the number of relevant elements, E the number of elements in the collection, and ef_c the element frequency of the candidate.

The set of all possible expansion candidates is usually very large and contains many unimportant and misleading expansions, so we have to select the best b of them for generating the expanded query. This problem already exists for content-based expansion of keyword queries, and several possible weights have been proposed in the literature that go beyond naively ordering terms by their weight. We use the so-called Robertson Selection Values (RSV) proposed by Robertson [16]. For a candidate c, its RSV has the form $RSV(c) = w_{RSJ}(c) \cdot (p - q)$, where $p = r_c/R$ is the estimated probability of the candidate occurring in a relevant element's candidate set and q is the probability that it occurs in a nonrelevant element's set. We ignore candidates that occur only within the documents of elements with known relevance as they have no potential to generate more relevant results outside these documents. We order the union of the remaining candidates by their RSV and choose the top b of them, where b is a configuration parameter of the system. To be able to generate a valid NEXI query in the next step, we have to limit the A and AD candidates chosen to contain the same ancestor tag.

4.3 Generating an Expanded Query

Using the top-b candidates, we generate a content-and-structure query from the original keyword query. This expansion is actually straight-forward, and the generated query has the following general structure:

```
//ancestor-tag[A+AD constraints]//*[keywords+C+D constraints]
```

As an example, if the original query was 'XML' and we selected the A candidate (anc,article,'IR'), the AD candidate (article,bib,'index') and the D candidate (desc,p,'index'), the expanded query would be

```
//article[about(.,'IR') and about(//bib,'index')]//*[about(.,'XML')
and about(//p,'index')]
```

Each of the expansions is weighted, where the weight is the candidate's RSJ weight adjusted by a factor that depends on the candidate's class. C and D candidates help finding new relevant results, so they should get a high weight; we allow for C and D conditions at most the weight of all original keywords (to make sure that the new constraints don't dominate the query's results). As an example, for a query with four keywords and six C and D expansions, the factor for each expansion is $\frac{4}{6}$. On the other hand, A and AD conditions are satisfied by most – if not all – elements of a document, so they generate a huge amount of new result elements, most of which will be nonrelevant. Their weight should therefore be smaller than the weight of C and D conditions. We choose a fraction β of the accumulated weight of existing keyword conditions, with $\beta = 0.2$ in our experiments.

5 Architecture and Implementation

We have implemented the reranking approach from Section 3 and the query expansion approach from Section 4 within an automated system that can import queries and results from INEX and automatically generate feedback for the top-k results, using the existing INEX assessments.

Our Java-based implementation requires that important information about elements is precomputed: unique identifiers for the element (`eid`) and its document (`did`), its pre and post order to facilitate the evaluation of structural query conditions like the XPath axes [6] or any other similar information, its tag, and its terms (after stemming and stopword removal), together with their score. This information is stored in a database table with schema `(did,eid,term,tag,pre, post,score)` that contains one tuple for each distinct term of an element. Our current prototype reuses the `TagTermFeatures` table of TopX (see [22] that already provides this information. On the database side, we provide indexes on `(eid,did)` to efficiently find $d(e)$ for an element e and on `(did)` to efficiently collect all elements of a document. Inverse element and document frequencies of the different candidate classes are precomputed (e.g., while initially parsing the collection) and stored in database tables, too.

6 Evaluation of Feedback Runs

The evaluation of feedback runs for XML IR is a problem that has not yet been solved in a satisfying way. People have agreed that simply comparing the results of a run with feedback to the baseline run (which we call *plain* later) is unfair as the new run has information about relevant elements and hence is biased. The INEX Relevance Feedback track has used two different measures so far:

- In 2004, a variant of the residual collection technique was used. Here, all XML elements with known relevance must be removed from the collection before evaluation of the results with feedback takes place. This means not only each element used or observed in the RF process but also all descendants of that element must be removed from the collection (i.e., the residual collection, against which the feedback query is evaluated, must contain no descendant of that element). All ancestors of that element are retained in the residual collection.
- In 2005, the rank of results with know relevance is frozen, thus assessing only the effect of reranking the results with unknown relevance. We label

this approach *freezeTop* as usually the top-k results are used for feedback and hence frozen.

Using the residual collection approach opens up a variety of different evaluation techniques:

- *resColl-result*: only the elements for which feedback is given are removed from the collection,
- *resColl-desc*: the elements for which feedback is given and all their descendants are removed from the collection (this is the technique used in this year's RF track),
- *resColl-anc*: the elements for which feedback is given and all their ancestors are removed from the collection, and
- *resColl-doc*: for each element for which feedback is given, the whole document is removed from the collection.

We evaluate our approaches with all six evaluation techniques in the following section and try to find out if there are any anomalies.

7 Experimental Results

7.1 Settings

For all experiments we used our TopX engine that fully supports the evaluation of weighted content-and-structure queries. The baseline for all experiments is a run generated with our current TopX engine for the 2005 CO topics, with 1500 results for each topic[1]. Table 1 shows the macro-averaged precision for this run for the top-k ranked elements per topic, for different k; this corresponds to the average fraction of relevant results among the elements used for top-k feedback.

k	10	15	20
prec@k	0.0593	0.0519	0.0500

Table 1. Precision at k for the baseline run

Note that the average precision is much lower than in previous experiments with the INEX 2003 and 2004 CO topics where TopX yielded an average precision of 0.231 at the top-5 and still 0.174 at the top-20 results. We can effectively use only 10 of the 40 2005 CO topics (those with assessments and with at least

[1] We did not use the official run as baseline because the engine has evolved a lot since the time that run was produced. The main reason for the relatively low MAP value of this run compared to our official run is the result of topic 228 where the new run ranks the only result in rank 2 instead of 1, yielding a MAP difference for this topic of about 0.7.

one relevant result among the top-20 results) for the experiments, which makes the significance of the experiments at least questionable and at the same time explains the relatively low improvements shown in the following subsections. Table 2 gives some more information on this issue: it shows the number of topics in the baseline run that have a certain number of relevant results among the top k, for varying k. Again it is evident that most topics do not have any relevant results at all. As an additional problem, there are some topics with only a few relevant results, which makes possible that slight changes in the order of results cause huge differences in the resulting MAP values. We decided therefore to present only our results with top-20 feedback in the following as the other results would not be significant anyway.

k/r	0	1	2	3	4	5	6
5	23	1	3	0	0	0	-
10	19	5	0	2	0	1	0
15	18	5	0	2	1	0	1
20	17	4	1	3	0	0	2

Table 2. Number of topics in the baseline run with r relevant results among the top k results

For each run, we measured the MAP using inex_eval with the strict quantization and the latest set of assessments. We plan to evaluate our results with other metrics in the future.

7.2 Results for Reranking Queries

Table 3 shows the MAP values of our experiments with different combinations of features to rerank the initial results, providing relevance feedback for a different number of top elements of the baseline run and selecting the best 5 features of each dimension, for the different evaluation techniques.

The results are surprisingly different from our earlier results in [20]: the absolute improvemens are quite small, and the best results are achieved with either only P, only D, or both candidates. We attribute this to the fact that there are at most 10 topics with relevant results in the top-k; the remaining topics provide only negative feedback which seems to be not helpful here. Additionally, it is evident that the different evaluation methods don't agree at all about which combination gives the best results. Other than that, it is interesting that some runs (like the D run which looks like the absolutely best choice with the freezeTop evaluation) do great with some evaluation techniques, but perform worse than the baseline with others. This is an anomaly that should be investigated further.

evaluation	baseline	C	P	C+P	D	C+D	D+P	C+D+P
plain	0.0367	0.0465	0.1008	0.0534	0.0911	0.0492	**0.1120**	0.0563
resColl-result	0.0262	0.0343	**0.0581**	0.0216	0.0412	0.0312	0.0579	0.0228
resColl-anc	0.0267	0.0340	0.0581	0.0198	0.0400	0.0297	**0.0589**	0.0219
resColl-desc	0.0330	0.0180	0.0489	0.0142	0.0284	0.0132	**0.0498**	0.0151
resColl-doc	0.0309	0.0140	**0.0480**	0.0114	0.0249	0.0097	0.0468	0.0126
freezeTop	0.0367	0.0367	0.0371	0.0353	**0.0373**	0.0369	0.0362	0.0358

Table 3. MAP values for top-20 feedback runs with different configurations and different evaluation methods, for the reranking approach

7.3 Results for Queries with Structural Constraints

Table 4 shows a comparison of MAP values with the different evaluation techniques of our experiments with different combinations of candidate classes for query expansion, providing relevance feedback for the top 20 elements of the baseline run and selecting the best 10 candidates for expansion.

The results are less impressive than we had expected after earlier experiments with the 2003 and 2004 CO topics. The best of our techniques (which consistently is the combination of all candidate classes for all five evaluation techniques, not counting the plain run) yields a performance gain of about 5%–20%, whereas we could show up to 100% in the other experiments (with resColl-desc). We think that there are mainly two reasons for this difference: The 2005 topics are inherently more difficult than the older topics (which is also reflected in the much lower MAP scores for the best runs this year), and there are only 10 topics where our approaches have a chance to enhance the quality (because Robertson-Sparck-Jones weights are not useful without relevant results) with top-20 feedback and even fewer for the other runs. Absolute values are slightly higher than the values achieved with the reranking approach, so choosing the best candidates out of a pool of expansion candidates instead of picking a fixed number from each class seems to perform better.

evaluation	baseline	C	D	C+D	A	AD	A+AD	A+C+D+AD
plain	0.0367	0.0419	**0.0707**	0.0406	0.0646	0.0663	0.0654	0.0513
resColl-result	0.0262	0.0294	0.0300	0.0295	0.0309	0.0306	0.0294	**0.0324**
resColl-anc	0.0267	0.0350	0.0356	0.0305	0.0298	0.0294	0.0296	**0.0366**
resColl-desc	0.0330	0.0353	0.0355	0.0355	0.0363	0.353	0.0346	**0.0365**
resColl-doc	0.0309	0.0317	0.0313	0.0314	0.0321	0.0310	0.0315	**0.0325**
freezeTop	0.0367	0.0378	0.0374	0.0375	0.0384	0.0378	0.0380	**0.0387**

Table 4. MAP values for top-20 feedback runs with different configurations and different evaluation methods, for the query expansion approach

References

1. G. Amati, C. Carpineto, and G. Romano. Merging XML indices. In *INEX Workshop 2004*, pages 77–81, 2004.
2. A. Balmin et al. A system for keyword proximity search on XML databases. In *VLDB 2003*, pages 1069–1072, 2003.
3. H. Blanken, T. Grabs, H.-J. Schek, R. Schenkel, and G. Weikum, editors. *Intelligent Search on XML Data*, volume 2818 of *LNCS*. Springer, Sept. 2003.
4. C. J. Crouch, A. Mahajan, and A. Bellamkonda. Flexible XML retrieval based on the extended vector model. In *INEX 2004 Workshop*, pages 149–153, 2004.
5. M. A. Gonçalves, E. A. Fox, A. Krowne, P. Calado, A. H. F. Laender, A. S. da Silva, and B. Ribeiro-Neto. The effectiveness of automatically structured queries in digital libraries. In *4th ACM/IEEE-CS joint conference on Digital libraries (JCDL04)*, pages 98–107, 2004.
6. T. Grust. Accelerating XPath location steps. In *SIGMOD 2002*, pages 109–120, 2002.
7. L. Guo et al. XRANK: ranked keyword search over XML documents. In *SIGMOD 2003*, pages 16–27, 2003.
8. L. Hlaoua and M. Boughanem. Towards context and structural relevance feedback in XML retrieval. In *Workshop on Open Source Web Information Retrieval (OSWIR)*, 2005. http://www.emse.fr/OSWIR05/.
9. W. Hsu, M. L. Lee, and X. Wu. Path-augmented keyword search for XML documents. In *ICTAI 2004*, pages 526–530, 2004.
10. G. Kazai et al. The INEX evaluation initiative. In Blanken et al. [3], pages 279–293.
11. S. Liu, Q. Zou, and W. Chu. Configurable indexing and ranking for XML information retrieval. In *SIGIR 2004*, pages 88–95, 2004.
12. Y. Mass and M. Mandelbrod. Relevance feedback for XML retrieval. In *INEX 2004 Workshop*, pages 154–157, 2004.
13. V. Mihajlovič et al. TIJAH at INEX 2004 modeling phrases and relevance feedback. In *INEX 2004 Workshop*, pages 141–148, 2004.
14. H. Pan, A. Theobald, and R. Schenkel. Query refinement by relevance feedback in an XML retrieval system. In *ER 2004*, pages 854–855, 2004.
15. G. Ramírez, T. Westerveld, and A. P. de Vries. Structural features in content oriented XML retrieval. In *CIKM 2005*, 2005.
16. S. Robertson. On term selection for query expansion. *Journal of Documentation*, 46:359–364, Dec. 1990.
17. S. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society of Information Science*, 27:129–146, May–June 1976.
18. J. Rocchio Jr. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1971.
19. I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(1), 2003.
20. R. Schenkel and M. Theobald. Feedback-driven structural query expansion for ranked retrieval of XML data. In *10th International Conference on Extending Database Technologies (EDBT 2006)*, Munich, Germany, Mar. 2006.
21. B. Sigurbjörnsson, J. Kamps, and M. de Rijke. The University of Amsterdam at INEX 2004. In *INEX 2004 Workshop*, pages 104–109, 2004.
22. M. Theobald, R. Schenkel, and G. Weikum. An efficient and versatile query engine for TopX search. In *VLDB 2005*, pages 625–636, 2005.

23. A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). available at `http://www.cs.otago.ac.nz/postgrads/andrew/2004-4.pdf`, 2004.
24. J.-N. Vittaut, B. Piwowarski, and P. Gallinari. An algebra for structured queries in bayesian networks. In *INEX Workshop 2004*, pages 58–64, 2004.
25. R. Weber. Using relevance feedback in XML retrieval. In Blanken et al. [3], pages 133–143.
26. Y. Xu and Y. Papakonstantinou. Efficient keyword search for smallest LCAs in XML databases. In *SIGMOD 2005*, pages 537–538, 2005.

NLPX at INEX 2005

Alan Woodley, Shlomo Geva

School of Software Engineering and Data Communications, Faculty of Information Technology, Queensland University of Technology
GPO Box 2434, Brisbane, Queensland, Australia
{ap.woodley@student.qut.edu, s.geva@qut.edu.au}

Abstract. XML information retrieval (XML-IR) systems aim to provide users with highly exhaustive and highly specific results. To interact with XML-IR systems, users must express both their content and structural requirement, in the form of a structured query. Traditionally, these structured queries have been formatted using formal languages such as XPath or NEXI. Unfortunately, formal query languages are very complex and too difficult to be used by experienced, let alone casual users; and are also too closely bound to the underlying physical structure of the collection. Hence, recent research has investigated the idea of specifying users' content and structural needs via natural language queries (NLQs). The NLP track was established at INEX 2004 to promote research into this area, and QUT participated with the system NLPX. Here, we discuss changes we've made to the system since last year, as well as our participation in INEX 2005.

1 Introduction

Information retrieval (IR) systems respond to user queries with a ranked list of relevant results. Traditionally, these results have been whole documents but since XML documents separate content and structure, XML-IR systems are able to return highly specific information to users, lower than the document level. However, to take advantage of this capability XML-IR users require an interface that is powerful enough to express their content and structural requirements, yet user-friendly enough that they can express their requirements intuitively.

Historically, XML-IR systems have used two types of interfaces, keyword-based and formal query language-based. Keyword-based systems are user-friendly, but lack the sophistication to properly express users' content and structural needs. In comparison, formal query language-based interfaces are able to express users' content and structural needs, but are too difficult to use, especially for casual users [7,9] and are bound to the physical structure of the document. Recently, investigation has begun into a third option for interfacing with XML-IR systems via a natural language interface that will allow users to fully express their content and structural needs in an intuitive and easy to use manner.

We have previously presented NLPX [10,11] an XML-IR system with a natural language interface. NLPX accepts natural language queries (NLQs) and translates them into NEXI queries. NEXI is an XPath-like formal query language that is used as a frontend to many existing XML-IR systems. NLPX participated in the natural language processing track of the 2004 INitiative for the Evaluation of XML Retrieval Workshop (INEX). INEX's NLP uses the same Content Only (CO) and Content and Structure (CAS) topics as its Ad-hoc track, however, as input systems use the topics' *Description* rather than *Title* element.

Since last year's participation we have made several improvements to NLPX. Here we discuses three major improvements: inclusion of more special connotations, introduction of shallow parsing and inclusion of more templates. We also describe our participation in the INEX 2005 NLP track and present our results.

2 Motivation

We have already outlined the motivations for an XML-IR natural language interface in our previous work [10,11]; however, for completeness we include them here. The motivations stem from the problems with formal XML-IR query languages and are two fold: first, formal query languages are difficult to use, and second, they are too tightly bound to the physical structure of documents.

First, formal query languages are too difficult for many users to correctly express their information need. Two very good examples of this have occurred at the 2003 and 2004 INEX Workshops. In 2003 INEX used the XPath [3] formal language to specify structured queries; however, 63% of the proposed queries had major semantic or syntactic errors. Furthermore, the erroneous queries were difficult to fix, requiring 12 rounds of corrections. In response to this problem, O'Keefe and Trotman [7] designed a simplified version of XPath called NEXI, which was used in INEX 2004. When NEXI was used, the error rate dropped to 12%, with the number of topic revision halved [9].While these figures are limited to two formal languages, O'Keefe and Trotman investigated other structured query languages such as HyTime, DSSSL, CSS and XIRQL and concluded that all of them are very complicated and difficult to use. Therefore, if experts in the field of structured information retrieval are unable to correctly use complex query languages, one cannot expect an inexperienced user to do so. In fact, recent research by van Zwol et. al [13] confirmed the difficulty that casual users have in formulating NEXI queries. However, we feel that users would be able to intuitively express their information need in a natural language.

Secondly, formal query languages are too tightly bound to the physical structure of documents; hence, users require an intimate knowledge of documents' composition in order to express their structural requirements properly. So, in order for users to retrieve information from abstracts, bodies or bibliographies, they will need to know the actual names of those tags in a collection (for instance: *abs*, *bdy*, and *bib*). While this information may be obtained from a document's DTD or Schema there are situations where the proprietor of the collection does not wish users to have access to those files. Or, in the case of a heterogeneous collection, a single tag can have multiple names (for example: abstract could be named *abs*, *a*, or *abstract*). This is a problem identified by participants in the INEX 2004 heterogenous track [6]. In contrast, structural requirements in NLQs are expressed at a higher conceptual level, allowing the underlying document's structure to be completely hidden from users. Other proposed solutions include the use of metatags to map between collections [6] and extensions to NEXI [9].

3 Previous Work by Authors

This paper expands on the previous work of the authors presented in [10,11]. We submitted our system, NLPX, to the 2004 INEX Natural Language Processing Track where it performed very successfully (1st in CAS, 2nd in CO). INEX's NLP track used the same topics and assessments as its Ad-hoc track; however, participating systems used a natural language query as input, rather than a query a formal language (NEXI) query. Examples of both query types are expressed in Figure 1. Note that the query actually contains two information requests, first, for sections about compression, and second, for articles about information retrieval. However, the user only wants to receive results matching the first request. We refer to the former as returned requests/results and the latter as support requests/results.

NEXI: //article[about(.,'information retrieval')] //sec[about(./, compression)]

NLQ: Find sections of articles about image and text compression in articles about efficient information retrieval

Fig. 1. A NEXI and Natural Language Query

We had previously participated in INEX's Ad-hoc track with GPX, a system that accepted NEXI formatted queries. Therefore, we decided to use GPX as a backend system. This allowed us to concentrate on developing a frontend that translated natural language queries to NEXI. Translation involved three steps that derived syntactic and semantic information from the natural language query (NLQ). We refer to these three steps as the NLPX framework and outline them below:

1. First we tagged words in the NLQ as either a special connotation or by their part of speech. Special connotations are words of implied semantic significance. We differentiated between three types: Structures (such as section, abstract) that specified structural requirements, Boundaries (such as contains, about) that separated structural and content requirements, and Instructions (such as find, retrieve) that indicated if we had a return or support request. Words corresponding to special connotations were hard-coded into the system and matched to query words by a dictionary lookup. Remaining words were tagged by their part of speech (such as noun, verb, conjunction) via a Brill Tagger [2].

2. Second, we matched the tagged NLQs to query templates. The templates were derived from the inspection of previous INEX queries. Since the NLQs occurred in shallow context they required only a few templates, significantly less than if one wished to capture natural language as a whole. Each template corresponded to an information request. Each request had three attributes: Content, a list of terms/phrases expressing content requirements, Structure, a logical XPath expression expressing structural requirements, and an Instruction, "R" for return requests, and "S" otherwise.
3. Finally, the requests were merged together and output in NEXI format. Return requests were output in the form **A[about(.,C)]** where **A** is the request's structural attribute and **C** is the request's content attribute. When all return requests were processed, support requests were inserted. The insert position was located by comparing the structural attributes of return and support requests and by finding their longest shared descendant. The output of support requests had the form **D[about(E,F)]** where **D** is the longest matching string, **E** is the remainder of the support's structural attribute and F is the support's content attribute.

4 Improvements

Since our participation in INEX 2004 we have made several improvements to NLPX, here we outline three major improvements and outline their motivation. Our first two improvements were to increase the number of special connotations and templates recognised by NLPX. These improvements correspond to the first two steps of the NLPX framework established in Section 3. These improvements increased the range of queries NLPX could handle, thereby increasing its robustness. The third improvement was to implement a shallow parsing stage between the first two framework steps. The shallow parser grouped together query terms into atomic semantic units before full parsing. This allowed for further lexical analysis to be performed on the units, leading to an overall increase in retrieval performance. Here, we discuss these three improvements in detail.

4.1 Additional Special Connotations

The INEX natural language queries are very diverse in nature, presenting a challenge for all those wishing capture their syntactic and semantic meaning, via a natural language inference. In NLPX, we tag query words of semantic importance as special connotations. Previously, NLPX recognised three special connotations: Instructions, Boundaries and Structures. These connotations were able to handle many of the INEX queries, however, they were not able to handle some of the more novel NLQs. Therefore, we have extended the number of conations recognised by NLPX to allow for a broader range of queries to be handled. Here we describe the special connotations we added to NLPX.

4.1.1 Negators

The first connotation added to NLPX was negators. Negators fulfil the user's information need, by explicitly stating the information content that the user **does not** want to retrieve, rather than the information content that they want to retrieve. Negators are expressed in NEXI by the use of a minus symbol (-), and are expressed in NLQs by the use of words such as *no*, *non* or *not*. An example of a negation occurs in topic number 139.

NEXI: //article[(about(.//bb//au//snm, Bertino) or about(.//bb//au//snm , Jajodia)) and about(.//bb//atl, security model) and about(.//bb//atl, -"indexing model" - "object oriented model")]

NLQ: We wish to identify papers that cite work by authors Bertino or Jajodia that deal with "security models". Security models should thus be the subject in the title of the cited papers by Bertino or Jajodia. We are interested in any kind of security models that Bertino or Jajodia developed (e.g. authorization models). We are **not** interested in other kind of models (e.g. objet oriented/indexing models).

Fig. 2. Topic Number 139. An example of the use of a negator.

4.1.2 Strengtheners

The second connotation we added to NLPX was strengtheners. Users employ strengthens to add weighting to query terms that are highly important to their information need. Strengthens are expressed in NEXI by the use of the plus (+) symbol, and are expressed in NLQs by the use of terms and phrases such as *particularly* and *major focus*. An example of a strengthener occurs in topic number 137.

NEXI: //article [about(.//abs, "digital library") or about(.//ip1, "digital library")]

NLQ: Find articles having digital libraries as their **major focus**, which means that the topic should be treated in the abstract or ingresses of the document to be relevant

Fig. 3. Topic Number 137. An example of the use of a strengthener.

4.1.3 Reverse Boundaries

The third connotation added was reverse boundaries. Previously we had identified a boundary as a query term that separates structural items and content items. NLPX uses boundaries to pair structures with their respective content. Examples of boundaries are query terms such as *talk about* or *contains*. Reverse boundaries have a similar function to ordinary boundaries, since NLPX also uses them to pair together structures and content; however, reverse boundaries occur after the content items rather than before. Often reverse boundaries are past tense versions of ordinary boundaries such as *talked about* or *contained*. An example of a reverse boundary occurs in topic number 160.

NEXI: //article[about(., image retrieval)]//sec[about(., "latent semantic indexing")]

NLQ: We are looking for sections in articles where "image retrieval" is **talked about**, that describe "latent semantic indexing".

Fig. 4. Topic Number 160. An example of the use of a reverse boundary.

4.1.4 Self-Reference Topics

The fourth connotation added was a self-reference topic. A self-reference topic occurs when some part, usual a content item, of the topic is referred to later on in the topic using a pseudonym. Self-reference topics are conceptually similar to a noun subsequently been referenced via a pronoun. An example of self-reference topic occurs in topic number 161 where the phrase *that topic* refers to the previously mentioned content terms *database access methods for spatial and text data*.

NEXI: //article[about(., database access methods for spatial data and text)]//bm//bb[about(./atl, database access methods)]

NLQ: Find bibliography entries about database access methods for spatial and text data from articles related to **that topic**.

Fig. 5. Topic Number 161. An example of the use of a self-reference topic

4.1.5 Inclusion of Stopwords

The final new special connotations recognised by NLPX were stopwards. Stopwords are words that occur in too frequently to be of any value in IR systems and are often ignored. Our backend system GPX already ignores some stopwords; however, we incorporated also them into NLPX since we wanted NLPX to be a generic interface that could be used by any XML-IR backend system. Rather than use the same stoplist used in GPX that is derived from frequently occurring (>50,000 times) terms in the INEX corpus we used a standard stop list defined in [4]. Once again we made this decision so that NLPX would be a more generic interface.

4.2 Shallow Parsing

The second improvement we made to NLPX was to add an intermediate step of shallow parsing between our lexical tagging and template matching. Shallow parsing, also called text chunking, is the process of dividing sentences into atomic, nonoverlapping segments (called chunks), and then classifying them into grammatical classes. It is usually performed after part of speech tagging, and as demonstrated by Abney [1] it can be used as a precursor to full parsing. Alternatively it can be used in other tasks such as index term generation, information extraction, text summation and bilingual alignment. Initial research into shallow parsing was focused on identifying noun phrases; however, more recent work has extended its reach to include general clause identification.

There are two types of chunks that are systems recognised:

- Explicit Chunks: These are chunks that are explicitly defined by users by adding parenthesises around important phrases in the query. For the purpose of NLPX characters that signified a parenthesises were commas, colons, semi-colons, brackets and quotation marks. Generally, parenthesises were added to important content phrases rather than other types of phrases.
- Implicit Chunks: These are chunks that are not explicitly defined by users, but rather derived by analysing the grammatical properties and/or context of query terms. It is used to group together terms of implied significance in the system. A classic example is to group together adjectives and nouns to forma single noun phrase. In NLPX we identify four chunks of significance: Instructions, Structures, Boundaries (include Reverse Boundaries) and Content.

We have previously incorporated a shallow parser in our previous work [12]. In that version a process called transformation-based learning (TBL) [2] to learn when to include query terms into a chunk based on both its grammatical properties (the tag of the current term) and its context (the tags of surrounding terms). This process was based upon the work of Ramshaw and Marcus [8] who originally used it to group together noun phrases. We extended their theories to work on structured queries. Unfortunately, we did not have time to retrain our system to recognise the new special connotations introduced earlier, therefore, we based the decision solely on the tag of the current term.

4.3 Additional Templates

The final improvement we made to NLPX was the addition of new templates. These additions were needed to handle both the new special connotations and the grouping of query terms into chunks. Figure 7 presents the templates that NLPX previously recognised:

Query: Request+
Request : CO_Request | CAS_Request
CO_Request: NounPhrase+
CAS_Request: SupportRequest | ReturnRequest
SupportRequest: Structure [Bound] Content+
ReturnRequest: Instruction Structure [Bound] Content+

Fig. 6. Existing NLPX Query Templates

Note that these templates work only on a word rather than a chunk level. However, it was straightforward to migrate the templates since the set of four single–term terminals (Instruction, Structure, Boundary and Content) had corresponding chunk classes. However, we also added new query templates to the NLPX, which we describe here.

4.3.1 Conjuncting Structures

The first template added to NLPX was used to handle conjucting structures. This occurs when two structures are separated by a conjunction (for example and, or). In this situation it is implied that users wish to search elements that match either of the structures. Figure 8 presents the templates added to the system while Figure 9 presents topic 127, an example of conjuecting structures.

Structure: StructureChunk [OtherStructure+]
OtherStructure : Conjunction StructureChunk

Fig. 7. Conjucting Structures Query Templates

NEXI: //sec//(p| fgc)[about(., Godel Lukasiewicz and other fuzzy implication definitions)]

NLQ : Find **paragraphs or figure-captions** containing the definition of Godel, Lukasiewicz or other fuzzy-logic implications

Fig. 8. Topic Number 127. An example of the use of conjucting structure

4.3.2 Reverse Boundaries

The second template added to NLPX was to handle the cases of reverse boundaries. When NLPX encounters a Reverse Bounding it immediately matches the previously parsed content items with the current structure begins a new request. Figure 10 presents the new query templates used to handle reverse boundaries and Figure 11 presents topic 160, which contains an example of a reverse boundary (and previous presented in figure 4).

SupportRequest: Structure [Bound] Content+ |
Structure Content+ [ReverseBound]
ReturnRequest: Instruction Structure [Bound] Content+ |
Instruction Structure Content+ [ReverseBound]

Fig. 9. Reverse Boundary Query Templates

NEXI: //article[about(., image retrieval)]//sec[about(., "latent semantic indexing")]

NLQ: We are looking for sections in articles where "image retrieval" is **talked about**. that describe "latent semantic indexing".

Fig. 10. Topic Number 160. An example of the use of a reverse boundary.

4.3.3 Parenthetical Information Requests

Parenthetical information requests occur when a new information request occurs in the middle of another information request. Usually this occurs when a boundary element occurs after a completed information request, thereby indicating that a instruction or a structure has preceded it. When this occurs, NLPX must fully handle the new information request, before returning to handle the remaining content information. Figure 11 presents the new query templates used to handle parenthetical information requests and Figure 12 presents topic 160, which contains an example of a parenthetical information requests (and previous presented in figure 4).

SupportRequest: Structure [SupportRequest] Bound Content+ |
ReturnRequest: Instruction Structure [ReturnRequest] Bound Content+

Fig. 11. Parenthetical Information Request Templates

NEXI: //article[about(., image retrieval)]//sec[about(., "latent semantic indexing")]
NLQ: We are looking for sections in articles where "image retrieval" is **talked about**, that describe "latent semantic indexing".

Fig. 12. Topic Number 145. An example of the use of a parenthetical information request.

5 System Backend

Once the NLQ has been tagged, chunked and matched to templates it is transformed into a NEXI query using the existing NLPX system. This is a two stage process. First we expanded the content of the query, by deriving phrases based on its lexical properties, such as noun phrases that include adjectives participles. Then we format a NEXi query based upon its instruction, structure and content values. We pass this NEXI query to our existing GPX system for processing, as if they were a standard Ad-hoc query. GPX accepts NEXI queries, and returns a ranked list of XML elements. To produce results GPX collects leaf elements from its index and dynamically creates their ancestors. GPX's ranking scheme rewards leaf elements with specific terms and penalises leaf elements with common terms. It also rewards ancestors with multiple relevant children and penalises ancestors with a single relevant child. Finally, phrases are heavily rewarded, where an occurrence of a phrase in a result is defined as all phrase words in the query occurring in the leaf element, even if they do not occur continuously. To perform focused retrieval GPX selects the nodes highest ranked nodes along an element tree and discounts any lower ranked overlapping elements. A more comprehensive description of GPX can Be found in our accompanying paper as well as earlier work [5]

6 INEX 2005

6.1 INEX 2005 Submissions

We made 4 submissions to the INEX 2005 NLP track, 2 each for the CO and CAS topics. For all the submission we followed the same principle steps as outlined above, however, we prepared 2 variations of each task: one that recognised queries' structural constraints and one that ignored them. In effect this produced not only traditionally CO and CAS submissions, but also hybrid CO as CAS (i.e. CO+S) and CAS as CO submissions. Overall, 1 submission was made in each of the CO and CO+S tasks and 2 submissions were made in the CAS task.

6.2 INEX 2005 Results

This section discusses the results of out 4 submissions. We present results for the CO+S and CAS tasks. Since the CO task did not specify any structural constraints its results are not included here. Due to length constraints we have not included any graphs, however, we have included tables.

6.2.1 CO+S Results. Tables 1 -4 present the results for the CO+S tasks. In each of the tables we have compared the results of the NLP submission with the results of a baseline that used the original NEXI title as input. We present results for the Focused, Thorough and FetchBrowse tasks. Note that for the Focused task 2 submissions were produced, one that accepted the highest-ranking element on an element path (Focused) and one that accepted leaves.

Table 1. The Results of the COS Focused Submissions

Metric	Quant	Score NLPX Focused	Score Base	Ratio (N/B)
nXCG[10]	Strict	0.1132	0.0692	1.6358
	Gen	0.2017	0.1569	1.2855
nXCG[25]	Strict	0.1031	0.0712	1.4480
	Gen	0.1759	0.1309	1.3438
nXCG[50]	Strict	0.1670	0.0674	2.4777
	Gen	0.1578	0.1240	1.2726
MAP	Strict	0.0273	0.0126	2.1667
	Gen	0.0647	0.0595	1.0874

Table 2. The Results of the COS Focused (Leaves) Submissions

Metric	Quant	Score NLPX Leaves	Score Base	Ratio (N/B)
nXCG[10]	Strict	0.1115	0.0538	2.0725
	Gen	0.1960	0.1328	1.4759
nXCG[25]	Strict	0.1220	0.1209	1.0091
	Gen	0.1717	0.1354	1.2681
nXCG[50]	Strict	0.1056	0.1087	0.9715
	Gen	0.1670	0.1212	1.3779
MAP	Strict	0.0391	0.0274	1.4270
	Gen	0.0710	0.0640	1.1094

Table 3. The Results of the COS Thorough Submissions

Metric	Quant	Score NLPX	Score Base	Ratio (N/B)
nXCG[10]	Strict	0.0482	0.0235	2.0511
	Gen	0.1756	0.1675	1.0484
nXCG[25]	Strict	0.0634	0.0591	1.0728
	Gen	0.1783	0.1665	1.0709
nXCG[50]	Strict	0.0855	0.0789	1.0837
	Gen	0.1758	0.1662	1.0578
MAP	Strict	0.0020	0.0021	0.9524
	Gen	0.0609	0.0608	1.0016

Table 4. The Results of the COS FecthBrowse Submissions

Metric	Quant	Score NLPX	Score Base	Ratio (N/B)
MAP	Strict	0.0009	0.0026	0.3462
	Gen	0.0146	0.0510	0.2863

As the results show NLPX performs strong in comparison with the baseline. In particular in the Focused and Thorough tasks it outperforms the baseline in almost all of the metrics. This is a significant improvement on last year, when NLPX was unable to outperform the baseline. Unfortunately, it the baseline outperformed NLPX in the FetchBrowse task, however, this was a trend experienced by all participants in the NLP track. Further research will need to be conducted to see why this occurs.

8.2.2 CAS Submissions. Tables 5 -8 present the results for each of the CAS tasks. In each of the tables we have compared the results of the NLP submission with the results of a baseline that used the original NEXI title as input. We present results for the Focused, Thorough and FetchBrowse tasks.

Table 5. The Results of the SSCAS Submissions

Metric	Quant	Score NLPX Leaves	Score Base	Ratio (N/B)
nXCG[10]	Strict	0.1250	0.1000	1.2500
	Gen	0.2374	0.2517	0.9432
nXCG[25]	Strict	0.1378	0.1578	0.8733
	Gen	0.2859	0.2885	0.9910
nXCG[50]	Strict	0.3738	0.1528	2.4463
	Gen	0.3050	0.3681	0.8286
MAP	Strict	0.0218	0.0251	0.8685
	Gen	0.1087	0.1357	0.8010

Table 6. The Results of the SVCAS Submissions

Metric	Quant	Score NLPX Leaves	Score Base	Ratio (N/B)
nXCG[10]	Strict	0.0800	0.0400	2.0000
	Gen	0.1100	0.0848	1.2972
nXCG[25]	Strict	0.0662	0.0662	1.0000
	Gen	0.1100	0.1081	1.0176
nXCG[50]	Strict	0.0582	0.0662	0.8792
	Gen	0.1150	0.1086	1.0589
MAP	Strict	0.0070	0.0078	0.8974
	Gen	0.0323	0.0282	1.1454

Table 7. The Results of the VSCAS Submissions

Metric	Quant	Score NLPX Leaves	Score Base	Ratio (N/B)
nXCG[10]	Strict	0.1333	0.1167	1.1422
	Gen	0.2423	0.2039	1.1883
nXCG[25]	Strict	0.1133	0.1267	0.8942
	Gen	0.2446	0.2531	0.9664
nXCG[50]	Strict	0.1833	0.1200	1.5275
	Gen	0.2491	0.2493	0.9992
MAP	Strict	0.0090	0.0107	0.8411
	Gen	0.0646	0.0620	1.0419

Table 8. The Results of the VVCAS Submissions

Metric	Quant	Score NLPX Leaves	Score Base	Ratio (N/B)
nXCG[10]	Strict	0.1222	0.1222	1.0000
	Gen	0.2197	0.2520	0.8718
nXCG[25]	Strict	0.1644	0.1257	1.3079
	Gen	0.2136	0.2281	0.9364
nXCG[50]	Strict	0.2698	0.1142	2.3625
	Gen	0.2110	0.2080	1.0144
MAP	Strict	0.0079	0.0086	0.9186
	Gen	0.0758	0.0708	1.0706

Once again NLPX performs strongly against the baseline. Most of the time NLPX either outperforms the baseline or achieves a score that is very close to the base (>90%). Once again this is a significant improvement over the previous years attempts ant verifies our belief that natural language interfaces have the potential to be a viable alternative to formal languages.

8 Conclusion

Here we presented the improvements made to our existing XML-IR NLP interface. Overall three improvements were made: the addition of more special connotations, application of shallow parsing and inclusion of more templates. These improvements have resulted in a performance increase in comparison with our previous system, both in CO and CAS queries, where our backend system using NLPX performs comparably to – and many times outperformed – our baseline system using NEXI input. This validates the claim that natural language is a potential viable alternative to formal query languages in XML-IR.

Reference

1. Abney, S.: Parsing by Chunks. In: Principle-Based Parsing. Kluwer Academic Publisher (1991)
2. Brill, E.: A Simple Rule-Based Part of Speech Tagger. In: Proceedings of the Third Conference on Applied Computational Linguistics (ACL), Trento, Italy (1992) 152–155
3. Clark J., DeRose, S.: XML Path Language XPath Version 1.0. W3C Recommendation, The World Wide Web Consortium, November 1999 available at http://www.w3.org/TR/xpath.
4. Fox, C: Lexical Analysis and Stoplists. In: Frankes, W.B., Baeza-Yates, R. (eds.): Information Retrieval: Data Structures and Algorithms, Prentice-Hall, Upper Saddle River, New Jersey, United States of America (1992) Chapter 7 102-130.
5. Geva, S.: GPX - Gardens Point XML Information Retrieval INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S., Szlavik Z. (eds.): Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl, Germany, December 6–8, 2004, Revised Selected Papers. Lecture Nodes in Computer Science, Vol 3493. Springer-Verlag, Berlin Heidelberg New York (2005) 221–222
6. Larson, R.: XML Element Retrieval and Heterogenous Retrieval: In Pursuit of the Impossible? In Proceedings of INEX 2005 Workshop on Element Retrieval Methodology, Glasgow, Scotland (2005) 38-41.
7. O'Keefe, R., Trotman, A.: The Simplest Query Language That Could Possibly Work, In: Fuhr N., Malik, S. (eds.): INEX 2003 Workshop Proceedings. Dagstuhl, Germany (2003) 167–174
8. Ramshaw, L. Marcus, M.: Text Chunking Using Transformation-Based Learning, In: Proceedings of the Third Workshop on Very Large Corpora (1995) 82-94.
9. Trotman, A., Sigurbjörnsson, B.: NEXI: Now and Next, In: Fuhr, N., Lalmas, M., Malik, S., Szlavik Z. (eds.): Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl, Germany, December 6–8, 2004, Revised Selected Papers. Lecture Nodes in

Computer Science, Vol 3493. Springer-Verlag, Berlin Heidelberg New York (2005) 410–423

10. Woodley, A., Geva, S.: NLPX: An XML-IR System with a Natural Language Interface, In: Bruza, P., Moffat, A., Turpin, A (eds.): Proceedings of the Australasian Document Computing Symposium, Melbourne, Australia (2004) 71–74.
11. Woodley, A., Geva, S.: NLPX at INEX 2004, In: Fuhr, N., Lalmas, M., Malik, S., Szlavik Z. (eds.): Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl, Germany, December 6–8, 2004, Revised Selected Papers. Lecture Nodes in Computer Science, Vol 3493. Springer-Verlag, Berlin Heidelberg New York (2005) 393–406
12. Woodley, A., Geva, S.: Applying Error-Driver Transformation-Based Learning to Structured Natural Language Queries, In: Proceedings of the 2005 International Conference on Cyberworlds. IEEE Computer Society, to appear in 2005.
13. van Zowl, Roelof, Bass, J., van Oostendorp, H., Wiering, F.: Query Formulation for XML Retrieval with Brick. In Fuhr, N., Lamas, M., Trotman, A. (eds.): In Proceedings of INEX 2005 Workshop on Element Retrieval Methodology, Glasgow, Scotland (2005) 75-83.

From natural language to NEXI, an interface for INEX 2005 queries

Xavier Tannier

École Nationale Supérieure des Mines de Saint-Etienne
158 Cours Fauriel
F-42023 Saint-Etienne, France
tannier@emse.fr

Abstract. Offering the possibility to query any XML retrieval system in natural language would be very helpful to a lot of users. In 2005, INEX proposed a framework to partipants that wanted to implement a natural language interface for the retrieval of XML documents, independantly of the search engine. This paper describes our contribution to this project and presents some opinions concerning the task.

1 Introduction

1.1 Motivation

Asking a question in everyday language ("natural language") and getting a relevant answer is what the everyday user really miss in the process of Information Retrieval (IR). Moreover, as natural language is the best way so far to explain our information need, using it should help a system if the query was analysed correctly. However, at present, Natural Language Processing (NLP) techniques are not developed enough to come close to the human perception of language, and actual results are not yet up to what we could expect [1, 2].

In the case of "traditional" IR, where documents are considered as text only (*flat* documents), classical search engines need a query composed of a list of keywords. Writing such a query is quite simple for the casual user, and the value added by NLP approaches is not worth the complexity of these techniques.

On the other hand, many natural language interfaces (NLI) for querying structured documents (databases) have been developed, most of them transforming natural language into Structured Query Language (SQL) [3, 4, 5]. This is probably because the benefits that can be gained in that case are much higher than in traditional information retrieval. Indeed, SQL (and any structured query language used for XML retrieval as well) is hardly usable by novice and casual users. Moreover such languages impose to know the structure of the database (or of the documents).

But database querying is a strict interrogation. It is not information retrieval. The user knows what kind of data is contained in the database, the information need is precise, and a correct query necessarily leads to a correct answer. This means that the natural language analysis must interpret the query perfectly and unambiguously, failing which the final answer is incorrect and the user disatisfied. For this reason notably, natural language interfaces for databases only apply to very restricted domains. Even in these domains, the answer to a query is often *"I did not understand your query"*.

XML retrieval stands between these two domains (see Tab. 1). *Document-oriented* XML files [6], as well as databases, contain some structural information, and the use of a NLI would be justified. But in XML IR, as in traditional IR, the information need is loosely defined and there is no perfect answer to a query. A NLI is then a part of the retrieval process, and thus it can interpret some queries imperfectly, and still return useful results. The problem is then made "easier" to solve... and we can even imagine an interface getting better results than manual queries (which is a nonsense in databases). Moreover more general applications can be designed. In return, such an interface has to be very robust, and all queries must be analysed, even imperfectly. It is not conceivable that the system returns no answer because it did not understand the question.

Table 1. Some features of flat, semi-structured and structured documents in an Information Retrieval point of view.

	Flat documents	*semi-structured documents (XML)*	*structured documents (DB)*
Content	text only	text + structure	structure + data
Information need	text only	content and/or structure	
Query	keywords	**Structured query languages**	
Interpretation	**loose (IR)**		strict

1.2 INEX and Natural Language Tasks

The INitiative for Evaluation of XML Retrieval (INEX) aims at evaluating the effectiveness of Information Retrieval systems for XML documents. The INEX collection groups a set of 16819 articles from the IEEE Computer Society, written in XML, with a set of topics and human assessments on these topics.

In 2005 campaign, two different types of topics have been designed [7]:

- *Content Only + Structure* (CO+S) topics, as indicated by their name, refer only on textual content, but the user can nevertheless add some structural hints to help the system.
- *Content And Structure* (CAS) topics allow a user that know the structure of the documents to formulate constraints on structural elements that he/she wants to be searched for.

We participated to the campaign for both categories, but our approach focuses principally on CAS topics. A simplified example of INEX 2005 topic is given in Fig. 1. The element `castitle` is written in NEXI [8], a formal language for XML retrieval.

```
<inex_topic topic_id="203" query_type="CO+S" ct_no="5">

    <title>code signing verification</title>
    <castitle>//article//sec[about(., code signing verification)]</castitle>
    <description>
        Find documents or document components, most probably sections, that
        describe the approach of code signing and verification.
    </description>
    <narrative>
        I am working in a company that authenticates a wide range of web data
        base applications from different software vendors. [...] To be relevant,
        a document or document component must describe the whole process of
        code signing and verification, which means [...]
    </narrative>

</inex_topic>
```

Fig. 1. Example of INEX 2005 topic. The `title` element is used for Content-Only search, `castitle` for structural hints and CAS representation in NEXI. `description` is used by Natural Language Processing tasks participants, while the `narrative` is reserved for human assessors.

NEXI CAS queries have the form //**A[B]**//**C[D]** where A and C are paths and B and D are filters. We can read this query as *"Return C descendants of A where A is about B and C is about D"*. B and D correspond to disjunctions or conjunctions of 'about' clauses **about(//E, F)**, where E is a path and F a list of terms. The `'title'` part of Fig. 1 gives a good example of a query formulated in NEXI. More information about NEXI can be found in [8].

In 2005 INEX campaign, two different tasks aimed to involve Natural Language Processing. In the first one, called NLQ (Natural Language Queries), participants had to consider only the `description` part of the topics and to return a set of XML elements (or doxels) corresponding to the request. No matter how they performed their search, or where the NLP was used. The evaluation of NLQ systems was the same as for the *ad-hoc* task.

In the second one, NLQ2NEXI, on which this paper focuses, the aim was to translate natural language queries into `title` (keyword list) and `castitle` (NEXI) elements from the `description`. Here the idea is to build a generic interface that could used by any retrieval system reading NEXI queries. Automatically generated topics have then been run with a search engine E provided by the organizers. In this case, the evaluation is twofold:

1. a comparison between the effectiveness of each NLQ2NEXI system.
2. a comparison between each system and a baseline obtained by running the system E on initial (manual) topics, in order to quantify the trade-off in performance.

2 Natural Language Query Analysis

In our approach, requests are analysed through several steps:

1. A part-of-speech (POS) tagging is performed on the query. Each word is labeled by its word class (*e.g.:* noun, verb, adjective...). To carry out this task we chose the open-source free tool TreeTagger [9].
2. A POS-dependant semantic representation is attributed to each word. For example the noun *'information'* will be represented by the predicate *information(x)*, or the verb *'identify'* by *evt*(e_1, *identify)*.
3. Context-free syntactic rules describe the most current grammatical constructions in queries and questions. Low-level semantic actions are combined with each syntactic rule. Two examples of such operations, applied to the description of Topic 130 (INEX 2004: *"We are searching paragraphs dealing with version management in articles containing a paragraph about object databases."*), are given in Fig. 2. The final result is a logical representation shown in the left part of Fig. 3. This representation is totally independant from the queried corpus, it is obtained by general linguistic operations.
4. The semantic representation is then reduced with the help of specific rules:
   - a recognition of some typical constructions of a query (*e.g.: Retrieve + object*) or of the corpus (*e.g.: "an article written by [...]"* refers to the tag *au – author*);
   - and a distinction between semantic elements mapping on the structure and, respectively, mapping on the content;

   Figure 3 shows the specific rules that apply to the example.
5. A treatment of relations existing between different elements;
6. The construction of a well-formed NEXI query.

Steps 1 to 5 are explained in more details in [10], as well as necessary corpus knowledge and the effect of topic complexity on the analysis. The representation obtained at the end of Step 5 does not depend on any retrieval system or query language. It could be transformed (with more or less information loss) into any existing formal language.

Transformation process from our representation to NEXI is not straightforward. Remember that a NEXI query has the form //**A[B]**//**C[D]**.

- At content level, linguistic features (like `noun_modifier` in the example) cannot be kept and must be transformed in an appropriate manner (see Sect. 3).
- At structural level, a set of several tag identifiers (that can be DTD tag names or wildcards) has to be distributed into parts A, B, C and D, that we respectively call support requests, support elements, return requests and return elements. These four parts A, B, C and D are built from our representation (Fig. 3) in the following way:
  - C is the 'framed' (selected) element name (see Fig. 3 and its caption);
  - D is composed of all C children (relation *contains*) and their textual content (relation *about*);

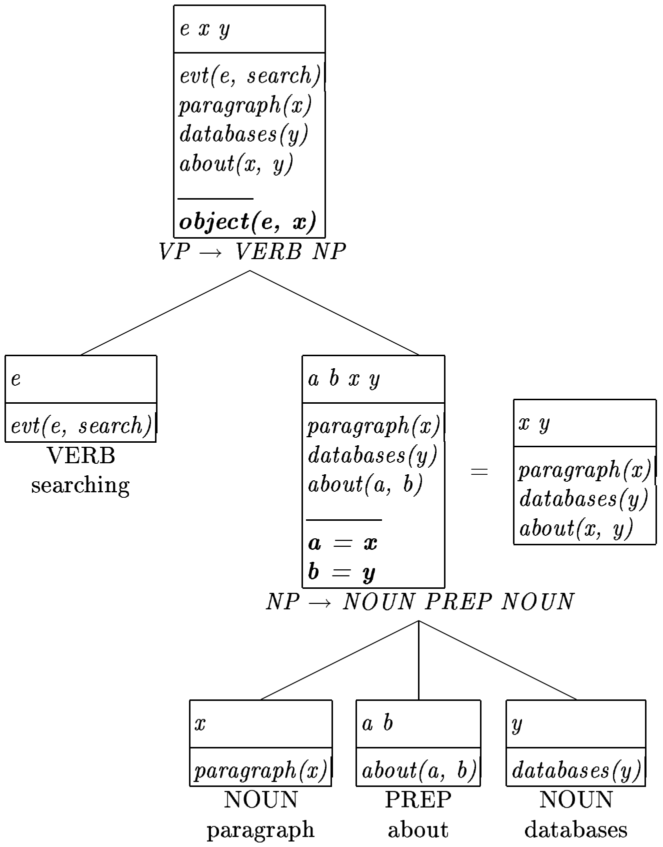


Fig. 2. Example of rule application for the verbal phrase "*searching paragraphs about databases*" (rules *NP → NOUN PREP NOUN* and *VP → VERB NP*). Basic semantic representations are attributed to part-of-speeches (leaf components). When applying syntactic rules, components are merged and semantic actions are added (here identity relations and verbal relation predicate – bold predicates).

- A is the highest element name in the DTD tree, that is not C or one of its children;
- B is composed of all other elements and their textual content.

Wildcard-identified tags of the same part are merged and are considered to be the same element. See an example in Sect. 4.

3 Noun phrases

Our system generates some linguistic-oriented predicates. The main ones are `np_property`, `noun_modifier` and `adjective`. NEXI format requires the 'about'

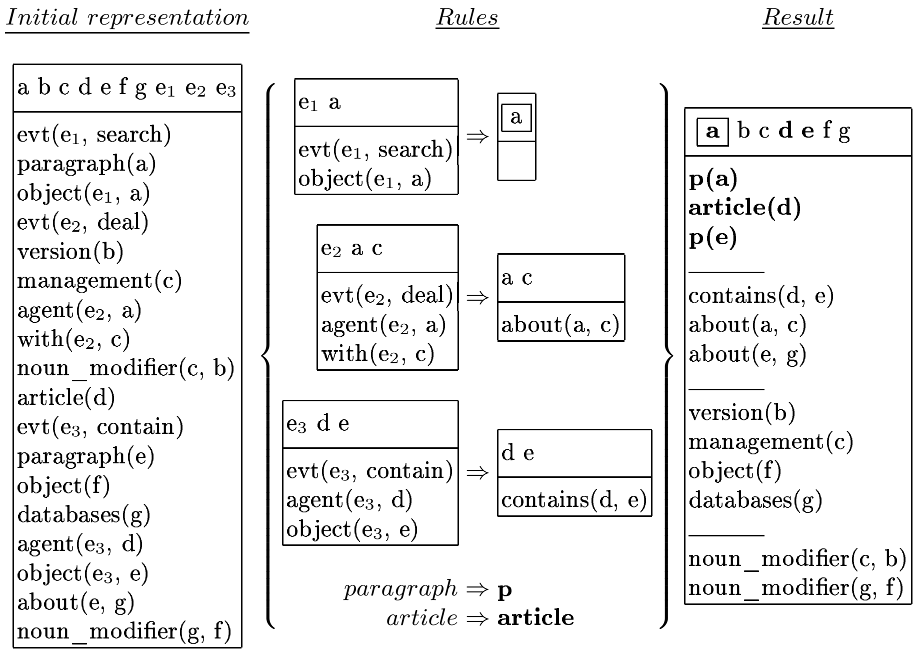


Fig. 3. The semantic analysis of Topic 130 (left), is reduced by some generic rules (center), leading to a new representation (right). Bold predicates emphasize words representing XML tag names and the framed letter stands for the element that should be returned to the user. The first three rules deal with verbal phrases *"to search sth"*, *"to deal with sth"* and *"to contain sth"*.

clauses to contain only textual content. Phrases can be represented with quotation marks. We chose to consider only noun phrases treatment here, because other relations are translated in a straightforward way.

From an IR point of view, noun phrases have the general form [11]:

$$NP \rightarrow det^{*}\ pre^{*}\ head\ post^{*}$$

. . . Where *det* is a determiner, *pre* (*premodifier*) is an adjective, a noun or a coordinated phrase, *head* is a noun and *post* (*postmodifier*) is a prepositional phrase or a relative clause.

In our representation, relations between premodifiers and head nouns are expressed by predicates `noun_modifier` (if the premodifier is a noun) or `adjective` (if the premodifier is an adjective). Prepositional relations between NPs (*i.e.* the form $NP \rightarrow NP_{head}\ PREP\ NP_{post}$) are represented by `noun_property`.

All forms have been considered when analysing the natural language queries, but we distinguished two specific constructions of NPs to build the formal queries.

3.1 Simple noun phrases

In English, the simplest noun phrases are a succession of adjectives or nouns followed by a head noun:

$$NP \rightarrow (ADJ\ /\ NOUN)+\ NOUN \tag{1}$$

These multi-word terms are less ambiguous than simple nouns, and generally refer to a particular domain [12]. They are not subject to many syntactical variations (see next section), and it is quite probable that such terms representing the same concept have the same form in most occurrences of a collection. For all these reasons, these simple NPs are very interesting in information retrieval. Some examples (extracted from INEX 2005 topics) are given in Tab. 2 (with an additional rule for proper names: $NP \rightarrow PN+$).

With NEXI, phrases are represented between quotation marks. All sequences of words obeying to Rule 1 are then transcribed between quotation marks.

Table 2. Examples of simple noun phrases (rule 1) in INEX 2005 topics.

Topics	Noun phrase
204	“semantic networks” (ADJ NOUN)
231	“graph theory” (NOUN NOUN)
210	“multimedia document models” (NOUN NOUN NOUN)
211	“global positioning systems” (ADJ NOUN NOUN)
204	“Dan Moldovan” (PN PN)

3.2 Complex noun phrases

Nouns or noun phrases linked to each other by prepositions are semantically very significant [13, 14]:

$$NP \rightarrow NP\ (PREP\ NP)+ \tag{2}$$

They occur as frequently as constructions made with Rule 1 (see Tab. 3). However it is quite hazardeous to consider them as a unique multi-word term in the same way. In particular, they are subject to many variations in their form. *Fabre and Jacquemin* [15] distinguished five different simple syntactic forms that could represent the same concept in French. For example, even without semantic variation (as synonymy), the NP *“annotation in image retrieval”* found in Topic 220 can be modified with no or little semantic change into *“annotate images for retrieval”*, *“retrieve annotated images”*, *“annotated image retrieval”*, *“retrieval of annotated images”*, *“images have been annoted for retrieval”*, etc.

Moreover such a phrase does often not occur at all in a relevant element. In a phrase having the form *“NP_1 PREP NP_2”*, we have noted that one of the

Table 3. Examples of complex noun phrases (Rule 2) in INEX 2005 topics.

Topics	Noun phrase
208	history of Artificial Intelligence
216	the architecture of a multimedia retrieval system
217	user-centered design for web sites
219	the granularity of learning objects
220	annotations in image retrieval
233	development of synthesizers for music creation
276	evaluation measure for clustering

sub-NPs represents the *context*, while the other one represents the *subject* of the current sentence. The role of each part depends on the structure of the document.

For example, suppose we look for an element dealing with *"evaluation measure for clustering"* (Topic 276). In an article about *clustering* on the whole, we just need to look for the term *"evaluation measure"*. Inversely, an article about *evaluation measures* in general must contain an element treating *"clustering"*.

We have noted, after 2004 campaign, that this issue was an important source of mis-retrieval for search engines. In the case of topic descriptions containing NP_1 $PREP$ NP_2, where NP_2 was the context in most documents, many retrieved doxels contained NP_1 in a bad context, and then were not relevant. For example, a search for *"navigation systems for automobiles"* (Topic 128) returned many doxels about navigation systems in planes or ships in the first ranks.

In this case, to remedy this problem, we would like to perform a *contextual research* (for *"navigation systems"* in the context of a section or an article about *"automobiles"*, or inversely), but also a *conditional research* within a single doxel (if a doxel is relevant with *"automobiles"*, then check for *"navigation systems"*).

Unfortunately this kind of features can hardly be represented with a single NEXI query. Even so we tried to simulate such a behaviour. We noticed that the most frequent configuration was "NP_1 in the context of NP_2" when the topic description contained a NP_1 $PREP$ NP_2 phrase. We decided to translate such NPs in the following way:

- Contextual search: Addition of NP_2 into a support part concerning the whole article (root element).
- Conditional search: Addition of a sign '+' before NP_2 in the current part.

For example, *"a paragraph about navigation systems for automobiles"* can be translated into:

```
/article[about(., automobiles)]//p[about(., "navigation systems")
                                   AND about(., +automobiles)]
```

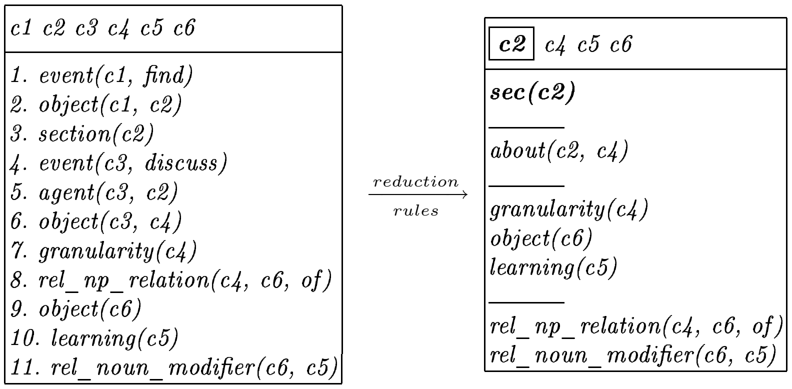
In our tests with INEX 2004 collection, this approach led to a increase in precision of about 10 %. But then the support element construction is quite artificial, and this is done to the detriment of strict evaluation metrics (strict quantization and strict interpretation of target and/or support element requirements [16]). By choosing this strategy we admit that we focus principally on vague interpretation and generalised quantization.

4 Example

We give here a significant example, with the analysis of a slightly simplified version of Topic 219 (INEX 2005). Several syntactic parsings could be possible for the same sentence. In practice a "score" is attributed to each rule release, depending on several parameters In our sample topic only the best scored result is given.

(219) Find sections that discuss the granularity of learning objects.

Figure 4 shows the three major steps of the analysis of this topic. The left frame represents the result of Step 3 (see Sect. 2). Some IR- and corpus-specific reduction rules are then applied and lead to right frame: the term *section* is recognized as tag name **sec** (line 3); the construction *"c2 discusses c4"* is changed into **about(c2, c4)** (lines 4 to 6). The other relations are kept. Translation into NEXI is performed as explained above.



```
//article[about(., "learning objects")]//sec[about(., granularity) AND
                          about(., +"learning objects")]
```

Fig. 4. Semantic representations of Topic 219, and automatic conversion into NEXI.

5 Limits

5.1 Limits of the task

Translation of natural language queries into a formal language like NEXI encounters some limits, mainly due to the fact that the natural language interface cannot give some specific instructions to the retrieval system. The formal language, if not especially designed for this aim, is a pivot preventing from any "communication" between both systems. For example, it is not possible to consider the following features within single NEXI queries[1]:

- NEXI does not allow to perform any conditional search (see Sect. 3.2). The use of '+' sign is not semantically reliable and is often not considered by search engines.
- NEXI cannot either deal with contextual search: A reference to the context occurs preferentially before the retrieved element, in the paragraph preceding it, or in the introduction of the section, etc. Directly refering to the article as a support part of the query (as we did) is too vague.
- NEXI does not bring any proximity operators for terms or structure (but retrieval models can compensate, many systems allow a flexible treatment of phrases [17], and some consider the proximity of doxels [18]).
- It is not possible to represent non-hierarchical relations between elements with NEXI (precedence for example).
- Finally, NEXI is only a query language. It is not designed to deal with any linguistic features. With the linguistic analysis, the interface finds some interesting relations between terms (or elements), as semantic relations (agent, object, etc.), but the translation forces us to give this knowledge up.

On the one hand, formal languages will always stay more precise than natural languages. If sometimes the system with a NLI outperforms the same system with a hand-made NEXI query, this is because the interface found a better, more complete and/or more adequate way to represent the information need. On the other hand, for all the reasons above, the use of these formal languages, if they are not thought with this aim in mind, leads to some loss of information.

If an interface is very interesting because it can be "plugged" to (hopefully) any kind of formal languages, and then be applied to many existing systems, it would also be worthwhile to go further and to build a system with a self-made pivot or without pivot at all.

5.2 Limits of the evaluation

In the *ad-hoc* task, the input is constant and the retrieval systems are different. To evaluate these systems we look at their output (a ranked list of XML elements). In NLQ2NEXI task, the challenge is precisely to produce the input, and the evaluation is performed indirectly, through the use of a search engine that is common to all participants (different inputs, same system). This way we can make sure that the differences in retrieval performance are really due to the quality of the input, and it becomes possible to compare all NLQ2NEXI systems with each other.

[1] In addition to this list, NEXI is not designed to deal with many database-oriented constraints, but we are not either interested by this aspect.

Another way to evaluate interfaces is to compare them with a manual baseline. The same system is run on official NEXI queries[2] (manually written by the author of each topic). In this case, automatic processes are compared with a manual process. Like all human interventions (and IR evaluation is full of them), this introduces a new bias: automatic systems are compared with a query built by a given person at a given time. Probably some different manual translations of the topic description would have led to better results. Moreover, many CO+S topics do not have any NEXI `castitle`[3].

Finally, manual translations from description to NEXI are not always faithful, even if this is much better than it was in 2004 [19]. In particular, many CAS subtopics seem to have a problem with NEXI constraints. Topic 251 is characteristic of this issue:

(251) We are searching paragraphs which are descendant of a section dealing with web information retrieval.

In the official transcription of the description into NEXI, the paragraphs are considered to be dealing with web information retrieval:

```
//article//sec//p[about(., web retrieval)]
```
[4]

Even if this interpretation is syntactically correct, it seems obvious that any human people would understand that the section is concerned by the verbal phrase (*dealing with web information retrieval*):

```
//article//sec[about(., web retrieval)]//p
```

But this form is not correct in NEXI (where the returned element must contain an *about* clause [8]).

The narrative part of this topic confirms the author's NEXI title, but adds to the confusion: *"the paragraphs which are descendants of section describing the topic related to web information retrieval are also regarded as relevant. However, compared with paragraphs described above, these are considered less relevant".*

5.3 Limits of our system

The following is a non-exhaustive list of problems encountered by our natural language interface. In our opinion, these issues represent the most important factors that make the system not work well for some topics. We do not broach here the "usual" difficulties that NLP has in traditional information retrieval (spelling mistakes, noise produced by non query terms, anaphoras, pragmatic issues...) , but rather those that are specific to structural constraints.

[2] What we call the "official" NEXI title of a topic is the query proposed by the author (see the example of Fig. 1). This NEXI query is used by the INEX *ad-hoc* task participants.

[3] Besides, a study on the performances of automatic NEXI titles compared to CO official titles would probably be very interesting.

[4] By the way we can note that *"web information retrieval"* has been replaced by *"web retrieval"*.

Lexical ambiguity. Classical lexical problems in IR are semantic relations between different words (synonymy, hyponymy, etc.) and words that have multiple meanings (homographs). Homographs raise a new problem in XML retrieval, where some words can be understood as normal content-based query terms, but also as tag names (or a synonym of a tag name). Using a simple dictionnary of synonyms to detect references to tag names is obviously not enough. For example the words *"document"* and *"information"* are, most of the time, used for refering to XML elements (*"Find information/documents about"*). But how to deal with a query about *"multimedia document models"* (Topic 210) or *"incomplete information"* (Topic 224)? What if the query is *"Retrieve information about information retrieval"*? Actually it does not seem so difficult to handle this with specific syntactic features, but we clearly under-estimated this issue so far.

Syntactic ambiguity. Some pretty convoluted queries are quite difficult to analyse properly:

(145)We are looking for paragraphs in articles (about information retrieval)$_{PP}$ (dealing with relevance feedback)$_{VP}$.

A human can understand without (too much) difficulty that a relevant paragraph should deal with relevance feedback (*VP* linked to the noun *"paragraph"*):

```
//article[about(.,information retrieval)]//p[about(.,relevance feedback)]
```

But this is a problem for our system, that tends to link phrases to the closest element as possible (in the absence of other constraints). In this case both *PP* (prepositional phrase) and *VP* (verbal phrase) are linked to the noun *"articles"*[5]. This kind of mistakes in structural constraints may result in very bad performances for these complex queries.

Corpus-dependant knowledge. Throughout the query analysis, we use several kinds of information about the corpus, among which the DTD (and the terms associated with tag names), but also some specific linguistic constructions. For example, as shown by Fig. 5, a query about *"information by Moldovan"* (Topic 204) implicitely refers to an author (tag `'au'` in INEX collection); *"works citing Beaza-Yates"* (Topic 280) introduces a bibliographic element. All these rules are necessary to analyse many queries properly, but are an obstacle to the extension of the tool to general corpora, or to heteregenous collections[6].

[5] In this particular case this solution does not lead to any well-formed NEXI query, so the correct NEXI form is finally selected.

[6] Note that these rules are structure-specific, but not domain-specific (in the case of INEX, this means that no rules have been set up especially for computer science information retrieval).

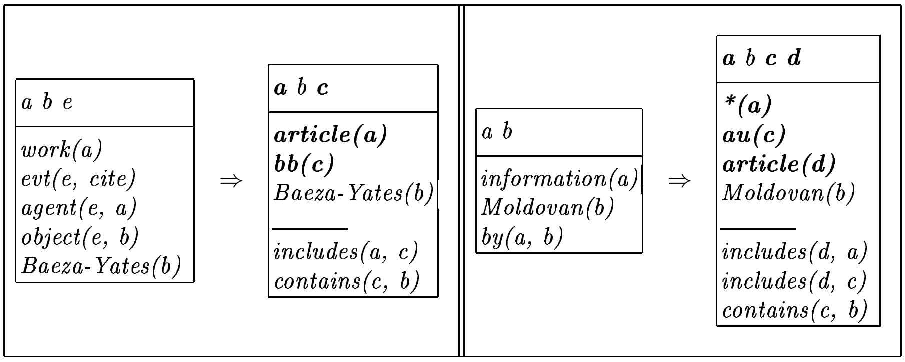


Fig. 5. Examples of corpus-dependant rules, applied on *"works citing Baeza-Yates"* (left, Topic 280) and *"information by Moldovan"* (right, Topic 204).

6 Results

Results for NLQ2NEXI task are very good. For all tasks, in almost all metrics, the baseline (see Sect. 5.2) is outperformed by at least one participant run, and often by all participants. In the curves that we chose (Fig. 6), the difference between the best run and the baseline is between 2 % (for strict structural evaluation) and 35 % (for CO+S), and this is quite representative of the 42 result graphs.

The second comment is that participants do not get their best results in the same sub-tasks: While we (Ecole des Mines de Saint-Etienne) obtain the best scores in CAS with a vague interpretation of elements (6.a and 6.b – and we expected to be more successful there than in other tasks, see Sect 3.2), University of Klagenfurt performs better in strict interpretation (6.c) and Queensland University of Technology gets the best results in CO task (6.d). The approaches are therefore quite different, and this promises very instructive and constructive confrontations in the future.

7 Conclusion

INEX 2005 NLQ2NEXI task proves that the help brought by an natural language interface is very effective. NEXI queries that are automatically obtained from a description in English lead to better results than manual queries, yet written by experts. This is the proof that natural language explanations of an information need are not only easier to formulate, but also more effective. The results also confirm the assumptions made in the introduction: building a natural language interface for XML retrieval is much different than doing it for database querying or traditional IR.

Moreover, techniques used by participants are quite different; Teams have a lot to learn from each other, and global results should improve a lot in the future.

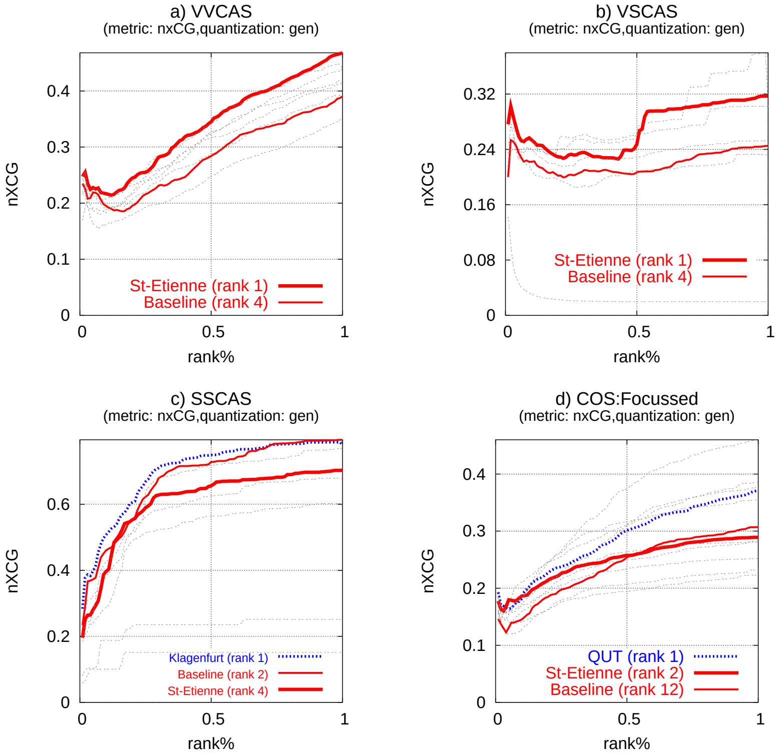


Fig. 6. Official NLQ2NEXI results for VVCAS, VSCAS, SSCAS and COS.Focussed tasks, normalized extended cumulated gain, generalised quantization [16].

But each technique produces good scores for a given task to the detriment of another one, and the best way to progress is probably to define a new model taking all what we need into account (like conditional and contextual searches proposed in this article).

References

[1] Strzalkowski, T., Lin, F., Wang, J., Perz-Carballo, J.: Evaluating Natural Language Processing Techniques in Information Retrieval. [20] 113–145

[2] Sparck Jones, K.: What is the role of NLP in text retrieval? [20] 1–24

[3] Androutsopoulos, I., G.D.Ritchie, P.Thanisch: Natural Language Interfaces to Databases – An Introduction. Journal of Natural Language Engineering **1** (1995) 29–81

[4] A.Copestake, Jones, K.S.: Natural Language Interfaces to Databases. The Knowledge Engineering Review **5** (1990) 225–249

[5] Perrault, C., Grosz, B.: Natural Language Interfaces. Exploring Articial Intelligence (1988) 133–172

[6] Fuhr, N., Großjohann, K.: XIRQL: A Query Language for Information Retrieval in XML Documents. In Croft, W., Harper, D., Kraft, D., Zobel, J., eds.: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York City, New York, USA, ACM Press, New York City, NY, USA (2001) 172–180

[7] Sigurbjörnsson, B., Trotman, A., Geva, S., Lalmas, M., Larsen, B., Malik, S.: INEX 2005 Guidelines for Topic Development (2005) http://inex.is.informatik.uni-duisburg.de/2005/internal/pdf/TD05.pdf.

[8] Trotman, A., Sigurbjörnsson, B.: Narrowed Extended XPath I (NEXI). [21] 16–40

[9] Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In: International Conference on New Methods in Language Processing. (1994)

[10] Tannier, X., Girardot, J.J., Mathieu, M.: Analysing Natural Language Queries at INEX 2004. [21] 395–409

[11] Arampatzis, A., van der Weide, T., Koster, C., van Bommel, P.: Linguistically-motivated Information Retrieval. In Kent, A., ed.: Encyclopedia of Library and Information Science. Volume 69. Marcel Dekker, Inc., New York, Basel (2000) 201–222

[12] Moreau, F., Sébillot, P.: Contributions des techniques du traitement automatique des langues à la recherche d'information. Technical report, IRISA, France (2005)

[13] Tzoukermann, E., Klavans, J.L., Jacquemin, C.: Effective use of natural language processing techniques for automatic conflation of multi-word terms: the role of derivational morphology, part of speech tagging, and shallow parsing. In: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Philadelphia, PA, USA, ACM Press, New York City, NY, USA (1997) 148–155

[14] Arampatzis, A.T., Tsoris, T., Koster, C.H.A., van der Weide, T.P.: Phrase-based Information Retrieval. Information Processing & Management **34** (1998) 693–707

[15] Fabre, C., Jacquemin, C.: Boosting Variant Recognition with Light Semantics. In: Proceedings of the 18th International Conference on Computational Linguistics, COLING 2000, Saarbrücken (2000) 264–270

[16] Kazai, G., Lalmas, M.: INEX 2005 Evaluation Metrics (2005) http://inex.is.informatik.uni-duisburg.de/2005/inex-2005-metricsv4.pdf.

[17] Geva, S., Leo-Spork, M.: XPath Inverted File for Information Retrieval. In Fuhr, N., Lalmas, M., Malik, S., eds.: Proceedings of the second Workshop of the Initiative for the Evaluation of XML retrieval (INEX), December 15–17, 2003, Schloss Dagstuhl, Germany (2004) 110–117

[18] Sauvagnat, K., Boughanem, M., Chrisment, C.: Searching XML documents using relevance propagation. In: String Processing and Information Retrieval, Padoue, Italy, Springer-Verlag, New York, NY, USA (2004) 242–254

[19] Woodley, A., Geva, S.: NLPX at INEX 2004. [21] 382–394

[20] Strzalkowski, T., ed.: Natural Language Information Retrieval. Kluwer Academic Publisher, Dordrecht, NL (1999)

[21] Fuhr, N., Lalmas, M., Malik, S., Szlàvik, Z., eds.: Advances in XML Information Retrieval. Third Workshop of the Initiative for the Evaluation of XML retrieval (INEX). In Fuhr, N., Lalmas, M., Malik, S., Szlàvik, Z., eds.: Advances in XML Information Retrieval. Third Workshop of the Initiative for the Evaluation of XML retrieval (INEX). Volume 3493 of Lecture Notes in Computer Science., Schloss Dagstuhl, Germany, December 6–8, 2004, Springer-Verlag, New York, NY, USA (2005)

Processing Heterogeneous Collections in XML Information Retrieval

Diego Vinícius Castro Pereira, Klérisson Vinícius Ribeiro Paixão, and
Maria Izabel Menezes Azevedo

Department of Computer Science,
State University of Montes Claros, Montes Claros(MG), Brazil
mimaizabel@gmail.com, klerisson@hotmail.com, diegovcastro@yahoo.com.br

Abstract. By conception, our model explores the diversity of XML markups, having its potential really explored only in heterogeneous collections. To evaluat its performance, we inscribe in INEX 2004 Heterogeneous track with 34 other institutions, and from them, only 5, included ours, submitted runs. In this paper, we describe how the approach we used in INEX 2004 and 2005 process heterogeneous collections, without any mapping of DTDs.

1 Introduction

The conception of our model[3] had as starting point the observation and theoretical confirmation[1] that the markups of the XML[4] document are semantically related to the content that they delimit. We consider the structure of XML standard as a new source of evidence, capable to assist in the identification of the information contained in documents without, however, being a necessary condition to its identification.

In accordance with this premise, our model does not use formal aspects of the XML, such as the DTDs. Our objective is to associate the markups of XML with their content, based on statistical measures similar to those that relate the frequency of terms to the document information, in the standard vector space model.

Thus, by conception, our model explores the diversity of markups of XML, having its potential really avaliated only in heterogeneous collections, where the diversity of structures will allow better linking between the semantic of markups and its content.

To make possible the evaluation of our model, we inscribe in heterogeneous track, in INEX 2004[6]. The operational difficulties had not allowed the conclusion of the stage of judgment and we did not obtain a quantitative evaluation of our model, but we send 6 submissions, and we were one among the 5 institutions that had obtained it.

In this article, we detail how our model processes heterogeneous collections, presenting how is calculated each one of the sub factors introduced in the conventional vectorial model to explore the characteristics of XML language.

The remain of this article is organized in following the way: Section 2 presents a revision of other works directed to the processing of heterogeneous collections. Section 3 details the calculation of each subfactor in heterogeneous collections. Section 4 presents he results and section 5 concludes the article.

2 Related Works

At INEX'04 according to Sauvagnat and Boughanem[11]"the idea behind the heterogeneous track is that information seeker is interested in semantically meaningful answer irrespectively to the structure of documents". So, this idea motivated then to present a model which uses the propagation method of relevance. This model is based in automatically indexing and introduces an interesting query processing that is able to process sub-queries eventually linked logically. For this, the first step is to transform NEXI topics into XFIRM queries. And then, this new query is decomposed into sub-queries. After each sub-queries has been processed the result of each one of them is propagated to generate whole result of query. However, mapping structural conditions from one DTD into other was a problem, and to solve it they presented one DTD built manually, comparing the different DTDs.

Other relevance work was presented by Larson[7] which continued to explore the approach of fusion to XML retrieval. This approach to the Heterogeneous Track was to treat the different collections as separate database with their own DTD, but these distinctions database can be treated as a single database on his system. Other important work described by Larson was the configuration file which could specify some subset of tags to be used with the same meaning, for example `//p`, `//p1`, `//tf` for "paragraphs" searches. Apparently, Larson did not have problems with tags without mean such as Fld001, but he did not mentioned.

Moreover, at "A Test Platform for the INEX Heterogeneous Track"[2] was presented an approach for creating a unified heterogeneous structure from the heterogeneous data source. To build this unified conceptual model first they identify groups of concepts semantically similar. To do this, they used a tool called WordNet develop by Christine Fellbaum[5] which is able to detect similarity between "editor" and "edition", for example. Finally, to treat tags without mean is necessary to capture the DTD comments preceding then and from the descriptions look for the best cluster to put them.

All the participants of Heterogeneous Track had difficulties to treat a single document that was 217 Mb in size. So, to solve this trouble, Larson[7] proposed to treat each of main sub-elements as separated documents, and Lehtonen[8] proposed to divided it into fragments, but he explores a size-based division. Although, large documents are problem at "Het Track", they are not a relevance problem to "Het Track", but to retrieval systems.

3 XML Factor Calculation

On article A Universal Model For XML Information Retrieval published on INEX 2004, we have defined how the standard vector space model statistics will be calculated just adapting them to the element division of information, present on XML documents. On the next paragraphs, we will describe how the factor, *(fxml)*, will process heterogeneous collections to allow the return of elements from documents with independent existence structure of any map between them, given by equation[9][10]:

$$\rho(Q, D) = \sum ti \in Q \cap D \frac{W_q(ti) * W_d(ti) * fxml(ti, e)}{Q * D} \tag{1}$$

We will explore the follow characteristics of XML standard:

- Your nested structure through the Nest Factor *(fnh)*;
- The similarity between the query structure and the documents, through the Structure Factor *(fstr)*;
- The Co-occurrence between terms and markups, through the Co-occurrence Factor *(focr)*.

Taking the expression

$$fxml(ti, e) = fnh(ti, e) * fstr(ti, e) * focr(ti, e) \tag{2}$$

3.1 Nesting Factor

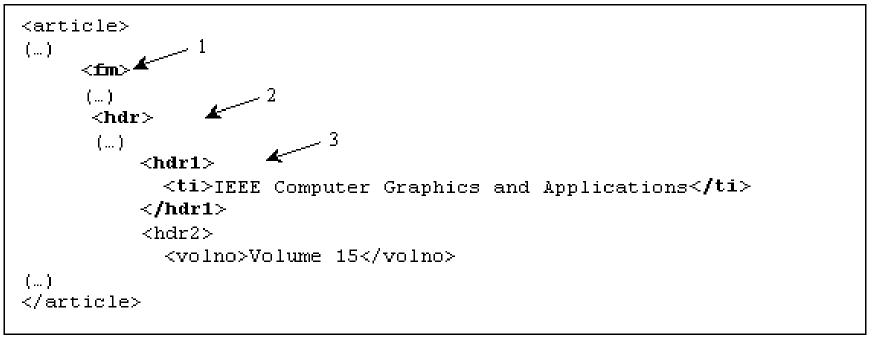
The Nesting Factor expresses the importance of terms considerating it position on the XML tree. Like the example of Augmentation Factor, this factor looks for reduce the term contribution to relevance of it ancestor elements for distant elements in XMLs tree. The factor proposed not have a fixed defined value, but have an inverse relation with the distance between the level of element that contains the term and the ancestor. Is given by:

$$fnh(ti, e) = \frac{1}{(1 + nl)} \tag{3}$$

where, *nl* is the number of levels from element *e* to its sub-element that contains the term *ti*, being able to vary between the following values:

- *fnh(ti,e) = 1*, for terms directly in elements *e*;
- *fnh(ti,e) = 1/nd*, where *nd* are depth of the XML tree.

In Fig. 1, we have *fnh(*`computer,fm`*) = 1/(1+3)*. This factor will reduce the term contribution for relevance of elements more distant, in direction to root of XML tree. It compensate the large frequency of a term in elements more extern by nesting of XML structure, which consider the terms of a sub-elements counted all them ancestors. With no this consideration the more extern elements tend to occupy always the first positions on ranking of important elements showed to user. With prejudice the intern ones. Not showing in it expression any sub-factor about markups from document the difference of collection will not affect it value neither becomes difficulty calculate.



```
<article>
(…)
     <fm>
     (…)
      <hdr>
       (…)
          <hdr1>
            <ti>IEEE Computer Graphics and Applications</ti>
          </hdr1>
          <hdr2>
            <volno>Volume 15</volno>
(…)
</article>
```

Fig. 1. The Nesting Factor

3.2 Structure Factor

The Structure Factor denotes how a query with structural restriction (CAS) are satisfied by context of determinate element. It was defined considerating how big the markups number as big will be your chance in attend the express information necessity on query. Mathematically is given by relation between the common markups restriction query and the total structural restriction on query. Where,

$$fstr(ti,e) = \frac{(common\_markups + 1)}{(nr\_qmarkups + 1)} \tag{4}$$

Where,

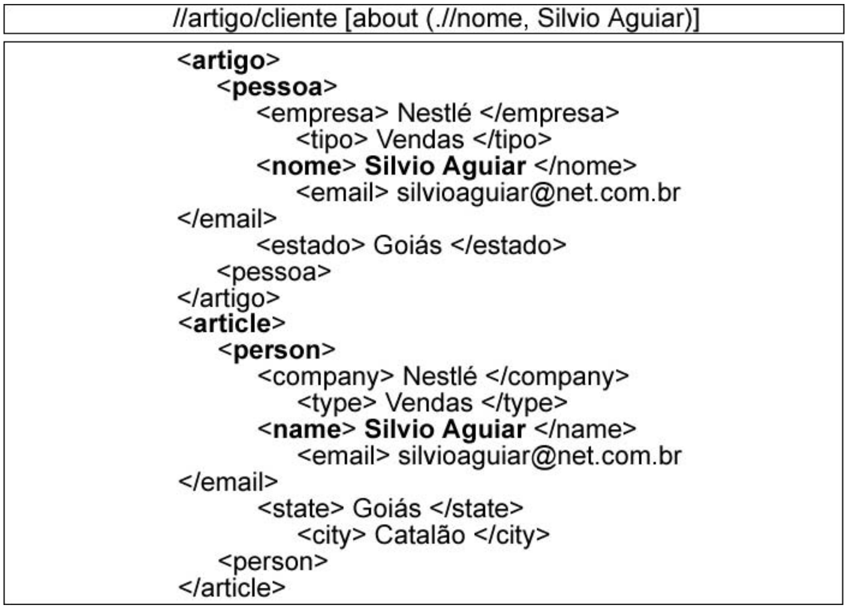
- *common_markups* are the common markups number in the query structural constraints and in the context of element *e* that contains *ti*;
- *nr_markups* are the markups number in the query structural containts.

It can vary from:

- *fstr(ti,e) = 1/(nr_qmarkups+1)*, when no query's constraints markups appears in the context of *ti*;
- *fstr(ti,e) = 1* when all query's structural constraints markups appears in the context of *ti*.

To the query's picture, express in NEXI[12], processed above the heterogeneous collection will have the following results.
To the element `/artigo/pessoa/nome` at the first article:

- *nr_qmarkups = 2;*
- *common_markups = 2;*
  and
- *fstr(ti, e) = (2+1)/(3+1) = 3/4.*



```
//artigo/cliente [about (.//nome, Silvio Aguiar)]
```

```
<artigo>
    <pessoa>
        <empresa> Nestlé </empresa>
            <tipo> Vendas </tipo>
        <nome> Silvio Aguiar </nome>
            <email> silvioaguiar@net.com.br
</email>
        <estado> Goiás </estado>
    <pessoa>
</artigo>
<article>
    <person>
        <company> Nestlé </company>
            <type> Vendas </type>
        <name> Silvio Aguiar </name>
            <email> silvioaguiar@net.com.br
</email>
        <state> Goiás </state>
            <city> Catalão </city>
    <person>
</article>
```

Fig. 2. The Structure Factor

To element `/article/person/name` at the second article:

- *nr_markups = 2;*
- *common_markups = 2;*
  and
- *fstr(ti,e) = (0+1)/(3+1) = 1/4.*

This factor will valorize more the element `/artigo/pessoa/nome` at the first article, where *fstr* is equal *3/4* whose context better attends the structural constraints present in the query. But won't impede the element `/article/person /name` at the second article, where *fstr* is equal *1/4* also returned to user garanting the models application on heterogeneous collections.

It is important in CAS queries, where the user express the elements that will better fit its information need. For CO queries, *fstr* will always be equal 1, because:

- *nr_qmarkups = 0* (CO queries not have a structural restriction);
- *common_markups = 0* (there is not common markups in no structural constraints queries);
  doing
- *fstr (ti,e) = 1.*

Consequentially, in this case the factor *fstr* will not influence the relevance equation.

3.3 CO-occurrence factor

The last factor, Co-occurrence Factor, express the semantic link between markups and its content. For express mathematically this semantic relation, we utilize the standard vectorial model principle on relation terms and documents: as big the frequency in that term appears delimited by determinated markup, as big the semantical link between them. The value this semantical relation, for determinated query, will depends of frequency of terms and markups in collection too. This factor is calculated by equation:

$$focr(ti, e) = cf(ti, e) * idf(ti, e) * N * icf(e) \tag{5}$$

where,

- *cf(ti,e)* is the number of times the markup of element e, denoted by m, delimits a textual content containing term *ti*. In other words, number of co-occurrences of term *ti* and markup m in the collection;
- *idf(ti,e)* is the inverse of the number of elements e that contain *ti*.

So, $cf(ti, e){*}idf(ti, e)$, is the reason between the number of times term *ti* appears with m for the numbers of the elements contain *ti* in the collection.

- *icf(e)* is the inverse of the number of times markup m appears in the collection.
- *N* is the total number of elements in the collection;
  finally,
- *icf(e)* express the popularity of markups m in the collection.

If a user do the following query for a heterogeneous collection in conformity with the **??**:

Will get the answers in this following way: `/article/person/name` whose co-occurrence factor is given by:

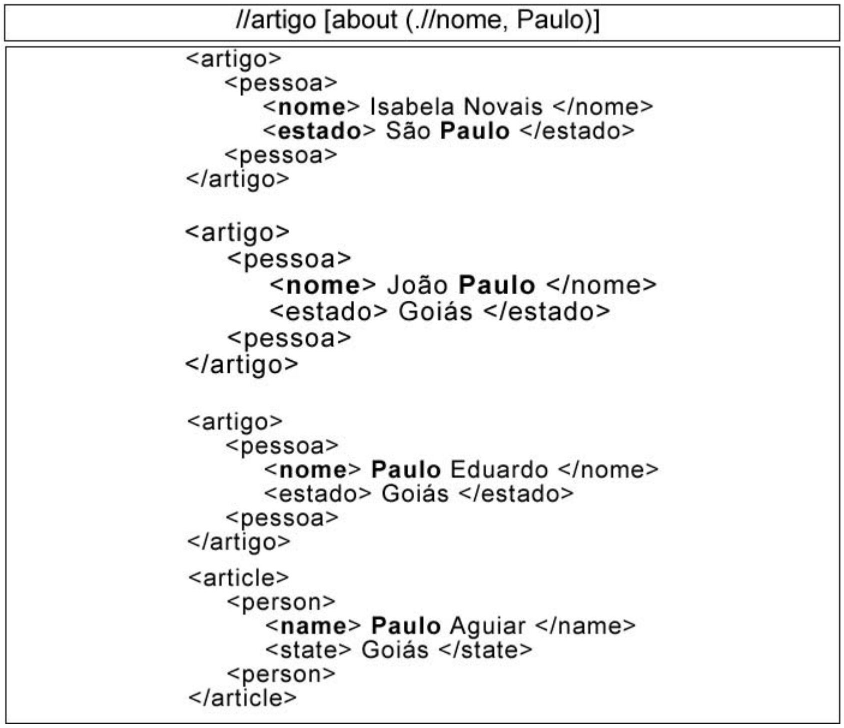
- *cf(Paulo, name) = 1;*
- *idf(Paulo) = 1/4;*
- *icf(name) = 1/1;*
- *N = 14;*
- *focr(Paulo, name) = (1 ∗ 1/4) ∗ (1/1 ∗ 14) = 3.5.*

`/artigo/pessoa/nome` whose the co-occurrence factor is given by:

- *cf(Paulo, nome) = 2;*
- *idf(Paulo) = 1/4;*
- *icf(nome) = 1/3;*
- *N = 14;*
- *focr(Paulo, nome) = (2 ∗ 1/4) ∗ (1/3 ∗ 14) = 2.333.*

`/artigo/pessoa/estado` whose the co-occurrence factor is given by:

- *cf(Paulo, estado) = 1;*



```
//artigo [about (.//nome, Paulo)]
```

```
<artigo>
    <pessoa>
        <nome> Isabela Novais </nome>
        <estado> São Paulo </estado>
    <pessoa>
</artigo>

<artigo>
    <pessoa>
        <nome> João Paulo </nome>
        <estado> Goiás </estado>
    <pessoa>
</artigo>

<artigo>
    <pessoa>
        <nome> Paulo Eduardo </nome>
        <estado> Goiás </estado>
    <pessoa>
</artigo>

<article>
    <person>
        <name> Paulo Aguiar </name>
        <state> Goiás </state>
    <person>
</article>
```

Fig. 3. The Co-occurrence Factor

- $idf(Paulo) = 1/4;$
- $icf(estado) = 1/3;$
- $N = 14;$
- $focr(Paulo,\ estado) = (1 * 1/4) * (1/3 * 14) = 1.166$

The fact of `/article/person/name` and `/artigo/pessoa/nome` be returned to user denotes the capacity of our system works with different structure collections.

This disposition occurs because `<name>` is super valorized, in virtue of it rareness in the collection.

And the element `/artigo/pessoa/estado` be returned although not attend the query but obtain the minor co-occurrence factor.

The co-occurrence factor valorize the co-occurrence terms and markups, considerating the markups popularity. With this factor we pretend explore the characteristic language XML originated from your conception: the presence of markups that describe it contents.

For a better efetivity of our model is important that there's a strait semantical relation between terms and markups that will be more in heterogeneous collections, given the probable diversity of documents structure.

Concluding, the XML Factor *(fxml)* explore the characteristics of XML language, searching for semantic between terms, searching for information behind the words.

4 Results

We submitted runs to the INEX Initiative for heterogeneous collections, but as its assessments were not concluded, we have no Recall/Precision Curves. It follows a sample of an answer to a query showing results from many sub-collections, confirming that our model can deal with different DTDs.
For query:

```
//article[about(.//author,Nivio Ziviani)]
```

we get the following answer:

```
<topictopic-id=‘‘2’’> ...
<result>
    <subcollection name=‘‘ieee’’/>
    <file>co/2000/ry037</file>
    <path>/article[1]/fm[1]/au[1]</path>
    <rank> 3</rank>
</result> ...
<result>
    <subcollection name=‘‘dblp’’ />
    <file>dblp</file>
    <path>/dblp[1]/article[177271]/author[4]</path>
    <rank>6</rank>
</result> ...
<result>
    <subcollection name=‘‘CompuScience’’ />
    <file>exp-dxf1.xml.UTF-8</file>
    <path>/bibliography[1]/article[23]/author[1]</path>
    <rank> 30</rank>
</result>
...
<result>
    <subcollectionname=‘‘hcibib’’ />
    <file>hcibib</file>
    <path>/file[1]/entry[229]/article[1]/author[1]</path>
    <rank>139</rank>
</result>
```

5 Conclusion

To be written after the conclusion of Heterogeneous track.

References

1. S. Abiteboul, P. Buneman and D. Suciu.: Data on the Web - From Relations to Semistructured Data in XML. Mogan Kaufmann Publishers, San Francisco, California, (2000) 27–50.
2. S. Abiteboul, I. Manolescu, B. Nguyen, N. Preda.: A Test Plataform for the INEX Heterogeneous Track. INEX (2004) LNCS
3. M. Azevedo, L.Pantuza e N. Ziviane.: A Universal Model for XML Information Retrieval. INEX (2004) LNCS 3493 311–321 (2005).
4. T. Bray, J. Paoli, C. M. Sperberg-McQueen and E. Maler.: Extensible Markup Language (XML) 1.0. 2nd ed. http://www.w3.org/TR/REC-xml, Cct 2000. W3C Recommendation 6 October (2000).
5. C. Fellbaum.: *WordNet*: An Electronic Lexical Database. MIT Press. (1998).
6. N. Fuhr and M. Lalmas.: INEX document Collection. http://inex.is.informatik.uni-duisburg.de:2004/internal/, Duisburg, (2004).
7. R. R. Larson.: Cheshire II at INEX ’04: Fusion and Feedback for the Adhoc and Heterogeneous Tracks. INEX (2004) LNCS 3493 322–336.
8. M. Lehtonen.: Extirp 2004: Towards Heterogeneity. INEX (2004) LNCS 3493 372–381.
9. B. Ribeiro-Neto e R. Baeza-Yates.: Modern Information Retrieval. Addison Wesley. (1999) pp. 27–30.
10. G. Salton e M. E. Lesk.: Computer evaluation of indexing and text processing. Journal of the ACM. 15(1) (1968) 8–36.
11. K. Sauvagnat, M. Boughanem.: Using a relevance propation method for Adhoc and Heterogeneus tracks in INEX 2004. INEX (2004) LNCS 3493 337–348.
12. A. Trotman and B. Sigurbjornsson.: Narrowed extended xpath i. In In INEX 2004 Workshop Proceedings, page ..., (2004).

The Interactive Track at INEX2005

Birger Larsen[1], Saadia Malik[2] and Anastasios Tombros[3]

[1] Dept. of Information Studies, Royal School of LIS, Copenhagen, Denmark
blar@db.dk

[2] Fak. 5/IIS, Information Systems, University of Duisburg-Essen, Duisburg, Germany
malik@is.informatik.uni-duisburg.de

[3] Dept. of Computer Science, Queen Mary University of London, London, UK
tassos@dcs.qmul.ac.uk

Abstract. In its second year, the Interactive Track at INEX focused on addressing some fundamental issues of interactive XML retrieval. In addition, the track also expanded by including two additional tasks and by attracting more participating groups. A total of 11 research groups and 108 test persons participated in the three different tasks that were included in the track. In this paper, we describe the main issues that the Interactive Track at INEX 2005 tries to address, and the methodology and tasks that were used in the track.

1 Introduction

The overall motivation for the Interactive Track at INEX is twofold. First, to investigate the behaviour of users when interacting with components of XML documents, and secondly to investigate and develop approaches for XML retrieval which are effective in user-based environments.

One of the major outcomes of the Interactive Track in 2004 was the need to investigate methods that can be supportive during the search process based on features extracted from the XML formatting [1, 2]. Problems that might be solved using such methods include overlapping components, and the presentation of retrieved elements in the hit list.

In the baseline system that was offered in 2005 these two issues were addressed. This offered us the opportunity to study how overall user search behaviour was affected by these changes when compared to the behaviour observed in 2004.

In addition, the following aims were addressed in 2005 following the recommendations of the INEX Methodology Workshop at the Glasgow IR Festival[1]:

- To elicit user perceptions of what is needed from an XML retrieval system. The aim is to see whether element retrieval is what users need: Does element retrieval make sense at all to users, do users prefer longer components, shorter components or whole documents, would they rather have passages than elements, etc.

[1] See http://www.cs.otago.ac.nz/inexmw/ for the proceedings and presentation slides.

- To identify an application for element retrieval. This year, a mixture of topics that were simulated work tasks [3] (based on topics from the ad hoc track) and information needs formulated by the test persons themselves. The aim of including the latter was to enable studies of what characterises the tasks users formulate, and to see what kinds of applications users might need an element retrieval system for.
- To introduce an alternative document collection with the Lonely Planet collection as an optional task in order to broaden the scope of INEX and to allow test persons with different backgrounds (e.g. educational) to participate.

The format of the Interactive Track in 2005 was deliberately of an exploratory nature, and has relatively broad aims rather than addressing very specific research questions. Element retrieval is still in its infancy and many basic questions remain unanswered as shown by the discussions at the IR Festival. Aside from the automatic and detailed logging of test persons as used last year, more emphasis was placed on producing qualitative results. Many of the aims stated above were therefore dealt with through careful interviewing and detailed questionnaires. A total of three tasks were available to the track participants: one compulsory task that all participants had to fulfil with a minimum number of test persons, and two optional tasks. These tasks combined several element retrieval systems, topic types and XML collections. By providing a multitude of different perspectives it is our hope that the Interactive Track can aid in illuminating some of the core issues in element retrieval.

The remainder of the paper is organised as follows: The three tasks are described briefly in Section 2, followed by details of the participating groups in Section 3. In depth descriptions of Task A and Task C are given in Sections 4 and 5 respectively, whereas Task B is only described briefly in Section 2. Concluding remarks are given in Section 6.

2 Tasks in the INEX2005 Interactive Track

2.1 Task A - Common baseline system with IEEE collection

In this task each test person searched three topics in the IEEE collection: Two simulated work tasks provided by the organisers, and one formulated by the test person herself in relation to an information need of her own. The baseline system used by all participants was a java-based element retrieval system built within the Daffodil framework[2], and was provided by the track organisers. It has a number of improvements over last year's baseline system, including handling of overlaps, better element summaries in the hit list, a simpler relevance scale, and various supportive interface functionalities. Task A was compulsory for all participating groups with minimum of 6 test persons.

[2] See http://www.is.informatik.uni-duisburg.de/projects/daffodil/index.html.en

2.2 Task B - Participation with own element retrieval system

This task allowed groups who have a working element retrieval system to test their system against a baseline system. Groups participating in Task B were free to choose between the IEEE collection or the Lonely Planet collection, and had a large degree of freedom in setting up the experiment to fit the issues they wanted to investigate in relation to their own system. If the IEEE collection was used Daffodil was offered as baseline system. For the Lonely Planet collection a baseline system was kindly provided by Roelof van Zwol from the Contentlab at Utrecht University[3]. The recommended experimental setup was very close to that of Task A, with the main difference that simulated work tasks should be assigned to test persons rather than freely chosen. This in order to allow for direct comparisons between the baseline system and the local system.

Task B was optional for those groups who had access to their own element retrieval system, and was separate from task A. Thus additional test persons needed to be engaged for task B. See [7] in this volume for an example of an experimental setup used in Task B.

2.3 Task C - Searching the Lonely Planet collection

This task allowed interested groups to carry out experiments with the Lonely Planet collection. Each test person searched four topics which were simulated work tasks provided by the organisers. The system (B^3–SDR) provided by Utrecht University was used in this task. The system is a fully functional element retrieval system that supports several query modes. Task C was optional for those groups who wished to do experiments with the new collection, and was separate from task A and B. Thus additional test persons needed to be engaged for task C. Note that the Lonely Planet collection allows for test persons that do not have a computer science background (In contrast to the IEEE CS collection used in Task A).

Detailed experimental procedures including questionnaires and interview guides for all three tasks were provided to the participants. In addition, a specification of a minimum logging format was provided for local systems in Task B. As for last year, minimum participation in the INEX Interactive Track did not require a large amount of work as the baseline system for Task A was provided by the track. The bulk of the time needed for participating groups was spent on running the experiments; approximately 2 hours per test person.

3 Participating groups

A total of 12 research groups signed up for participation in the Interactive Track and 11 completed the minimum number of required test persons. Their affiliations and distribution on tasks are given in Table 1 below. All 11 groups participated in Task A

[3] See http://contentlab.cs.uu.nl/

with a total of 76 test persons searching on 228 tasks. Only one group, University of Amsterdam, participated in Task B, but with 14 test persons searching on 42 tasks. Four groups participated in Task C with 18 test persons searching 72 tasks. A total of 108 test persons from the 11 active participants took part in the Interactive Track. In comparison, in 2004, 10 groups took part with 88 test persons.

	Task A	**Task B**	**Task C**[4]
Research Group	**Test Persons (Topics)**	**Test Persons (Topics)**	**Test Persons (Topics)**
CWI, University of Twente, The Netherlands	6 (18)	-	4 (16)
Kyungpook National University, Korea	12 (36)	-	-
Oslo University College, Norway	8 (24)	-	-
Queen Mary University of London, England	6 (18)	-	-
RMIT University, Australia	6 (18)	-	4 (16)
Royal School of LIS, Denmark	6 (18)	-	6 (24)
Rutgers University, USA	6 (18)	-	4 (16)
University of Amsterdam, The Netherlands	6 (18)	14 (42)	-
University of Duisburg-Essen, Germany	6 (18)	-	-
University of Tampere, Finland	8 (24)	-	-
Utrecht University, The Netherlands	6 (18)	-	-
Total	76 (228)	14 (42)	18 (72)

Table 1. Research groups participating in the Interactive Track at INEX2005.

4 Task A

4.1 Document corpus

The document corpus used in Task A was the 764 MB corpus of articles from the IEEE Computer Society's journals (version 1.8, merged new & old collection).

4.2 Relevance assessments

The intention was that each viewed element should be assessed with regard to its relevance to the topic by the test person. This was, however, not enforced by the system as we believe that it may be regarded as intrusive by the test persons [4]. In addition, concerns have been raised that last year's composite two dimensional scale was far too complex for the test persons to comprehend [5, 6]. Therefore it was chosen to simplify the relevance scale, also in order to ease the cognitive load on the

[4] At the time of writing Task C has not been completed. The numbers are estimates based on feedback

test persons. The scale used was a simple 3-point scale measuring the usefulness (or pertinence) of the element in relation to the test person's perception of the task:

2 – Relevant
1 – Partially Relevant
0 – Not Relevant

Please note that in contrast to the assessments made for the ad hoc track, there was no requirement on the test persons to view each retrieved element as independent from other viewed components. We have chosen not to enforce any rules in order to allow the test persons to behave as close as possible to what they would normally do.

For Task C we experimented with a slightly more complex relevance scale (see Section 6.2 below).

4.3 System

The baseline system used in Task A is a java-based element retrieval system built within the Daffodil framework. The HyREX retrieval engine[5] was used as backend in the baseline system.

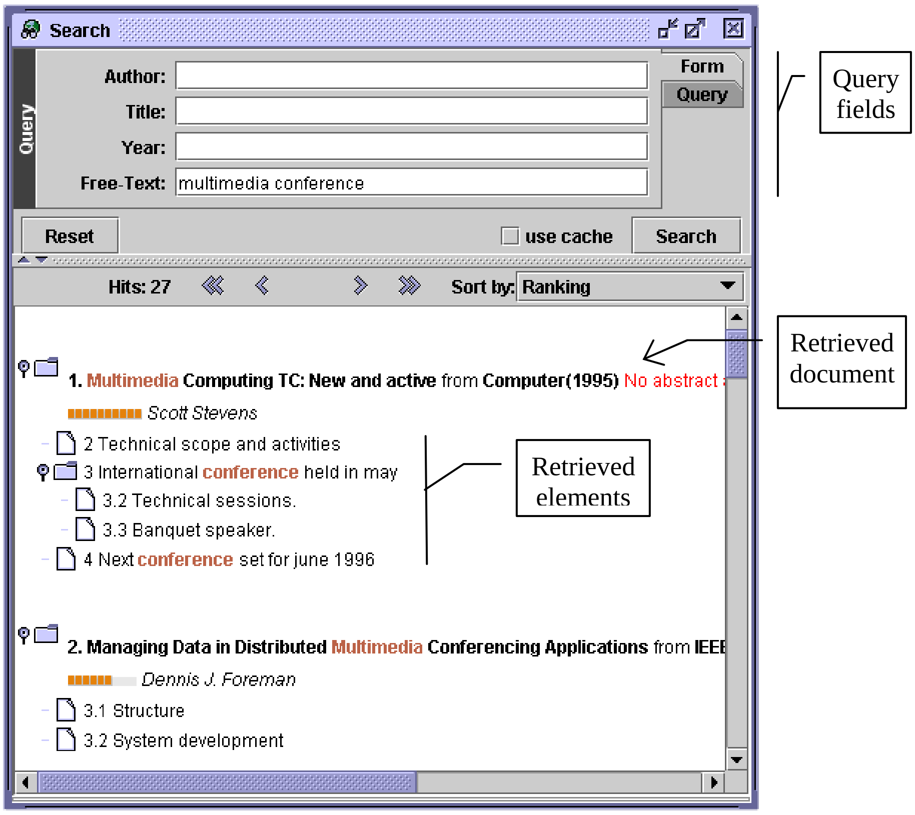
Fig. 1 below shows the query and results list interface of the baseline system. After entering a query and pressing "Search" a search progress indicator informs the test person about the number documents found. A related term list also appeared, suggesting alternative search terms (not shown). The results are presented as documents and in some cases the system indicates which elements that may be most closely related to the query.

Double-clicking a document or an element opens this in a new window as shown in Fig. 2 below. This is split in two panes: one with a Table of Contents of the whole document, and one with the full text of the selected element. The selected element is displayed on the right. On the left, the Table of Contents indicates the currently viewed element, other retrieved elements, viewed and assessed elements. The relevance scale is implemented as simple icons to be clicked:

▣ - Relevant
◪ - Partially Relevant
□ - Not Relevant

The logging in the baseline system was saved to a database for greater flexibility and stability. The log data comprises of one session for each topic the test person searches. The log for each session records the events in the session, both the actions performed by the test person and the responses from the system.

[5] http://www.is.informatik.uni-duisburg.de/projects/hyrex/



Author:

Title:

Year:

Free-Text: multimedia conference

□ use cache

Hits: 27

Sort by: Ranking

1. Multimedia Computing TC: New and active from Computer(1995) No abstract

Scott Stevens

- 2 Technical scope and activities
- 3 International conference held in may
  - 3.2 Technical sessions.
  - 3.3 Banquet speaker.
- 4 Next conference set for june 1996

2. Managing Data in Distributed Multimedia Conferencing Applications from IEEE

Dennis J. Foreman

- 3.1 Structure
- 3.2 System development

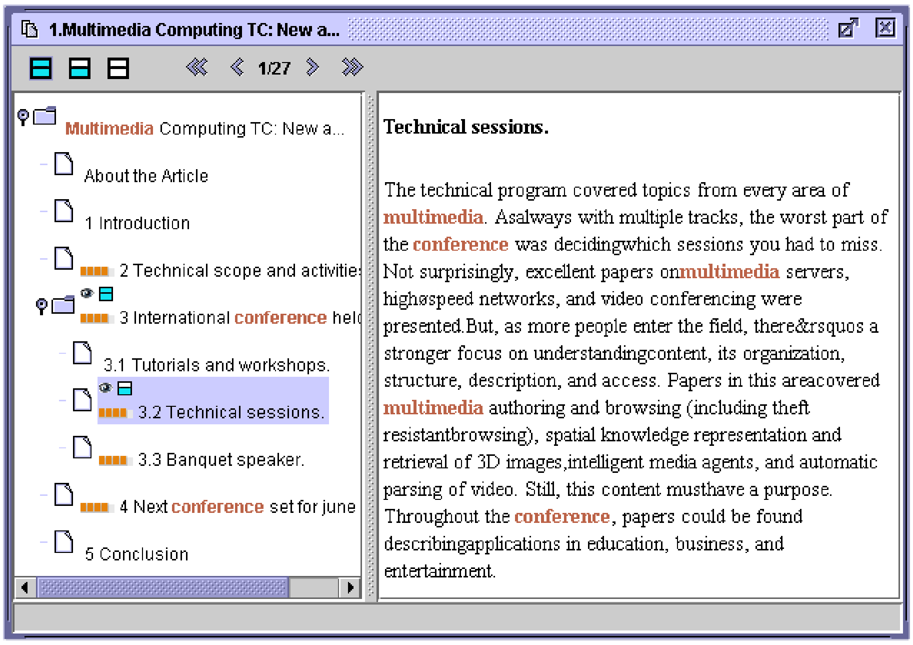
Fig. 1. Query box and result list display in the baseline system used in Task A.

4.4 Tasks/Topics

In order to study the questions outlined in Section 1 above related to the needs for element retrieval systems and possible applications of such systems, both real and simulated information needs were used in Task A.

The test persons were asked to supply examples of own information needs. As it may be hard for the test persons to formulate topics that are covered by the collection, the test persons emailed two topics they would like to search for 48 hours before the experiment. The experimenters then did a preliminary search of the collection to determine which topic had the best coverage in the collection. The topics supplied by the test persons were not all well-suited to an element retrieval system, but they all had a valuable function as triggers for the structured interview where it was attempted to elicit user perceptions of what they need from an element retrieval system, and to identify possible applications for element retrieval. They may also be valuable for the formulation of topics for next year’s track. Therefore, both topics were recorded and submitted as part of the results.

The simulated work tasks were derived from the CO+S and CAS INEX 2005 ad-hoc topics, ignoring any structural constraints. In order to make the topics comprehensible by other than the topic author, it was required that the ad hoc topics not only detail *what* is being sought for, but also *why* this is wanted, and in what *context* the information need has arisen. This information was exploited for creating simulated work task situations for Task A that, on the one hand will allow the test persons to engage in realistic searching behaviour, and on the other provide a certain level of experimental control by being common across test persons[6].


1.Multimedia Computing TC: New a...
1/27
Multimedia Computing TC: New a...
About the Article
1 Introduction
2 Technical scope and activitie
3 International conference held
3.1 Tutorials and workshops.
3.2 Technical sessions.
3.3 Banquet speaker.
4 Next conference set for june
5 Conclusion

Technical sessions.

The technical program covered topics from every area of multimedia. Asalways with multiple tracks, the worst part of the conference was decidingwhich sessions you had to miss. Not surprisingly, excellent papers onmultimedia servers, highøspeed networks, and video conferencing were presented.But, as more people enter the field, there&rsquos a stronger focus on understandingcontent, its organization, structure, description, and access. Papers in this areacovered multimedia authoring and browsing (including theft resistantbrowsing), spatial knowledge representation and retrieval of 3D images,intelligent media agents, and automatic parsing of video. Still, this content musthave a purpose. Throughout the conference, papers could be found describingapplications in education, business, and entertainment.

Fig. 2. Full text result in the baseline system used in Task A.

For Task A, six topics were selected and modified into simulated work tasks. In last year's track we attempted to identify tasks of different types and to study the difference between them, but with out great success. This year a simple partition has been made into two categories:

- General tasks (G category), and
- Challenging tasks (C category), which are more complex and may be less easy to complete.

In addition to their own information need, each test person chose one task from each category. This allows the topic to be more "relevant" and interesting to the test person. A maximum time limit of 20 minutes applied for each task. Sessions could finish before this if the test person felt they have completed the task.

[6] See the work of Borlund for more information on simulated work tasks, e.g. Borlund, 2003 (http://informationr.net/ir/8-3/paper152.html)

4.5 Experimental design

4.5.1 Experimental matrix

A minimum of 6 test persons from each participating site were used. Each test person searched on one simulated work task from each category (chosen by the test person) as well as one of their own topics. The order in which task categories were performed by searchers was permuted in order to neutralise learning effects. This means that one complete round of the experiment requires 6 searchers.

The basic experimental matrix looked as follows:

Rotation 1:	OT, STG, STC
Rotation 2:	STC, OT, STG
Rotation 3:	STG, STC, OT
Rotation 4:	STG, OT, STC
Rotation 5:	STC, STG, OT
Rotation 6:	OT, STC, STG

Where OT = Own task, and STG, STC are the two 2 simulated work task categories. As can be seen from Table 1 above some groups did more than 6 test persons. It was attempted to coordinate the permutation rotations across these groups to arrive at an equal distribution of across the track.

4.5.2 Experimental procedure

The experimental procedure for each test person is outlined below.

1. Experimenter briefed the searcher, and explained the format of the study
2. Tutorial of the system was given with a training task, and experimenter answered any questions
3. 'Instructions to searchers' handed out
4. Any questions answered by the experimenter
5. Entry questionnaire handed out
6. *Task description for the first category handed out, and a task selected*
7. *Pre-task questionnaire handed out*
8. *Task began, and experimenter logged in. Max. duration 20 minutes. Experimenter logged out.*
9. *Post-task questionnaire handed out*
10. Steps 7-10 were repeated for the two other tasks
11. Post-experiment questionnaire handed out
12. Interview

The system training, the three tasks and completion of questionnaires and interview were performed in one, continuous session. An Instructions to searchers document gave information to the searchers about the experiment and their role in it, including basic information about system information, an outline of the experimental procedure, and how to assess elements for relevance. A number of questionnaires and guidelines for post-experiment interviews were provided by the track organisers. The purpose of the semi-structured interview was to attempt to elicit user perceptions of what they need from an element retrieval system, and to identify possible applications for element retrieval.

5 Task C

Task C was optional for those groups who wished to experiment with the Lonely Planet collection, and was separate from Task A and B. Thus additional test persons needed to be engaged for Task C. Task C is meant as an exploratory task to initiate interactive experiments with the LP collection.

5.1 Document corpus

The document corpus used in Task C was the Lonely Planet collection deployed by Roelof van Zwol. The Lonely Planet collection consists of 462 XML documents with information about destinations, which is particularly useful for travellers that want to find interesting details for their next holiday or business trip. The collection is called the "WorldGuide" and has been provided by the publishers of the Lonely Planet guidebooks. The collection not only contains useful information about countries, but also includes information about interesting regions and major cities. For each destination an introduction is available, complemented with information about transport, culture, major events, facts, and an image gallery that gives an impression of the local scenery.

5.2 Relevance assessments in Task C

A slightly more complex approach was taken for the collection of relevance assessments in Tack C. The two-dimensional relevance scale was a modified version of a scale proposed at the INEX Methodology Workshop at the Glasgow IR Festival [6]. The relevance assessments were explained to the test persons as follows:

Two different dimensions are used to assess the relevance of an XML document component. The first determines the extent to which a document component *contains relevant information* for the search task. It can take one of the following three values: ***highly relevant***, ***somewhat relevant***, and ***not relevant***. A document component is ***highly relevant*** if it covers aspects of the search task *without containing* too much non-relevant information. A document component is ***somewhat relevant*** if it covers aspects of the search task and at the same time *contains* much non-relevant information. A document component is ***not relevant*** if it *does not cover* any aspect of the search task.

The second relevance dimension determines the extent to which a document component *needs the context* of its containing XML document to make full sense as an answer. It can take one of the following three values: ***just right***, ***too large***, and ***too small***. A document component is ***just right*** if it is reasonably *self-contained* and it needs *little* of the context of its containing XML document to make full sense as an answer. Alternatively, the document component can be either ***too large*** or ***too small***. A document component is ***too large*** if it *does not need* the context of its containing XML document to make full sense as an answer. A document component is ***too small*** if it can only make full sense within the context of its containing XML document.

Given the above relevance values, the final assessment score of a document component can take one of the following ***five*** values:

- **Not Relevant (NR)** – if the document component does not cover any aspect of the search task;
- **Partial Answer (PA)** – if the document component is *somewhat relevant* (i.e. covers only some aspects of the search task) and *just right* (i.e. it is reasonably self-contained but still needs some of the context of its containing XML document to make full sense);
- **Exact Answer (EA)** – if the document component is *highly relevant* (i.e. covers all, or nearly all, aspects of the search task without containing too much non-relevant information) and *just right;*
- **Broad Answer (BA)** – if the document component is either *highly* or *somewhat relevant* and *too large* (i.e. it is reasonably self-contained and does not really need the context of its containing XML document to make full sense); and
- **Narrow Answer (NA)** - if the document component is either *highly* or *somewhat relevant* and *too small* (i.e. it is not self-contained and can only make full sense in the context of its containing XML document).

The test persons could select one of these values from a T-shaped relevance assessment box as shown in Fig. 4 and Fig. 5.

5.3 System

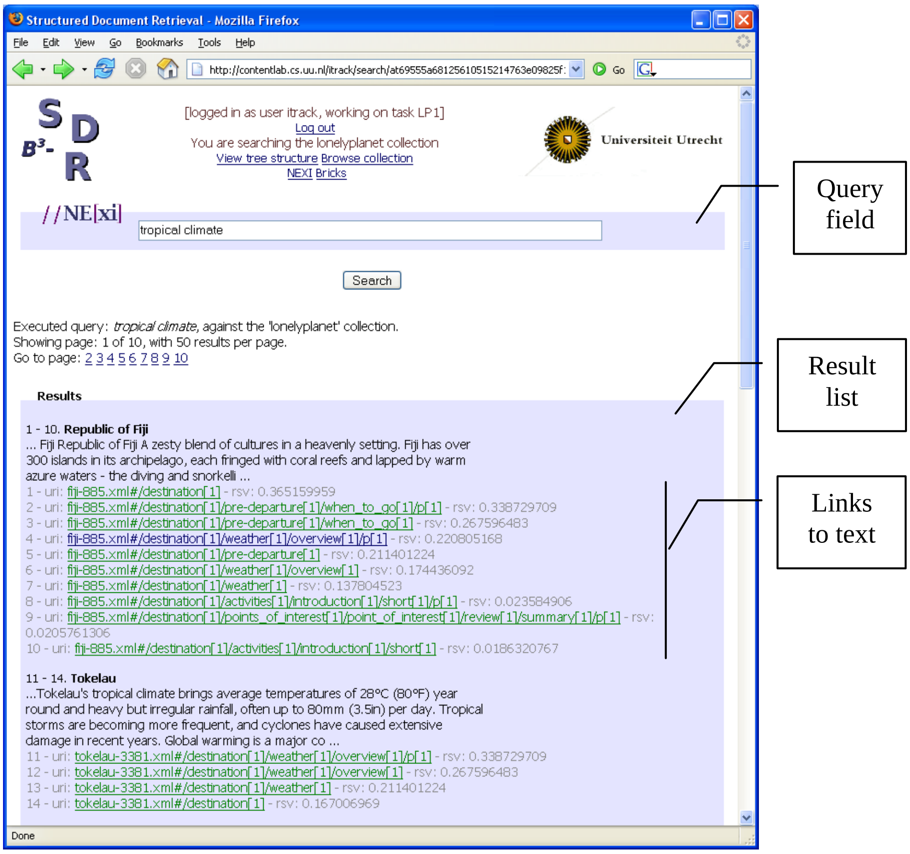
An interactive system for Task C was provided by Utrecht University. It is a fully functional element retrieval system which has been configured to suit Task C. There were two versions of the system: One which presented the results in context of the full text (i.e., highlighted), and an alternative version which presented the results in isolation. Fig. 3 shows the query ad result list interface common to both system versions. Fig. 4 and 5 shows the interface for the versions which showed results in context and isolated respectively.

5.4 Tasks/Topics

Eight topics that have previously been used for experiments with the Lonely Planet Worldguide were selected and modified into short simulated work tasks for Task C. The tasks were arbitrarily split into 2 categories, and each test person searched two tasks from each category.

5.5 Experimental design

A minimum of 4 test persons from each participating group were used in Task C. Each test person searched two simulated work tasks (chosen by the test person) from each of the two categories – a total of four per test person. The order in which task categories were performed was permuted in order to neutralise learning effects. This means that one complete round of the experiment required 4 searchers.


Structured Document Retrieval - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Go
SDR
B³-
Universiteit Utrecht
//NE[xi]
Search
Done
Query field
Result list
Links to text

http://contentlab.cs.uu.nl/itrack/search/at69555a68125610515214763e09825f:

[logged in as user itrack, working on task LP1]
Log out
You are searching the lonelyplanet collection
View tree structure Browse collection
NEXI Bricks

tropical climate

Executed query: *tropical climate*, against the 'lonelyplanet' collection.
Showing page: 1 of 10, with 50 results per page.
Go to page: 2 3 4 5 6 7 8 9 10

Results

1 - 10. **Republic of Fiji**
... Fiji Republic of Fiji A zesty blend of cultures in a heavenly setting. Fiji has over 300 islands in its archipelago, each fringed with coral reefs and lapped by warm azure waters - the diving and snorkelli ...
1 - uri: fiji-885.xml#/destination[1] - rsv: 0.365159959
2 - uri: fiji-885.xml#/destination[1]/pre-departure[1]/when_to_go[1]/p[1] - rsv: 0.338729709
3 - uri: fiji-885.xml#/destination[1]/pre-departure[1]/when_to_go[1] - rsv: 0.267596483
4 - uri: fiji-885.xml#/destination[1]/weather[1]/overview[1]/p[1] - rsv: 0.220805168
5 - uri: fiji-885.xml#/destination[1]/pre-departure[1] - rsv: 0.211401224
6 - uri: fiji-885.xml#/destination[1]/weather[1]/overview[1] - rsv: 0.174436092
7 - uri: fiji-885.xml#/destination[1]/weather[1] - rsv: 0.137804523
8 - uri: fiji-885.xml#/destination[1]/activities[1]/introduction[1]/short[1]/p[1] - rsv: 0.023584906
9 - uri: fiji-885.xml#/destination[1]/points_of_interest[1]/point_of_interest[1]/review[1]/summary[1]/p[1] - rsv: 0.0205761306
10 - uri: fiji-885.xml#/destination[1]/activities[1]/introduction[1]/short[1] - rsv: 0.0186320767

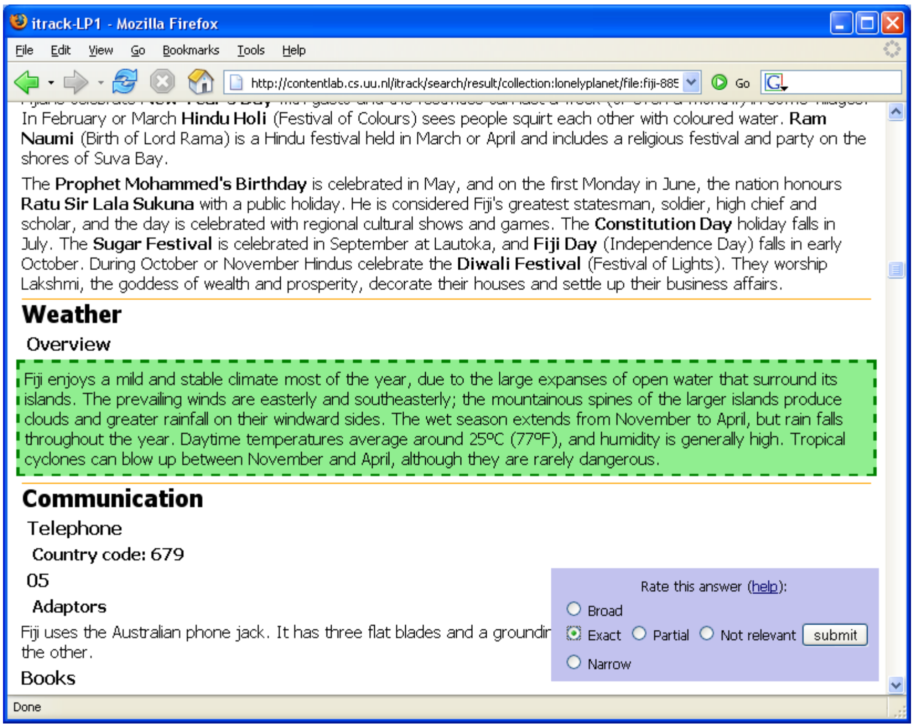
11 - 14. **Tokelau**
...Tokelau's tropical climate brings average temperatures of 28°C (80°F) year round and heavy but irregular rainfall, often up to 80mm (3.5in) per day. Tropical storms are becoming more frequent, and cyclones have caused extensive damage in recent years. Global warming is a major co ...
11 - uri: tokelau-3381.xml#/destination[1]/weather[1]/overview[1]/p[1] - rsv: 0.338729709
12 - uri: tokelau-3381.xml#/destination[1]/weather[1]/overview[1] - rsv: 0.267596483
13 - uri: tokelau-3381.xml#/destination[1]/weather[1] - rsv: 0.211401224
14 - uri: tokelau-3381.xml#/destination[1] - rsv: 0.167006969

Fig. 3. Query box and result list display used in Task C.

The basic experimental matrix looks as follows:

Rotation 1:	Iso-C1, Cxt-C2
Rotation 2:	Iso-C2, Cxt-C1
Rotation 3:	Cxt-C1, Iso-C2
Rotation 4:	Cxt-C2, Iso-C1

Where Iso = system with isolated results, and Cxt = system with results in context. C1 and C2 were the two simulated work task categories. The experimental procedure was very similar to the ones used in Task A. However, no interview was conducted at the end of the experiment. A number of questionnaires were provided by the track organisers.


itrack-LP1 - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://contentlab.cs.uu.nl/itrack/search/result/collection:lonelyplanet/file:fiji-885
Go
Done

In February or March **Hindu Holi** (Festival of Colours) sees people squirt each other with coloured water. **Ram Naumi** (Birth of Lord Rama) is a Hindu festival held in March or April and includes a religious festival and party on the shores of Suva Bay.

The **Prophet Mohammed's Birthday** is celebrated in May, and on the first Monday in June, the nation honours **Ratu Sir Lala Sukuna** with a public holiday. He is considered Fiji's greatest statesman, soldier, high chief and scholar, and the day is celebrated with regional cultural shows and games. The **Constitution Day** holiday falls in July. The **Sugar Festival** is celebrated in September at Lautoka, and **Fiji Day** (Independence Day) falls in early October. During October or November Hindus celebrate the **Diwali Festival** (Festival of Lights). They worship Lakshmi, the goddess of wealth and prosperity, decorate their houses and settle up their business affairs.

Weather

Overview

Fiji enjoys a mild and stable climate most of the year, due to the large expanses of open water that surround its islands. The prevailing winds are easterly and southeasterly; the mountainous spines of the larger islands produce clouds and greater rainfall on their windward sides. The wet season extends from November to April, but rain falls throughout the year. Daytime temperatures average around 25ºC (77ºF), and humidity is generally high. Tropical cyclones can blow up between November and April, although they are rarely dangerous.

Communication

Telephone

Country code: 679

05

Adaptors

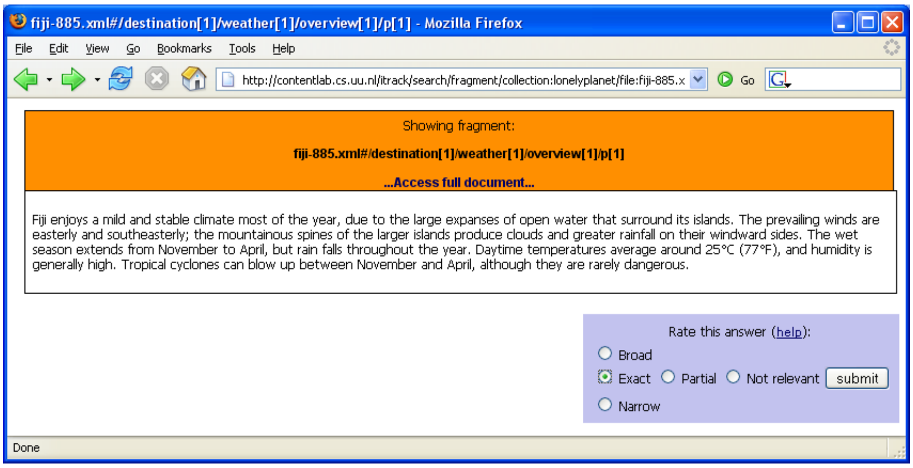
Fiji uses the Australian phone jack. It has three flat blades and a groundir
the other.

Books

Rate this answer (help):

○ Broad
● Exact ○ Partial ○ Not relevant submit
○ Narrow

Fig. 4. Task C system version which presented the results highlighted in context of the full text.


fiji-885.xml#/destination[1]/weather[1]/overview[1]/p[1] - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://contentlab.cs.uu.nl/itrack/search/fragment/collection:lonelyplanet/file:fiji-885.x
Go
Done

Showing fragment:

fiji-885.xml#/destination[1]/weather[1]/overview[1]/p[1]

...Access full document...

Fiji enjoys a mild and stable climate most of the year, due to the large expanses of open water that surround its islands. The prevailing winds are easterly and southeasterly; the mountainous spines of the larger islands produce clouds and greater rainfall on their windward sides. The wet season extends from November to April, but rain falls throughout the year. Daytime temperatures average around 25°C (77°F), and humidity is generally high. Tropical cyclones can blow up between November and April, although they are rarely dangerous.

Rate this answer (help):

○ Broad
● Exact ○ Partial ○ Not relevant submit
○ Narrow

Fig. 5. Task C system version which presented the results in isolation.

6 Concluding remarks

In its second year, the Interactive Track at INEX focused on addressing some fundamental issues of interactive XML retrieval. In addition, the track also expanded by including two additional tasks and by attracting more participating groups. A total of 11 research groups and 108 test persons participated in the three different tasks that were included in the track. In this paper, we have described the main issues that the Interactive Track at INEX 2005 tries to address, and the methodology and tasks that were used in the track. At the time of writing the results are still under analysis. It is hoped that we can present an initial analysis at the INEX2005 workshop at Schloss Dagstuhl.

7 Acknowledgments

As organisers we would like to thank the participating research groups for their work and input concerning the format of the track. In addition, we would like to thank Pia Borlund from the Royal School of Library and Information Science for help on constructing simulated work tasks, Claus-Peter Klas from University of Duisburg-Essen for help in running Daffodil, Roelof van Zwol and Sandor Spruit, Utrecht University, for making their system available for Task C, and finally Jovan Pehcevski and co-workers for developing their relevance scale for use in the interactive track.

8 References

1. Tombros, A., Larsen, B. and Malik, S. (2005): The Interactive Track at INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S. and Szlávik, Z. eds. *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers*. Berlin: Springer, p. 410-423. (Lecture Notes in Computer Science ; 3493)
2. Larsen, B., Tombros, A., and Malik, S. (2004): Interactive Track Workshop Report. Slides presented at the INEX Workshop, December 2004
[http://inex.is.informatik.uni-duisburg.de:2004/workshop.html#Reports]
3. Borlund, P. (2003): The IIR evaluation model: a framework for evaluation of interactive information retrieval. In: *Information Research, vol. 8, no. 3, paper no. 152.* [Available at: http://informationr,net/ir/8-3/paper152.html]
4. Larsen, B., Tombros, A. and Malik, S. (2005): Obtrusiveness and relevance assessment in interactive XML IR experiments. In: Trotman, A., Lalmas, M. and Fuhr, N. eds. *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, held at the University of Glasgow, 30 July 2005. Second Edition.* Dunedin (New Zealand): Department of Computer Science, University of Otago, p. 39-42.
[http://www.cs.otago.ac.nz/inexmw/Proceedings.pdf, visited 15-12-2005]
5. Pharo, N. and Nordlie, R. (2005): Context matters : an analysis of assessments of XML documents. In: Crestani, F. and Ruthven, I. eds. *Context: Nature, Impact, and Role : 5th International Conference on Conceptions of Library and Information Science, CoLIS 2005, Glasgow, UK, June 2005, Proceedings.* Berlin: Springer, p. 238-248. (Lecture Notes in Computer Science ; 3507)

6. Pehcevski, J., Thom, J. A. and Vercoustre, A.-M. (2005): Users and assessors in the context of INEX: Are relevance dimensions relevant? In: Trotman, A., Lalmas, M. and Fuhr, N. eds. Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, held at the University of Glasgow, 30 July 2005. Second Edition. Dunedin (New Zealand): Department of Computer Science, University of Otago, p. 47-62. [http://www.cs.otago.ac.nz/inexmw/Proceedings.pdf, visited 15-12-2005]
7. Kamps, J., de Rijke, M. And Sigurbjörnsson, B. (2005): University of Amsterdam at INEX 2005: Interactive Track. In: This volume.

University of Amsterdam at INEX 2005: Interactive Track

Jaap Kamps[1,2], Maarten de Rijke[2], and Börkur Sigurbjörnsson[2]

[1] Archives and Information Science, Faculty of Humanities, University of Amsterdam
[2] Informatics Institute, Faculty of Science, University of Amsterdam

Abstract. This is a preliminary report on the University of Amsterdam's participation in the INEX 2005 Interactive Track. We participated in Task A, a common baseline system with the IEEE collection, as well as in Task B, in which the baseline system is compared to a home-grown XML element retrieval system, xmlfind.

1 Introduction

This paper documents the University of Amsterdam's participation in the INEX 2005 Interactive Track. We conducted two experiments. First, we took part in the concerted effort of Task A, in which a common baseline system, Daffodil/-HyREX, is used to study test-persons searching the IEEE collection, Second, as part of the Interactive Track's Task B, we conducted a comparative experiment, in which the baseline retrieval system, Daffodil/HyREX, is contrasted with our home-grown XML element retrieval system, xmlfind.

The rest of the paper is organized as follows. Next, Section 2 documents the XML retrieval systems used in the experiment. Then, in Section 3, we detail the set-up of the experiments. The preliminary results of the experiments are discussed in Section 4. Finally, in Section 5, we draw some initial conclusions.

2 XML Retrieval Systems

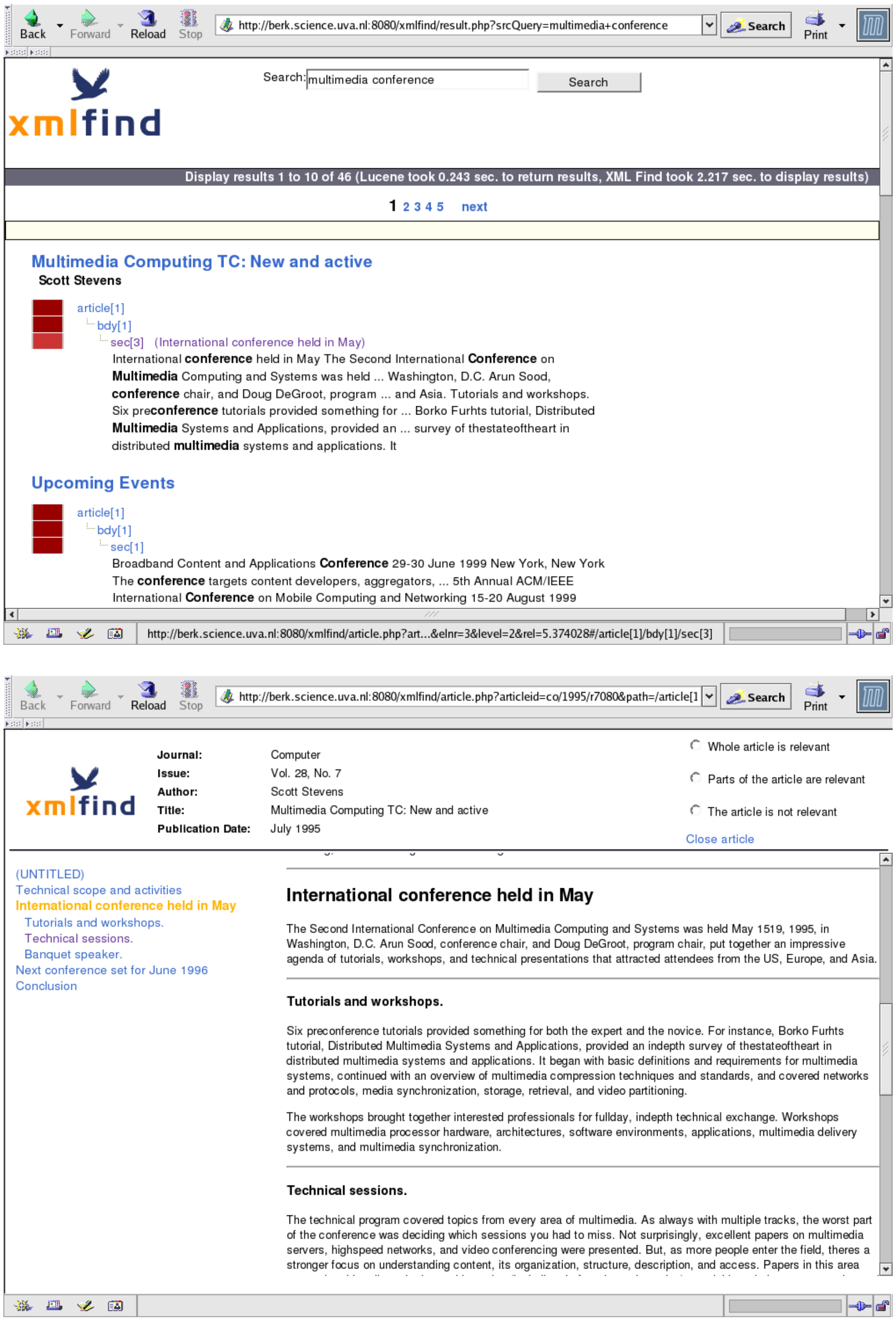
2.1 Baseline System: Daffodil

The Daffodil system supports the information seeking process in Digital Libraries [2]. As a back-end, the HyREX XML retrieval system was used [3]. For details, see [4].

2.2 Home-grown System: xmlfind

The xmlfind system provides an interface for a XML information retrieval search engine [1]. It runs on top of a Lucene search engine [5]. The underlying index contains individual XML element in the IEEE collection [6].

Figure 1(top) shows the search box and the result list. The results are grouped per article, where (potentially) relevant elements are shown. Clicking on any of



http://berk.science.uva.nl:8080/xmlfind/result.php?srcQuery=multimedia+conference

Search: multimedia conference

Display results 1 to 10 of 46 (Lucene took 0.243 sec. to return results, XML Find took 2.217 sec. to display results)

1 2 3 4 5 next

Multimedia Computing TC: New and active

Scott Stevens

article[1]
bdy[1]
sec[3] (International conference held in May)

International **conference** held in May The Second International **Conference** on **Multimedia** Computing and Systems was held ... Washington, D.C. Arun Sood, **conference** chair, and Doug DeGroot, program ... and Asia. Tutorials and workshops. Six pre**conference** tutorials provided something for ... Borko Furhts tutorial, Distributed **Multimedia** Systems and Applications, provided an ... survey of thestateoftheart in distributed **multimedia** systems and applications. It

Upcoming Events

article[1]
bdy[1]
sec[1]

Broadband Content and Applications **Conference** 29-30 June 1999 New York, New York The **conference** targets content developers, aggregators, ... 5th Annual ACM/IEEE International **Conference** on Mobile Computing and Networking 15-20 August 1999

http://berk.science.uva.nl:8080/xmlfind/article.php?art...&elnr=3&level=2&rel=5.374028#/article[1]/bdy[1]/sec[3]

http://berk.science.uva.nl:8080/xmlfind/article.php?articleid=co/1995/r7080&path=/article[1

Journal:	Computer
Issue:	Vol. 28, No. 7
Author:	Scott Stevens
Title:	Multimedia Computing TC: New and active
Publication Date:	July 1995

○ Whole article is relevant
○ Parts of the article are relevant
○ The article is not relevant

Close article

(UNTITLED)
Technical scope and activities
International conference held in May
Tutorials and workshops.
Technical sessions.
Banquet speaker.
Next conference set for June 1996
Conclusion

International conference held in May

The Second International Conference on Multimedia Computing and Systems was held May 1519, 1995, in Washington, D.C. Arun Sood, conference chair, and Doug DeGroot, program chair, put together an impressive agenda of tutorials, workshops, and technical presentations that attracted attendees from the US, Europe, and Asia.

Tutorials and workshops.

Six preconference tutorials provided something for both the expert and the novice. For instance, Borko Furhts tutorial, Distributed Multimedia Systems and Applications, provided an indepth survey of thestateoftheart in distributed multimedia systems and applications. It began with basic definitions and requirements for multimedia systems, continued with an overview of multimedia compression techniques and standards, and covered networks and protocols, media synchronization, storage, retrieval, and video partitioning.

The workshops brought together interested professionals for fullday, indepth technical exchange. Workshops covered multimedia processor hardware, architectures, software environments, applications, multimedia delivery systems, and multimedia synchronization.

Technical sessions.

The technical program covered topics from every area of multimedia. As always with multiple tracks, the worst part of the conference was deciding which sessions you had to miss. Not surprisingly, excellent papers on multimedia servers, highspeed networks, and video conferencing were presented. But, as more people enter the field, theres a stronger focus on understanding content, its organization, structure, description, and access. Papers in this area

Fig. 1. Screen shots of xmlfind: (top) result list, (bottom) detailed view.

Table 1. Experimental matrix for the comparative experiment.

#	Rotation	Task 1		Task 2		Task 3	
		Task	System	Task	System	Task	System
1	1	G-1	Daffodil	C-1	xmlfind	Own	choice
2	2	C-1	Daffodil	G-1	xmlfind	Own	choice
3	3	G-1	xmlfind	C-1	Daffodil	Own	choice
4	4	C-1	xmlfind	G-1	Daffodil	Own	choice
5	1	G-2	Daffodil	C-2	xmlfind	Own	choice
6	2	C-2	Daffodil	G-2	xmlfind	Own	choice
7	3	G-2	xmlfind	C-2	Daffodil	Own	choice
8	4	C-2	xmlfind	G-2	Daffodil	Own	choice
9	1	G-3	Daffodil	C-3	xmlfind	Own	choice
10	2	C-3	Daffodil	G-3	xmlfind	Own	choice
11	3	G-3	xmlfind	C-3	Daffodil	Own	choice
12	4	C-3	xmlfind	G-3	Daffodil	Own	choice
13	1	G-1	Daffodil	C-1	xmlfind	Own	choice
14	2	C-1	Daffodil	G-1	xmlfind	Own	choice

the elements will open a new window displaying the result. Figure 1(bottom) shows the full article with the focus on the selected element. The results display window has three planes. On the left plane, there is a Table of Contents of the whole article. On the right plane, the article is displayed with the selected part of the document in view. On the top plane, the article's title, author, etc. are displayed, as well as a menu for assessing the relevance of the result.

3 Experimental Setup

The whole experiment was run in a single session where test persons for both Task A and Task B worked in parallel. The test persons were first year Computer Science students.

3.1 Task A: Community Experiment

Task A is the orchestrated experiment in which all teams participating in the Interactive Track take part [4]. We participated in Task A with six test persons, who searched the IEEE Collection with the Daffodil/HyREX baseline system. There were three tasks: two simulated work tasks (a 'general' task and a 'challenging' task) and the test person's were asked to think up a search topic of their own. The experiment was conducted in close accordance with the guidelines, for further details we refer to [4].

3.2 Task B: Comparative Experiment

Task B is a comparison of the home-grown xmlfind system with the Daffodil/-HyREX baseline system. We participated in Task B with fourteen test persons.

Table 2. Topic created by test person.

A. *What are you looking for?* Who build the first computer and what did it look like?
B. *What is the motivation of the topic?* I would like to know how the history of the computer began and what the first computer looked like, was it very big or very small, did it have a monitor?
C. *What would an ideal answer look like?* The name of the inventor and a picture of how the first computer looked.

The experimental setup is largely resembling the setup of Task A. Again, test persons do two simulated work tasks (a 'general' and a 'challenging' task) as well as search for a topic they were asked to think up themselves. The experimental matrix is shown in Table 1. Every test person searches for two simulated tasks, each one with a different system. Next, the test persons search for their own topic with a system of their choice.

Due to the number of test persons involved, we were unable to conduct individual exit interviews. Instead, we used an extended post-experiment questionnaire.

4 Preliminary Results

We have only started to process the massive amount of data collected during the experiment. Each test person searched with four different accounts, one for each task, plus one or two additional accounts for training. This generated in total 94 search logs. Additional to this, each person filled in questionnaires before and after each task, and before and after the experiment, resulting in, in total, 160 questionnaires. Below, we will just give some preliminary results.

Own topics As part of the experiments, test persons were asked to think up a search topic of their own interest, based on a short description of the IEEE collection's content. Some topics created by test persons were excellent. Table 2 shows an example of a topic being (i) within the collection's coverage, (ii) reflecting a focused information need, and (iii) even containing potential structural retrieval cues. However, most topic were not so perfect. Even though test persons were asked to think up two different topics, almost half of the test persons did not create a very suitable topic. Frequently, topics addressed very practical advice on computer components or software, such as addressed in FAQs, and some were simply off-topic. Perhaps more positively, the own topics were for the vast majority focused, asking for very specific information that could, in principle, be contained in a relatively short piece of text.

Table 3. Responses by test persons.

13. *Did you like the idea that the search engine takes into account the structure of the documents? Why?*	14. *Do you find it useful to be pointed to relevant parts of long articles? Why?*
Yes, you will have a good overview of the total article/document.	Yes, because you are able to see which articles are worth reading and which are not.
Yes, for specific information this is very useful.	Yes, gives the user an idea about the article in question.
Yes, easier to see how long the article is.	You don't need to see other parts.
Yes, its easier to see the contents of the document, better navigation.	Yes, you don't have to dig into the article yourself.
Yes, it didn't bother me.	Yes, it's more easy to find what you're looking for.
Yes, less reading time, clear overview.	Yes, saves time.
Yes, it shortens search time.	Yes, because if scan-read long articles, you easily miss some relevant parts.
Yes, saves work.	Yes, works faster.
Yes, because its much faster.	Yes, its faster.
Yes, this way of finding information takes less time.	Yes, now you don't have to read the whole article. You can get straight to the part where the information is.
Yes, its easier to see where relevant information is located.	Yes, it takes less time to find the relevant parts.
Yes, it makes it easier to find specific paragraphs.	Yes, if programmed right it can save time.
Yes, it makes it a lot easier to find what you are looking for.	Yes, it is lots more easier.
Yes, because makes me have to search less.	Yes, to search less.

System of Choice Test persons in Task B were free to select with which of the two system they searched for the third topic. Out of the 14 test persons, 4 (28.6%) choose to search with the Daffodil/HyREX system, the other 10 (71.4%) choose to search with the xmlfind system.

General Views Test persons in Task B were, as part of the extended post-experiment questionnaire, asked a number of questions about their opinions on the concept of an XML retrieval engine. Table 3 lists the responses to two of the questions, where each row represents the same test person. The responses where equivocally positive, and the responses highlight many of the hoped advantages of an XML retrieval system.

5 Discussion and Conclusions

This paper documents the University of Amsterdam's participation in the INEX 2005 Interactive Track. We participated in two tasks. First, we participated in the concerted effort of Task A, in which a common baseline system, Daffodil/-HyREX, was used by six test-persons to search the IEEE collection, Second, we conducted a comparative experiment in Task B, in which fourteen test persons searched alternately with the baseline retrieval system, Daffodil/HyREX, and our home-grown XML element retrieval system, xmlfind.

We detailed the experimental set-up of the comparative experiment. Both experiments, involving in total twenty test persons, were conducted in parallel in a single session. This ensured that the experimental conditions for all test persons are very equal. Unplanned external causes, such as the down-time of the Daffodil/HyREX system were affecting all test persons equally. Due to the large number of test persons present at the same time, we had to minimize the need for experimenter assistance. This was accomplished by generating personalized protocols for all test persons. In these protocols, test persons were guided through the experiment by means of verbose instructions on the transitions between different tasks. Four experimenters were available, if needed, to clarify the instructions or provide other assistance. This worked flawlessly, and allowed us to handle the large numbers of test persons efficiently.

Although we are still in the process of analyzing the massive amount of data collected during the experiments, we discussed a few initial results. The general opinion on the XML retrieval systems was equivocally positive. Departing from earlier systems that return ranked lists of XML elements, both the Daffodil/HyREX and xmlfind are grouping the found XML elements per article (similar to the Fetch & Browse task in the Adhoc Track). Test persons seem to conceive the resulting system as an article retrieval engines with some additional features—yet with great appreciation for the bells and whistles!

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 612-13-001, 612.000.106, 612.000.207, 612.066.302, 612.069.006, and 640.001.501.

References

1. T. Bakker, M. Bedeker, S. van den Berg, P. van Blokland, J. de Lau, O. Kiszer, S. Reus, and J. Salomon. Evaluating XML retrieval interfaces: xmlfind. Technical report, University of Amsterdam, 2005.
2. Daffodil. Distributed Agents for User-Friendly Access of Digital Libraries, 2005. `http://www.is.informatik.uni-duisburg.de/projects/daffodil/`.
3. HyREX. Hyper-media Retrieval Engine for XML, 2005. `http://www.is.informatik.uni-duisburg.de/projects/hyrex/`.
4. B. Larsen, S. Malik, and T. Tombros. The interactive track at inex2005. In *This Volume*, 2005.
5. Lucene. The Lucene search engine, 2005. `http://lucene.apache.org/`.
6. B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-Based Approch to XML Retrieval. In *INEX 2003 Workshop Proceedings*, pages 19–26, 2004.

Kyungpook National University at INEX 2005: Interactive Track

Heesop Kim and Heejung Son

Department of Library & Information Science, Kyungpook National University

Daegu, 702-701, South Korea

heesop@knu.ac.kr; sonhjung@postech.ac.kr

Abstract. The two basic objectives of this study were: (1) to investigate how the interface designs influence the searchers' satisfaction and (2) to examine how the users' backgrounds affect in their searching of the XML documents. To achieve these objectives, we used INEX iTrack Daffodil retrieval system which provided by the organizers of the INEX 2005 and recruited 12 students in different backgrounds. The major changes of the system interface are (1) structured presentation of documents and elements in the result list, (2) adoption of more specified search window and simpler relevance assessment scale, (3) application of small eye symbol for already viewed documents or elements and (4) adoption of own tasks from the searchers. The responses from the questionnaires of this year were analyzed and compared with those of INEX 2004 Interactive Track. In results, users' satisfaction on understandability of tasks and similarity to other tasks were increased, whereas their satisfaction on ease of learning the system, ease of using the system, and understandability of using the system were decreased with newly designed system interface in INEX 2005. To examine our second objectives we chosen two test groups with different backgrounds in this experiment. One group is consisted of 6 students who are majoring in Computer Engineering (CE) and expected to be more familiar with the IEEE test collection; the other is consisted of 6 students who are majoring in Library and Information Science (LIS) and expected to have more previous knowledge of searching techniques. 36 transaction log files, questionnaires and interview results were analyzed to look into the differences in search characteristics and attitude to the search system evaluation between these two groups. It is noteworthy that LIS group tends to show more activities in their query iteration, and spend more time in their searching than the CE group in XML document searching. However, CE group (Mean = 7.6 documents or components) showed more elaborate in their relevance assessments than the LIS group (Mean = 4.9 documents or components) do. The majority of CE group pointed out 'delay of response time,' 'broken display of numerical formulas and images,' and 'limited of Boolean operators' as the weakness of the newly designed interface, whereas the majority of LIS group

pointed out the adaptation of ‘search within the results,’ capability and consistency of ‘viewed ones indicating’ which related with the common search capabilities in general IR system. We need an in-depth analysis of the differences of these two groups in the light of interface design for more effective XML retrieval system.

B^3-SDR @ Interactive Track: User Interface Design Issues

Roelof van Zwol[1], Sandor Spruit[1], and Jeroen Baas[2]

[1] Utrecht University, Department of Computer Science, Center for Content and Knowledge Engineering, Utrecht, the Netherlands {roelof, sandor}@cs.uu.nl
[2] Elsevier, User Centered Design, Amsterdam, the Netherlands
j.baas@elsevier.com

Abstract. For structured document retrieval systems to work in practice, it is necessary to realize a match between the functionality of the interface offered to the user, and the behavior of the retrieval strategy that drives the retrieval process. The main difference between the traditional retrieval system and the structured document retrieval system, is the capability of the retrieval strategy to retrieve relevant document fragments, i.e. XML elements, as the result of a user information need instead of complete documents. In addition, a structured document retrieval system allows the user to specify his information need using both structural and content-based constraints. The potential to the user is enormous, provided that this new functionality can be efficiently integrated into the current user model for information seeking. This influences both the query formulation process, and the presentation of results.

In this article we present how various aspects of the retrieval strategy are integrated into the user interface for B^3-SDR . Within INEX, the de-facto language for specifying the user information need is NEXI, an XPath-flavored language, with support for content-based information retrieval. A user specifies the structural aspects of his information need using the path-expressions offered by XPath, while the content-based constraints of the information need are expressed using filters on the path-expressions. Although the NEXI query language is relatively simple, on can not expect from the average user that he 'speaks' fluent NEXI. Therefore the B^3-SDR interface offers the Bricks graphical approach for constructing NEXI queries, besides the command-line option.

With respect to the presentation of results, the B^3-SDR system offers a number of alternatives that impact the way a user evaluates the results retrieved by the system. Besides the problem of overlapping results, which occurs when the system retrieves an XML element and its 'overlapping' parent, the system also provides different document views.

By default the system provides an *in-context* view of a document, when the user wants to inspect one of the retrieved document fragments. In this case, the document fragment is highlighted within the original document. Alternatively, a user can request an *isolated* view of the document fragment, where the system only displays the part that was assumed relevant. The user has the option to inspect the entire document. The main objective of Task C of the INEX Interactive Track in 2005 was to examine this particular feature.

Context matters? User behaviour and element retrieval

Ragnar Nordlie and Nils Pharo
Oslo University College, Dept. of Journalism, Library and Information Science

The fundamental proposition underlying XML-assisted retrieval, at least as understood in the INEX context, is that users of information retrieval systems are better assisted if they are presented with search results that not only identify relevant documents, but also point directly to relevant parts of the documents. The validity of this proposition depends on at least two factors: that users are actually able to identify relevance upon being shown parts of documents at various levels of granularity; and if so, that they behave accordingly, i.e. that they will not feel compelled, for some reason, to view the whole article even if they have been shown and have looked at the relevant parts. In our paper we intend to investigate these two questions based on data from the 2004 and 2005 INEX Interactive Track. Our hypotheses are that users are more assured in their relevance judgements when they are based on full articles, and that their preferred search behaviour includes verifying relevance judgements by looking at the context of the viewed element.

In our paper "Context matters: an analysis of XML documents", we investigated the relevance assessments of the users in the 2004 INEX Interactive experiments, expressed in a three-part scale of "usefulness" and "specificity", respectively. We found that users looked relatively more frequently at whole articles than at elements of articles; that users had difficulties in dealing with the combined relevance measure that were used, but that they were more likely to judge a full article "useful" than an element of the same article, and that there was no corresponding tendency to judge an element more "specific" than the full article. Overall, there was a general tendency to judge elements as more relevant than full articles, but this did not hold for judgements of elements and the full text of the same article.

The 2005 user assessments make use of a single measure of relevance. This will hopefully eliminate some of the uncertainty in the interpretation of the users' judgements and provide a clearer picture of users' ability to judge relevance on various levels of granularity. We will follow up last year's investigation on the 2005 data, and extend it to consider a number of factors that might influence users' ability to make relevance judgements on document parts, and that we did not look into in the previous paper, notably

- the sequence in which the users view the components (full article first vs. component first)
- the size and granularity level of the component
- user- and task-dependent factors such as type of question, user experience and domain knowledge, and perceived and experienced difficulty of the search task.

A preliminary investigation of interview data from the 2005 data collection seem to indicate that users feel the need to assure themselves of the validity of their judgement by looking at the full article, regardless of what relevance judgement they have given to the components they look at. We will study the search logs to investigate to which extent this perceived need for assurance is reflected in actual user behaviour. A combined study of users' relevance assessments and actual search behaviour will provide a rich background for determining the potential usefulness of a retrieval system based on XML-partitioned documents.

XML documents clustering by structures with XCLS

Richi Nayak and Sumei Xu

School of Information Systems, Queensland University of Technology
Brisbane, Australia {r.nayak@qut.edu.au}

Abstract. We present the results of clustering the structural descriptions (ordered labelled trees) of XML documents with XCLS. We have reported 5 sub-tasks corresponding to 5 corpuses. XCLS is a novel clustering algorithm to assemble heterogeneous XML documents by measuring their level similarity with a *global* criterion function. XCLS does not require the pair-wise similarity to be computed between two individual documents, rather measures the similarity at clustering level utilising structural information of XML documents. XCLS utilises a Level structure format to represent the XML documents for efficient processing. The clustering quality depends on the calculation of level similarity and whether the level similarity can represent the documents' structural similarity correctly.

1 Introduction

With the continuous growth in XML data, the ability to manage collections of XML documents and to discover knowledge from them becomes essential for databases and information retrieval systems. Several tools are developed to deliver, store and querying XML data [2,4,10]. However, they do require efficient data management techniques such as indexing based on structural similarity to support an effective document storage and retrieval. The grouping of similar XML documents according to structural similarity helps to achieve this.

The clustering process categorizes the XML data based on a similarity measure without the prior knowledge on the taxonomy. There exists a number of clustering methods dealing with (structured) database objects and text (unstructured) data [2, 21]. The XML documents are semi-structured and hierarchal, representing not only the values of individual items but also the relationships between data items. The inherent flexibility of XML, in both structure and semantics, poses new challenges to find similarity among XML data.

Research on measuring the similarity of XML documents is gaining momentum [1,3,6,7,8]. Most of these methods rely on the notion of tree edit distance developed in combinational pattern matching – finding common structures in tree collection [14]. (A document is usually represented as a tree structure.) These methods are built on pair-wise similarity between two documents or document structures (represented as trees). The pair-wise similarity is measured using the *local* functions between each pairs of objects to minimise the intra-cluster similarity and maximize the inter-cluster similarity. The similarity value between each pair of trees is mapped into a *similarity matrix*. This matrix becomes the input to the clustering process using either the hierarchical agglomerative clustering algorithm or k-means algorithms [5]. They are generally computationally expensive when the data sources are large due to the need of pair wise similarity matching among diverse documents.

Our strategy is quite different from these pair-wise clustering approaches. It is inspired by the clustering algorithms developed for transactional data, characterized by high dimensionality and large volume, such as LargeItem [13] and Clope [12] that do not need to compute a pair wise similarity. These methods define the clustering criterion functions on the cluster level calling *global* similarity measures to optimize the cluster parameters. Each new object is compared against the existing clusters instead of comparing against the individual objects. Since the computations of these global metrics are much faster than that of pair-wise similarities, global approaches are very efficient. These methods are not suitable for XML documents. They do not consider the hierarchical structure of a document, (i.e. the level positions, context or relationships of elements) that is the main essence of XML data.

We apply the XML Clustering with Level Similarity (XCLS) algorithm to cluster the given INEX XML corpus data by structures only using *global* similarity measures. XCLS uses the *Level structure* format to represent the documents and the *LevelSim global* criterion function to measure the similarity at clustering level utilising the hierarchal relationships between elements of documents.

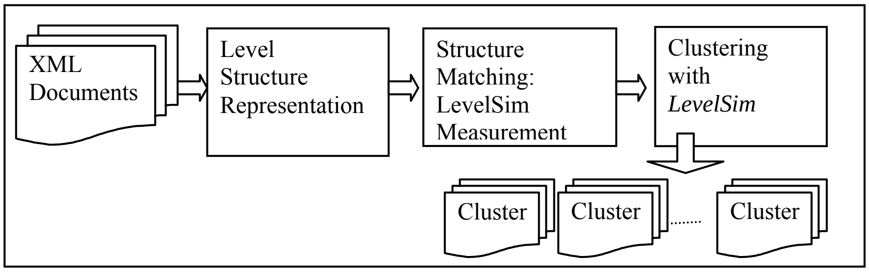

XML Documents
Level Structure Representation
Structure Matching: LevelSim Measurement
Clustering with *LevelSim*
Cluster
Cluster
....
Cluster

Figure 1: The XCLS approach

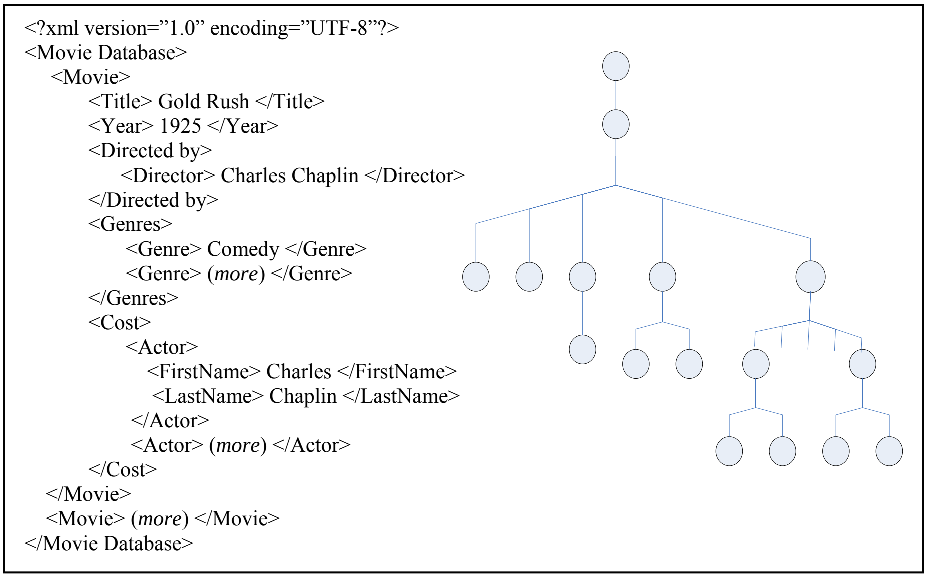
2 XML documents clustering with LevelSim (XCLS)

Given a set of XML documents *D*, the clustering solution *C*= {C_1, C_2 … *Cq*} is a partition of {D_1, D_2 … D_n}, such that [C_1 U C_2 U … U *Cq* = {D_1, D_2 … D_n}] and [C_i ∩ C_j = Φ] for any 1 ≤ i, j ≤ q, where *n* is the number of XML documents; *q* is the number of clusters. *Ci* denotes a cluster in the clustering solution. D_i denotes an XML document represented as a (newly introduced) level structure format.

The XCLS method (figure 1) first represents the XML documents in the Level structure form. The *global* criterion function, LevelSim, measures the similarity at clustering level considering hierarchical structures of the XML documents to cluster them, thus ignoring the need to compute the pair-wise similarity between two individual documents.

2.1 Level structure: Inferring of XML documents structure

To be applicable to general Web documents and any type of XML documents – well-formed, valid and ill-formed – the XCLS algorithms starts with inferring the structural information within the document represented as the ordered labelled tree.

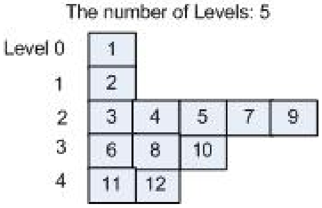


```
<?xml version="1.0" encoding="UTF-8"?>
<Movie Database>
   <Movie>
        <Title> Gold Rush </Title>
        <Year> 1925 </Year>
        <Directed by>
             <Director> Charles Chaplin </Director>
        </Directed by>
        <Genres>
             <Genre> Comedy </Genre>
             <Genre> (more) </Genre>
        </Genres>
        <Cost>
             <Actor>
                 <FirstName> Charles </FirstName>
                  <LastName> Chaplin </LastName>
              </Actor>
              <Actor> (more) </Actor>
        </Cost>
   </Movie>
   <Movie> (more) </Movie>
</Movie Database>
```

Figure 2: An XML Document (X_Movie) & its tree representation (T_Movie)

When inferring the structure, the focus is on paths of elements with content values (i.e. leaves in a document tree), without considering attributes in an XML document. The inferred structure preserves the hierarchy and the context of elements of the documents. For an element, multiple instances of values are ignored, as these are redundant information for structure's presentation and the occurrence of elements is not important for clustering in most cases. Additionally, XCLS does not consider the order of sibling when computing the similarity, as the order of sibling is not important for the clustering.

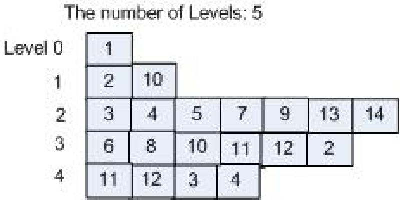
Figures 2 shows a XML document (X_Movie) and its corresponding structural tree (T_Movie). In order to enhance the clustering speed, the name of each element is denoted by a distinct integer.



The number of Levels: 5

Level 0	1				
1	2				
2	3	4	5	7	9
3	6	8	10		
4	11	12			

Figure 3: Level structure for T_Movie



The number of Levels: 5

Level 0	1						
1	2	10					
2	3	4	5	7	9	13	14
3	6	8	10	11	12	2	
4	11	12	3	4			

Figure 4: Level structure of the cluster

We define a novel concept of the **level structure** to show the level and the elements in each level of a tree structure. The Figure 3 shows the level structure for T_Movie. The level structure contains the information such as element values and their occurrences and levels in the hierarchy. We also define the level structure for clusters preserving the hierarchical information of document. Each level of a cluster contains a collection of elements of the same level for all documents within the cluster. The figure 5 shows a tree structure of a document on Actor information and its corresponding level structure. The Figure 4 shows the level structure of a cluster containing both the Movie and Actor documents.

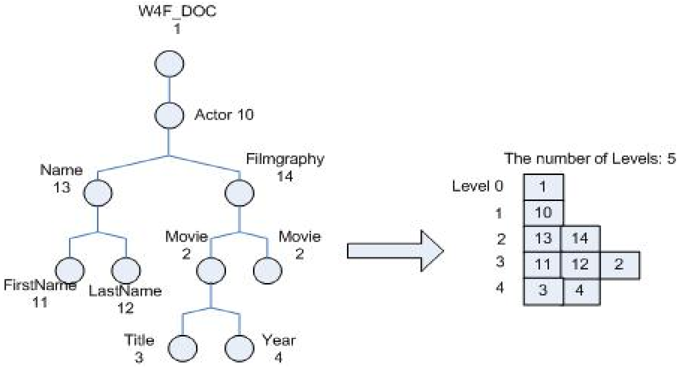


Figure 5: T_Actor and its level structure

2.2 Clustering *global* criterion function with level similarity (LevelSim)

Considering the level information and elements' relationships/context of XML data, a new solution for measuring structural similarity between two XML objects (cluster to tree, tree to tree, cluster to cluster) is developed which is called Level Similarity (LevelSim). It measures the common items in each corresponding level, and allocates different weight according to the level (i.e. high level (e.g. root) has more weight than low level (e.g. leaf)).

Elements are matched according to the level information of each object. We define the criterion function for matching two objects, a tree and a cluster, as each tree will be matched against the existing clusters. Cluster may contain only one tree as well.

The order of matching between two objects is important due to the structural information present in an XML document. The LevelSim when matching object 1 (tree) to object 2 (cluster) is defined as:

$$LevelSim_{1\rightarrow 2} = \frac{0.5 \times ComWeight_1 + 0.5 \times ComWeight_2}{TreeWeight \times Z}$$

$$= \frac{0.5 \times \sum_{i=0}^{L-1} CN_1^i \times (r)^{L-i-1} + 0.5 \times \sum_{j=0}^{L-1} CN_2^j \times (r)^{L-j-1}}{(\sum_{k=0}^{L-1} N^k \times (r)^{L-k-1}) \times Z}$$

$ComWeight_1$ denotes the total weight of the common elements in all levels considering the level information of object 1;

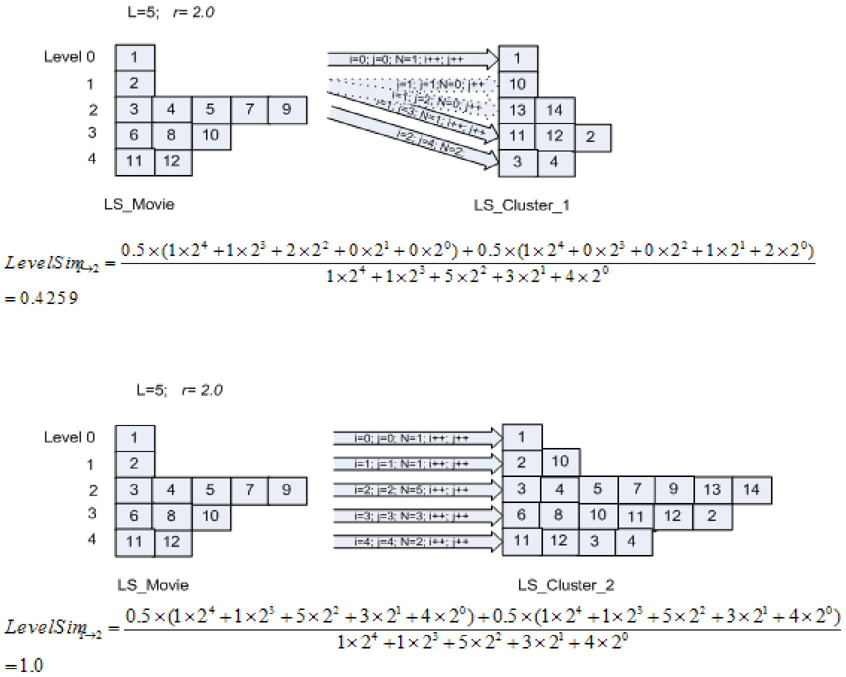
$ComWeight_2$ denotes the total weight of the common elements in all levels considering the level information of object 2;

TreeWeight denotes the total weight of all items in each level of the tree (object 1);

Z is the size of the cluster (the number of trees within cluster);
CN_1^i denotes the sum of occurrences of every common elements in level i of object 1;
CN_2^j denotes the sum of occurrences of every common elements in level j of object 2;
N^k denotes the number of elements in level k of the tree.

r is the increasing factor of weight, which is usually larger than 1 to indicate that the higher level elements have larger than lower level elements called as "Base Weight";
L is the number of levels in the tree.

LevelSim yields the values between 0 and 1; 0 indicates completely different objects and 1 indicates homogenous objects.



$$LevelSim_{1\to2} = \frac{0.5\times(1\times2^4+1\times2^3+2\times2^2+0\times2^1+0\times2^0)+0.5\times(1\times2^4+0\times2^3+0\times2^2+1\times2^1+2\times2^0)}{1\times2^4+1\times2^3+5\times2^2+3\times2^1+4\times2^0}$$
$$=0.4259$$

$$LevelSim_{1\to2} = \frac{0.5\times(1\times2^4+1\times2^3+5\times2^2+3\times2^1+4\times2^0)+0.5\times(1\times2^4+1\times2^3+5\times2^2+3\times2^1+4\times2^0)}{1\times2^4+1\times2^3+5\times2^2+3\times2^1+4\times2^0}$$
$$=1.0$$

Figure 6: Two different cases showing the process of matching a tree to a cluster

2.3 The process of structure matching between two objects

The steps to match elements of a tree (object 1) to elements of a cluster (object 2) are as follows:

1. Start with searching common elements in the 1st level of both objects. If at least one common element is found, mark the number of common elements with the level number in object 1 (N_1^0) and the number of common elements with the level number in object 2 (N_2^0), then go to step 2. Otherwise, go to step 3.
2. Move both objects to next level (level i++, level j++) and search common elements in these new levels; If at least one common element is found, mark the

number of common elements with the level number in object 1 (N_1^i) and the number of common elements with the level number in object 2 (N_2^j), then go to step 2. Otherwise, go to step 3.

3. Only move object 2 to next level (level j), then search common elements in the original level (i) of object 1 and the new level (j) of object 2. If at least one common element is found, mark the number of common elements with the level number in object 1 (N_1^i) and the number of common elements with the level number in object 2 (N_2^j), then go to step 2. Otherwise, go to step 3.
4. Repeat the process until all levels in either object have been matched.

After completion of structure matching the Level Similarity (LevelSim) is computed.

The Figure 6 shows two different cases of matching object 1 to object 2. In the first case, object 1 is the tree T_Movie, object 2 is the cluster only containing the tree T_Actor. In the second case, object 1 is the same, but object 2 is the cluster containing both T_Actor and T_Movie.

The operation *LevelSim* is not transitive. As a result, the level similarity between two objects is computed, $LevelSim_{1\to2}$ & $LevelSim_{2\to1}$, and the larger value between these two is chosen:

$$LevelSim = LevelSim_{1\to2} > LevelSim_{2\to1} \; ? \, LevelSim_{1\to2} : LevelSim_{2\to1} \; .$$

The *LevelSim* emphasizes different importance of elements in different level positions by allocating different weight to them. The hierarchical relationships of elements are also considered by counting occurrences of common elements sharing common ancestors. The derivation of level structure from a tree is straightforward; and the computation of *LevelSim* is quite effective.

```
/*Phase 1 – Allocation*/
For all XML trees to be clustered
•   read the next tree (represented as level structure);
•   compute the LevelSim between the tree and each existing cluster;
•   assign the tree to an existing cluster if maximum of LevelSim(s) is found
    between two objects and $LevelSim > LevelSim\_Threshold$;
•   otherwise, form a new cluster containing the tree.

/*Phase 2 – Reassignment (adjustment) */
For all XML trees
•   read the next tree (i.e. level structure);
•   compute the LevelSim between the tree and each existing cluster;
•   reassign the tree to an existing cluster if maximum of LevelSim(s) is found
    between two objects and $LevelSim > LevelSim\_Threshold$;
•   otherwise, form a new cluster containing the tree.

/*Stop if there is ano improvement in two iterations*/
```

Figure 7: The sketch of XCLS core clustering algorithm

2.4 Clustering with Level Similarity

The problem is to group each XML document within the XML sources into an existing cluster that have the maximum level similarity (*LevelSim)* or to a new cluster. The figure 7 outlines the algorithm that includes two phases of allocation and reassignment. In the allocation phase, clusters are progressively formed driven by the criterion function *LevelSim*. After that in reassignment phase, only a few iterations are required to refine the clustering and optimize the *LevelSim*. If no changes in the clustering solution formed by previous iteration, the clustering process is stopped. The XCLS algorithm uses a user-defined threshold *LevelSim_Threshold* below that the cohesion between two objects is not considered e.g., LevelSim < *LevelSim_Threshold*. This threshold (between 0 and 1) can be set according to the application requirement, if only highly homogenous documents are to be grouped the threshold is set higher (near 1) otherwise it is set as lower value (near 0).

3 Experimental Evaluation

The data used in experiments are the MovieDB corpus and the INEX corpus. The MovieDB corpus has 11 thematic classes and 11 possible structure classes. The INEX corpus has 6 thematic classes (Computer, Graphics, Hardware, Artificial Intelligence, Internet, Parallel) and 2 structural classes (Transaction journals vs Others). Some documents could not be parsed therefore are ignored during the clustering process. Some features of the corpus are shown in the table 1.

Dataset	**Bad trees (ill-formed)**	**Trees to be Clustered**	**Distinct Labels**
m-db-s-0-test	5	4811	195
m-db-s-1-test	4	4814	197
m-db-s-2-test	9	4809	197
m-db-s-3-test	9	4809	194
inex-s-test	0	6053	173

Table 1: Features of the data sets

The intra-cluster similarity is computed by measuring the level similarity between a pair of trees (i.e. XML document structures) within a cluster. The intra-cluster similarity of a cluster is the average of all pair-wise level similarities within the cluster. The intra-cluster similarity of the clustering solution is the average of the intra-cluster similarities of clusters taking into account the number of trees within each cluster. The Level Similarity between two trees is quite similar to the level similarity measure between a tree and a cluster. It is defined as:

$$LevelSim_{i\to j} = \frac{ComWeight_1(i\to j) + ComWeight_2(i\to j)}{TreeWeight_i + TreeWeight_j}$$

$$LevelSim_{j\to i} = \frac{ComWeight_1(j\to i) + ComWeight_2(j\to i)}{TreeWeight_i + TreeWeight_j}$$

$$LevelSim_{i,j} = LevelSim_{i \to j} > LevelSim_{j \to i}\ ?\ LevelSim_{i \to j} : LevelSim_{j \to i}$$

Where $ComWeight_1(i \to j)$ is the total weight of the common items in each level between tree$_i$ and tree$_j$ while matching tree$_i$ to tree$_j$ and using the level number of tree$_i$; $ComWeight_2(i \to j)$ is the same as above but using the level number of tree$_j$; $TreeWeight_i$ and $TreeWeight_i$ are the total weight of all items in each levels of tree$_i$ and tree$_j$ respectively, and are calculated in the same way when matching tree to cluster.

The intra-cluster similarity of a cluster C_i is defined as

$$IntraSim(C_i) = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} LevelSim_{i,j}}{0.5 \times n \times (n-1)}$$ where n is the number of trees in C_i.

For a clustering solution $C = \{C_1, C_2 \ldots C_k\}$, the intra-cluster similarity of the clustering is defined as:

$$IntraSim = \frac{\sum_{i=1}^{k} IntraSim(C_i) \times n_i}{N}$$ where n_i is the number of trees in C_i, N is the total number of trees in the Clustering and k is the number of clusters in the solution.

The **inter-cluster similarity** measures the separation among different clusters. It is computed by measuring the level similarity between two clusters. The inter-cluster similarity of the clustering solution is the average of all pair-wise level similarities of two clusters. The Level Similarity between two clusters is defined as similar to two trees, using the objects as clusters:

$$LevelSim_{i \to j} = \frac{ComWeight_1(i \to j) + ComWeight_2(i \to j)}{ClusterWeight_i + ClusterWeight_j}$$

$$LevelSim_{j \to i} = \frac{ComWeight_1(j \to i) + ComWeight_2(j \to i)}{ClusterWeight_i + ClsuterWeight_j}$$

$$LevelSim_{i,j} = LevelSim_{i \to j} > LevelSim_{j \to i}\ ?\ LevelSim_{i \to j} : LevelSim_{j \to i}$$

The inter-cluster similarity for the clustering solution $C = \{C_1, C_2 \ldots C_k\}$ is:

$$InterSim = \frac{\sum_{i=1}^{k} \sum_{j=i+1}^{k} LevelSim_{i,j}}{0.5 \times k \times (k-1)}$$ where k is the number of clusters in the clustering.

All the experiments were done in the machine with 2.8GHZ Intel Celeron CPU and 1G of RAM. Table 2 and 3 show the performance of XCLS on these data sets with the various parameters set during the experiments. One of the experimental runs has the parameter of max clusters set as 1000. It means XCLS automatically groups the documents into clusters without a prior knowledge. Another experimental run has the parameter of max clusters set as required clusters. In the first clustering process, the number of clusters are usually large (>100) depending on the similarity threshold. In the following iterations, it decreases automatically. Clustering is not controlled by level threshold, but is controlled by the level similarity between the tree and the original cluster or other clusters in the following iterations.

There are two parameters in XCLS: Base Weight and Similarity Threshold that need to set during run-time. A number of experiments with different values are carried out to evaluate the XCLS performance. The experiments conclude XCLS is not hypersensitive with these parameters Experiments have found the suitable values of *'Base Weight' between 1.3 and 2.5* and of *'similarity threshold' between 0.3 and 0.7.*

Dataset	Run	BaseWeight	SimThreshhold	IntraSim	InterSim
m-db-s-0-test	1	1.45	0.65	0.793437	0.093376
	2	1.5	0.65	0.791116	0.128861
m-db-s-1-test	1	1.5	0.25	0.664616	0.230575
	2	1.5	0.5	0.661578	0.183368
m-db-s-2-test	1	1.45	0.29	0.6006	0.270311
	2	1.5	0.8	0.608698	0.274835
m-db-s-3-test	1	1.45	0.235	0.562381	0.317
	2	1.5	0.8	0.570773	0.290998
inex-s-test	1	1.45	0.5	0.712204	0.180065
	2	1.5	0.5	0.71986	0.189902

Table 2: XCLS accuracy performance

Dataset	Run	PreProcessTime(S)	ClusteringTime(S)	Iterations
m-db-s-0-test	1	6.172	10.5	7
	2	6.094	10.281	7
m-db-s-1-test	1	7.172	14.031	8
	2	8.031	10.435	6
m-db-s-2-test	1	10.781	35.515	15
	2	11.156	32.468	15
m-db-s-3-test	1	11.953	41.687	15
	2	11.578	41.281	15
inex-s-test	1	27.453	12.14	4
	2	27.406	12.234	4

Table 3: XCLS scalability performance

Table 4 and 5 shows the time performance of XCLS on m-db-s-0 and inex-s. The pre-processing time includes the generation of level structure for all documents. The clustering time includes the time to group the represented level structures in clusters using LevelSim measurement.

The time complexity of pair-wise clustering algorithms is at least $O(m^2)$, where m is the number of elements in the documents. This is infeasible for large amount of data. XCLS computes the structure similarity between the document structure and clusters avoiding the need of pair-wise comparison. Its time complexity is *O(m×c×p×n)*: *m* is number of elements in documents; *c* is number of clusters; *p* is number of iterations; *n* is number of distinct elements in clusters. The documents grouped into a cluster should have similar structures and elements. So the number of distinct elements in clusters should always be less than the distinct elements in documents. The number of iterations is usually small and its maximum can be configured. Therefore, if the number of clusters is less than the number of documents (that is usually the case) the time cost is **linear** to the number of documents.

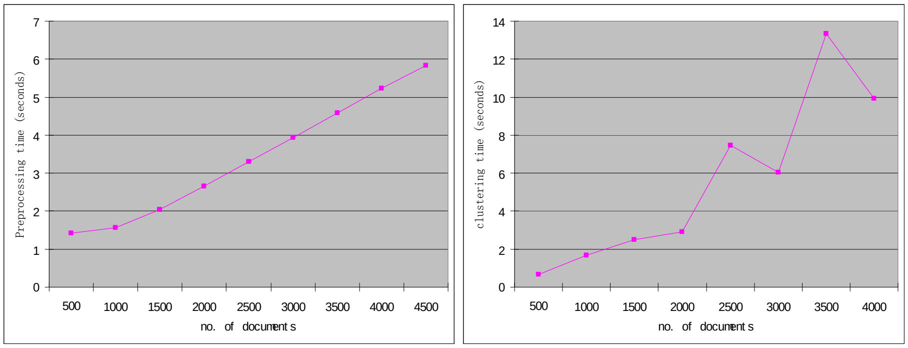


Table 4: The execution time of XCLS on m-db-s-0

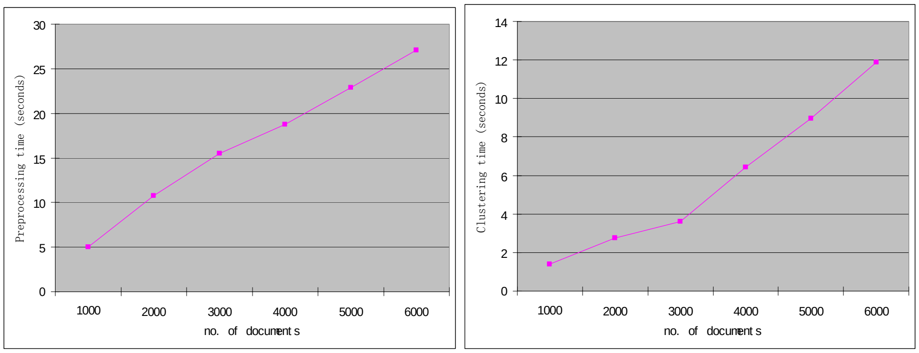


Table 5: The execution time of XCLS on inex-s

Experiments were also performed to test the sensitivity of XCLS on the order of input data using the same data source. The clustering process of XCLS was repeated many times with the input data sources presented in randomly order to XCLS. The results were not exactly same, but were very close. It shows that XCLS is *independence of the order input*.

4 Conclusions and Future Work

We used the XCLS algorithm based on the intuitive idea of the global criterion function *LevelSim* for easy and effective clustering solution of movie and INEX data sets by their structures. XCLS efficiently clusters a large number of documents from different domains and with a large number of distinct labels, without knowing the number of clusters. XCLS evaluates the structural similarity in the clustering level instead of document level, thus it does not need to calculate the pair-to-pair distance between documents. It considers the hierarchical structure of XML documents in the algorithm, but is not strict to the parent-child relationship, which makes it more reasonable in the heterogeneous environment. The experiments show the effectiveness of XCLS using the *global* criterion function in comparison to the clustering algorithms using a *locally* defined criterion function based on pair-wise similarity.

The main weakness of XCLS is that it can not cluster XML documents into a specific number directly, which means, the information of number of clusters does not help the clustering process at all. However, this information provides an important hint for the clustering. XCLS is not good at recognizing the details of the structure. The cluster is presented by level structures, which only record the elements and their occurrences, but not the trees they come from; i.e. it is impossible to recognize which part is from which tree in the level structures. Therefore, if trees have complex overlapped structures, the calculation of level similarity between a tree and a cluster is not accurate. It was easy to get the result of m-db-s-0 and inex-s, which have not been transformed. But for the corpuses m-db-s-1, m-db-s-2 and m-db-s-3, it is a challenge for XCLS. It is really hard to get the right values for the parameters.

XCLS needs some future work to improve its effectiveness. XCLS ignored the sematic similarity among documents, which is impractical in the flexible environment on web since people may use different tags to describe the same thing. As WordNet can organize English words into synonym sets and defined different relations link the synonym sets, it can be added to the pre-processing phase to recognize the sematic similarity among elements.

References

1. Bertino, E., Guerrini, G. & Mesiti, M. (2004). A Matching Algorithm for Measuring the Structural Similarity between an XML Document and a DTD and its applications. Information Systems, 29(1): 23-46.
2. Boag S. Chamberlin D, Fernández M, Florescu D, Robie J and Siméon J. “XQuery 1.0: An XML Query Language” W3C Working Draft, September, 2005. http://www.w3.org/TR/2005/WD-xquery-20050915/
3. Flesca, S., Manco, G., Masciari, E., Pontieri, L., & Pugliese, A. (2005). *Fast Detection of XML Structural Similarities.* IEEE Transaction on Knowledge and Data Engineering, Vol 7 (2), pp 160-175.
4. Guardalben, G. (2004), *Integrating XML and Relational Database Technologies: A Position Paper*, HiT Software Inc, retrieved May 1st ,2005, from http://www.hitsw.com/products_services/whitepapers/integrating_xml_rdb/integrating_xml_white_paper.pdf.
5. Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data Clustering: A Review. *ACM Computing Surveys (CSUR), 31*(3), 264-323.
6. Leung, H.-p., Chung, F.-l., & Chan, S. C.-f. (2005). On the use of hierarchical information in sequential mining-based XML document similarity computation. *Knowledge and Information Systems, 7*(4),pp 476-498.

7. Nayak R and Iryadi W (2006). XMine: A methodology for mining XML structure. To appear in *The Eighth Asia Pacific Web Conference*. January 2006, China.
8. Nayak R & Xia, F. B. (2004). "Automatic integration of heterogenous XML-schemas", *Proceedings of the International Conferences on Information Integration and Web-based Applications & Services*. Jakarta, Indonesia, Sec 27-29, pp. 427-437.
9. Nayak, R., Witt, R., and Tonev, A. (2002) Data Mining and XML documents, International Conference on Internet Computing, USA.
10. Xylem L. (2001). Xylem: A dynamic Warehouse for XML data of the Web," IDEAS'01, pp3-7, 2001.
11. Yergeau, F, Bray T, Paoli *J,* Sperberg-McQueen, C M and Maler E. (2004). Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation, February 2004, http://www.w3.org/TR/2004/REC-XML-20040204/
12. Ying Y, Guan X and You J. (2002), CLOPE: A Fast and effective clustering algorithm for transactional data,
13. Wang, K., Xu, C. (1999), *Clustering Transactions Using Large Items*, in the proceedings of ACM CIKM-99, Kansas, Missouri.
14. Zhang, K., & Shasha, D. (1989). *Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems*. SIAM Journal Computing, 18(6), 1245-1262.
15. Zhao, Y., & Karypis, G. (2002). *Evaluation of Hierarchical Clustering Algorithms for Document Datasets.* The 2002 ACM CIKM, USA.

A Flexible Structured-based Representation for XML Document Mining

Anne-Marie Vercoustre, Mounir Fegas, Saba Gul, and Yves Lechevallier

INRIA, Rocquencourt, France

Abstract. This paper reports on the INRIA group's approach to XML mining while participating in the INEX XML Mining track 2005. We use a flexible representation of XML documents that allows taking into account the structure only or both the structure and content. Our approach consists of representing XML documents by a set of their sub-paths, defined according to some criteria (length, root beginning, leaf ending). By considering those sub-paths as words, we can use standard methods for vocabulary reduction, and simple clustering methods such as K-means that scale well. We actually use an implementation of the clustering algorithm known as dynamic clouds that can work with distinct groups of independent variables. This is necessary in our model since embedded sub-paths are not independent.

Sequential Pattern Mining for Structure-based XML Document Classification

Calin Garboni, Florent Masseglia and Brigitte Trousse

INRIA Sophia Antipolis, AxIS Research Team
2004 route des lucioles - BP 93
06902 Sophia Antipolis, France
E-mail: {Surname.Name}@inria.fr

Abstract. This article is related to a new XML Document classification method based on extracting frequent sequential patterns applied on the structure of the documents of each predefined clusters. The article will be organised like this. Section 1. will define the sequential pattern mining problem in large databases and will remind the main sequential pattern mining definitions. Section 2. will introduce the principles of structure discovery from a set of XML documents. Our goal is to extract a schema that will be representative of the whole set of documents. In this context, "representative" will be interpreted as "frequent". In fact, we consider that a frequent sub-tree in a collection of XML document may be considered as a interesting knowledge regarding this collection. This sub-tree can be exploited as a further DTD or may stand for a characteristic of the collection (this is the main idea of our approach). After presenting related work, we describe in section 4 our approach composed of different steps (cleaning and format transformation, predefined clusters characterisation in terms of frequent sequential patterns, used measures to classify each document). Section 5 will present our experiments and our results on the Movie collection.

Keywords: XML Document, structure mining, clustering, classification, frequent sequential pattern

Categorization and Clustering of XML documents using Structure and Content Information

Ludovic DENOYER, Patrick GALLINARI

LIP6 - University of Paris 6

Abstract. The widespread use of XML has urged the need to develop tools to efficiently store, access and organize XML corpus. The INEX initiative has resulted in major improvements in XML retrieval systems, but today, related tasks, like categorization or structure matching, should be investigated. We consider here the problem of clustering XML documents using their structure.

In this paper, we propose a Belief networks-based stochastic model which is able to describe different kind of relations between structural elements. We show how these models can be used to transform semi structured document into vectors. We then use these vectors for clustering and categorization of XML documents. We test our model on the different corpora provided by the INEX XML Mining Challenge - http://xmlmining.lip6.fr.

Transforming XML trees for efficient classification and clustering

Laurent Candillier, Isabelle Tellier, Fabien Torre

GRAppA - Charles de Gaulle University - Lille 3
http://www.grappa.univ-lille3.fr
candillier@grappa.univ-lille3.fr
Pertinence - 32 rue des Jeuneurs -75002 Paris
http://www.pertinence.com

Abstract. Most of the existing methods we know to tackle datasets of XML documents directly work on the trees representing these XML documents. We investigate in this paper the use of a different kind of representation for the manipulation of XML documents. Our idea is to transform the trees into sets of attribute-values, so as to be able to apply various existing methods of classification and clustering on such data, and benefit from their strengths. We apply this strategy both for the classification task and for the clustering task using the structural description of XML documents alone. For instance, we show that the use of boosted C5 leads to very good results in the classification task of XML documents transformed so. The use of SSC in the clustering task benefits from its ability to provide as output an interpretable representation of the clusters found. Finally, we also propose an adaptation of SSC for the classification of XML documents, so that the produced classifier is understandable.

Clustering XML Documents using Self-Organizing Maps for Structures

F. Trentini, M. Hagenbuchner, A. Sperduti3, A.C. Tsoi, F. Scarselli1, and M. Gori

University of Siena, Siena, Italy
University of Wollongong, Wollongong, Australia
University of Padova, Padova, Italy
Australian Research Council, Canberra, Australia

Abstract. Self-Organizing Maps capable of encoding structured information will be used for the clustering of XML documents. Documents formatted in XML are appropriately represented as a graph structure. It will be shown that the Self-Organizing Maps can be trained in an unsupervised fashion to group XML structured data into clusters, and that this task is scaled in linear time with increasing size of the corpus.

INEX 2005 Multimedia Track

Roelof van Zwol[1], Gabriella Kazai[2], and Mounia Lalmas[2]

[1] Utrecht University, Department of Computer Science, Center for Content and
Knowledge Engineering, Utrecht, the Netherlands
roelof@cs.uu.nl
[2] Department of Computer Science, Queen Mary University of London, London,
United Kingdom
{gabs, mounia}@dcs.qmul.ac.uk

Abstract. In this article the activities of the INEX 2005 Multimedia track are reported. We succesfully realized our objective, to provide an evaluation platform for the evaluation of retrieval strategies for XML-based multimedia documents. In this first exploratory year the focus was on the retrieval of XML fragments, using both content-based text and image retrieval.

1 Challenge and Objectives

The main objective of the INEX 2005 multimedia track is to provide an evaluation platform for structured document retrieval systems that do not only include text in the retrieval process. Many structured document collections today also contain other types of media, such as images, speech, and video. To include these media types into the retrieval process and to produce a meaningful ranking is far from trivial. Using the structure of the document as a semantic/logical backbone for the retrieval of multimedia document fragments will allow us to investigate this problem from a new perspective. In the first year of the multimedia track, we provided an evaluation platform for the retrieval of multimedia structured document fragments, rather similar to the methodology used for the INEX Ad Hoc track.

The task set for the multimedia track was to retrieve relevant document fragments based on an information need with a structured multimedia character. A structured document retrieval approach in that case should be able to combine the relevances of the different media types into a single (meaningful) ranking that is presented to the user. The INEX multimedia track differs from other approaches in multimedia information retrieval, in the sense that it focuses on using the structure of the document to extract, relate, and combine the relevances of different multimedia fragments. The focus for 2005 was on the combination of text and image retrieval, using a strict interpretation of the structural components of the specified information need.

1.1 Track Outline

To give an impression of the activities deployed in the multimedia track a step-by-step outline is presented below.

- Acquisition of the Lonely Planet worldguide. One of the first steps was to acquire the a suitable XML collection that was easily accessible and contained well integrated multimedia objects. We acknowledge the Lonely Planet organization for providing us the WorldGuide XML collection.
- Extension of the NEXI query language. The original NEXI query language supported only text-based information retrieval. For the multimedia track, it is necessary to specify a similarity search on images, besides text. Therefore a small extension to the NEXI query language is defined.
- Baseline system for topic formulation. Both a text-based and an image-based retrieval system is provided to support the topic formulation process.
- Topic formulation procedure and selection. A topic formulation procedure is setup, similar is for the Ad-hoc track. Besides a search for relevant text fragments based on a particular information need the participants were asked to do a preliminary search for relevant images. This resulted in a pool of 25 topics, which is used for the retrieval performance experiment.
- Evaluation Methodology. We decided to keep the evaluation methodology simple, and use only binary relevancy judgments. This allowed us to adopt the TREC evaluation methodology.
- Assessment platform. The assessments are performed with the XRAI assessment tool provided by Dr. Benjamin Piwowarski of the University of Chile, Santiago. We are most grateful for his support. The XRAI tool is also used for the Ad-hoc track.
- Assessment procedure. Instead of the two-step assessment procedure, we only performed the first step, where relevant fragments of the document are highlighted, if the fragment satisfied all the requirements of the information need (SSCAS interpretation). In total eight participants took part in the assessment procedure. Two topics remained un-assessed.
- Evaluation of results. In total twenty-five runs were submitted by five participants (QMUL, QUTAU, RMIT, UTRECHT, and UTWENTE). Using the TREC evaluation tool, we reported the standard measures used in TREC.

1.2 Lonely Planet XML Document Collection

The corpus used for the INEX 2005 multimedia track is based on the Lonely Planet WorldGuide, made available by the Lonely Planet organization. The Lonely Planet collection consists of 462 XML documents with information about destinations, that is particularly useful for travellers that want to find interesting details for their next holiday or business trip. This particular collection is referred to as the ”WorldGuide” and can be viewed online at: `http://www.lonelyplanet.com/worldguide/`. The collection not only contains useful information about countries, but also includes information about interesting regions and major cities. For each destination an introduction is available, complemented with information about transport, culture, major events, facts, and an image gallery that gives you an impression of the local scenery.

2 Topic Formulation

This section start with some examples to introduce the approach followed for (1) the inclusion of content-based image retrieval into the specification of the information request in NEXI, and (2) the topic development procedure.

2.1 Examples

Within the multimedia track well focus on the content and structure topics, as these allow explicit formulation of the multimedia character in the information request, e.g. NEXI-CAS query. Consider for example the topic based on the Lonely planet collection below:

Example 1.

> **Information need**: *Find images depicting scuba diving activities for destinations with a tropical climate and that discuss exploring the beautiful underwater nature by diving activities.*
> **Information request**:

```
//destination[ about(.//weather,tropical climate)
     and about(.//activities, beautiful "underwater nature" diving)]
  //images//image[about(., scuba diving)]
```

The information need of Example 1 contains both textual and image elements. E.g. `about(.//weather,tropical climate)` specifies the condition requesting information about a tropical climate that is to be found within the XML element `weather` underneath a `destination` element. Furthermore, `image` elements are requested that depict scuba diving scenes.

Although the target elements of the above example are images, so far, simple textual retrieval approaches may be sufficient to produce the required output by searching image captions. However, a combination of text and image retrieval systems is encouraged within the track as these may in fact produce better results. Consider therefore Example 2.

Example 2.

> **Information need**: *Find images depicting scuba diving activities, like in BN5970 6.jpg, for destinations with a tropical climate and that discuss exploring the beautiful underwater nature by diving activities.*
> **Information request**:

```
//destination[about(.//weather,tropical climate)
     and about(.//activities, beautiful "underwater nature" diving)]
  //images//image[about(., scuba diving src:/image/BN5970_6.jpg)]
```

The extension to the information need of Example 2, where an example picture is specified, enforces the inclusion of content-based image retrieval techniques into the retrieval process. To specify the corresponding information request in NEXI, a small extension to the query language is needed (`src:/image/BN5970_6.JPG`).

2.2 Multimedia Extension to NEXI

By expressing both the content and image components of the information need within the same about clause, we are effectively overloading its meaning, leaving it to the retrieval system to decide if a text or image search (or both) is required. The reason for doing so is to emphasize the multimedia nature of the track. Using the extended about-clause, we can specify query constraints for a document fragment (which may be pure text, image, or a combination of multiple media) using a textual description (e.g. about(//image, scuba diving), about(//destination, scuba diving)) or using similar images (e.g. about(//image, src:/image/BN5970 6.jpg), about(//destination, src:/image/BN5970 6.jpg)) or any combination of the above, as used in Example 2. Various combinations of query conditions will require different strategies where text and image retrieval can be combined. The tracks focus is on the combination of the two techniques and therefore we encourage the submission of topics that force systems to implement content-based image retrieval (e.g. about(//destination, src:/image/BN5970 6.jpg)).

2.3 Topic Development Procedure

Below a desciption of the topic format is given, as used for the topic specification within the Multimedia track. Next, we will describe the topic development guidelines that where formulated to construct the topic pool.

Topic Format. The topic format for the multimedia track consists of the following fields: a description, castitle, and narrative. The following information should be contained in each of these fields:

- <**description**> A brief description of the information need, specifying any structural, textual, and visual requirements/composition on the content. The description must be precise, concise, and informative, but it must contain the same terms and the same structural requirements that appear in the castitle, albeit expressed in natural language.
- <**castitle**> A valid NEXI expression based on the Lonely Planet document collection that contains at least one about clause containing at least one image component. The expression is of the form //A[B] or //A[B]//C[D].
- <**narrative**> A detailed explanation of the information need and the description of what makes and element relevant or not. The narrative should explain not only what information is being sought, but also the context and motivation of the information need, i.e. why the information is being sought and what purpose it may serve. Assessments will be made on compliance to the narrative alone; it is therefore important that this description is clear and precise.

Topic Development Guidelines Each participating group was requested to submit 3 (CAS) topics. The additional constraints, as defined in the Topic development Guide for the INEX Ad Hoc track apply, for so far they are applicable and not overruled in the following steps:

Step 1: Initial topic statement. Create a one or two sentence description of the information you are seeking. This should be a simple description of the information need without regard to retrieval system capabilities or document collection peculiarities. Record the context and motivation of the information need, i.e. why the information is being sought.
Step 2: Exploration phase. In this step the initial topic statement is used to explore the collection. Obtain an estimate of the number of relevant elements, then evaluate whether this topic can be judged consistently. For this purpose two search engines were available, a text-based system and an image-based system, which could be used by the participants for topic development.
Step 2a: Assess the top 25 text fragments. Judge the relevancy of the retrieved text fragments (using binary relevance only). Each result should be judged on its own merits. Abandon a search, if there are fewer than 2 or more than 20 relevant text fragments in the result list.
Step 2b: Assess top 25 images. Since most participants did not have an off-the-shelves system available for the multimedia track, we have chosen to do a separate scan for the relevance of the image component. Therefore, the participants had to assess the top 25 images and judge their relevance (using binary relevance only). Each result should be judged on its own merits. Abandon the search, if there are fewer than 2 or more than 20 relevant images in the result list.
Step 2c: Inspect document matching. To assure that the document collection has a reasonable chance of completely fulfilling the text and image-based constraints of the information need a check at document level is needed. The participants were asked to count the number of documents that both satisfy the textual conditions and the image conditions. I.e. if a relevant text fragment was found, and there is an image that belongs to the same XML document, a match is found that corresponds to the information need. Abandon the topic if less than 3 matches are found over the top 25 results for both components.
Step 3: Write description, title, and narrative. During this step the participants completed the topic definition by writing the description, title, and narrative.
Step 4: Topic submission. Once finished, the topics were submitted using the on-line Candidate Topic Submission Form on the INEX website.

Topic Pool In total 8 participating groups submitted a total of 25 topics. In Example 3, one of the topics as submitted for the multimedia track is presented. It shows that any XML document fragment can be requested as the result of an information need, in this case the root element `destination` of the XML document is returned, while various conditions are formulated using both textual and visual requirements.

Example 3.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<inex_topic topic_id="2" inex_track="MM">
```

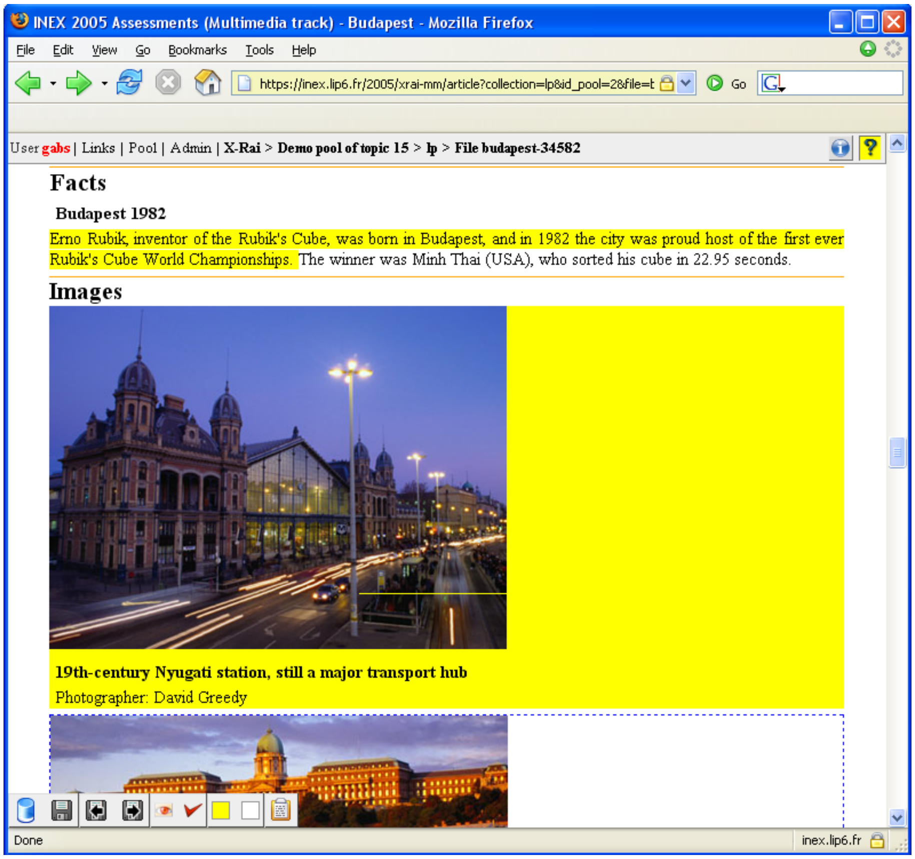
```
    <castitle>
        //destination[(about(., church) and about(., Europe)) or
            about(.//images//image, +church +Cathedral) or
            about(.//images//image, src:/images/BN6082_10.jpg)]
    </castitle>
    <description>
        I want information, text or images, about churches and church
        architecture in Europe.
    </description>
    <narrative>
        I'm planning a round trip in Europe with an aim to see as many
        interesting churches as possible. I want to have information
        about the locations, history and architecture of the churches.
        Also, I want to see pictures of the churches and cathedrals in
        Europe.
    </narrative>
</inex_topic>
```

3 Assessments

The definition of relevance used for the assessments is based on the definition employed in the INEX ad-hoc track with the exception that we measure exhaustivity only on a binary scale. In addition, reflecting the SSCAS task, we only consider an XML element relevant if it strictly matches the structural conditions specified within the query, i.e. only target elements may be relevant and only if they are contained in an XML document that satisfies the querys containment constraints.

Therefore, a given multimedia fragment is said to be relevant if it discusses (or depicts) the topic of request to any degree and if it strictly adheres to the structural conditions requested by the user. Similarly to the ad-hoc track, the assessment procedure follows the highlighting approach. However, given the binary nature of relevance, the assessment procedure for the multimedia track consists only of a single pass. During this single pass assessors were requested to highlight multimedia fragments that contain only relevant content, i.e. relevant content that contains no (or only minimal) non-relevant content. In the case of textual content, only relevant text fragments, e.g. words or sentences, should be highlighted. In the case of images, since currently it is not possible to highlight only a part of an image, the whole image should be highlighted if it contains relevant content (regardless of how much of the image may be non-relevant). The assessments are carried out with the XRAI assessment tool, developed by Dr. Benjamin Piwowarski of the University of Chile, Santiago. See Figure 1 for a snapshot of the interface, where fragments from the Lonely Planet collection are marked relevant (highlighted).

In total 8 participating groups took part in the assessment. As a result, 23 out of 25 topics have been assessed, providing us a solid basis for the evaluation in this first exploratory year of the multimedia track. Table 1 provides more


INEX 2005 Assessments (Multimedia track) - Budapest - Mozilla Firefox
File
Edit
View
Go
Bookmarks
Tools
Help
https://inex.lip6.fr/2005/xrai-mm/article?collection=lp&id_pool=2&file=t
Go
User gabs | Links | Pool | Admin | X-Rai > Demo pool of topic 15 > lp > File budapest-34582
Done
inex.lip6.fr

Facts

Budapest 1982

Erno Rubik, inventor of the Rubik's Cube, was born in Budapest, and in 1982 the city was proud host of the first ever Rubik's Cube World Championships. The winner was Minh Thai (USA), who sorted his cube in 22.95 seconds.

Images

19th-century Nyugati station, still a major transport hub

Photographer: David Greedy

Fig. 1. Snapshot of the XRAI interface.

detailed insight in the assessment results for the individual assessments. Topics *8*, *21*, *22*, and *24* are emphasized, because the level at which the assessment were performed, differentiated from the objectives in the castitle, and the topic description. These topics are removed from the official evaluation. However, after modifying the *castitles* of the topic description, these topics can be included for the 'extended' evaluation. This requires that the participants resubmit their runs, using the extended topic and assessment pools. An online evaluation tool is available for this purpose.

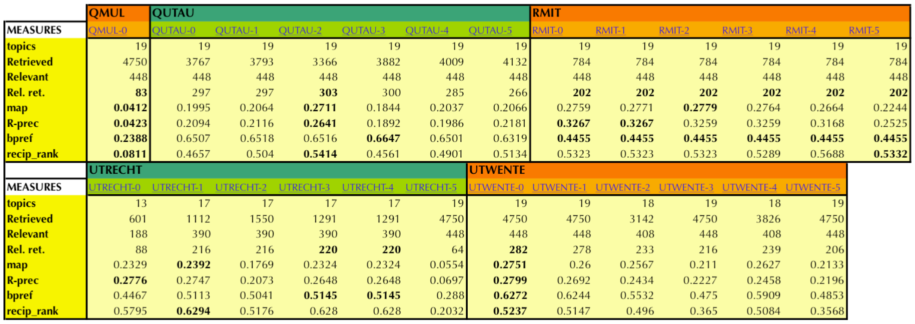
In this report we will only present the results for the official topic and assessment pools, based on the original submissions by the participants. The official evaluation is therefore based on a pool of 19 topics, whereas for the extended evaluation 23 topics are available.

Table 1. Details of the assessment per topic.

Topic	Relevant	Topic	Relevant
1	29	14	44
2	75	15	18
3	13	16	40
4	13	17	10
5	4	18	-
6	8	19	20
7	10	20	-
8	*5*	*21*	*25*
9	31	*22*	*21*
10	50	23	4
11	2	*24*	*77*
12	11	25	2
13	64		

4 Evaluation of Results

In this section, we will provide the summary table statistics, the interpolated recall-precision averages and the precision at document cutoff levels. These results are calculated using the TREC evaluation scripts, version 7.3.



	QMUL	QUTAU						RMIT					
MEASURES	QMUL-0	QUTAU-0	QUTAU-1	QUTAU-2	QUTAU-3	QUTAU-4	QUTAU-5	RMIT-0	RMIT-1	RMIT-2	RMIT-3	RMIT-4	RMIT-5
topics	19	19	19	19	19	19	19	19	19	19	19	19	19
Retrieved	4750	3767	3793	3366	3882	4009	4132	784	784	784	784	784	784
Relevant	448	448	448	448	448	448	448	448	448	448	448	448	448
Rel. ret.	**83**	297	297	**303**	300	285	266	**202**	**202**	**202**	**202**	**202**	**202**
map	**0.0412**	0.1995	0.2064	**0.2711**	0.1844	0.2037	0.2066	0.2759	0.2771	**0.2779**	0.2764	0.2664	0.2244
R-prec	**0.0423**	0.2094	0.2116	**0.2641**	0.1892	0.1986	0.2181	**0.3267**	**0.3267**	0.3259	0.3259	0.3168	0.2525
bpref	**0.2388**	0.6507	0.6518	0.6516	**0.6647**	0.6501	0.6319	**0.4455**	**0.4455**	**0.4455**	**0.4455**	**0.4455**	**0.4455**
recip_rank	**0.0811**	0.4657	0.504	**0.5414**	0.4561	0.4901	0.5134	0.5323	0.5323	0.5323	0.5289	0.5688	**0.5332**

	UTRECHT						UTWENTE					
MEASURES	UTRECHT-0	UTRECHT-1	UTRECHT-2	UTRECHT-3	UTRECHT-4	UTRECHT-5	UTWENTE-0	UTWENTE-1	UTWENTE-2	UTWENTE-3	UTWENTE-4	UTWENTE-5
topics	13	17	17	17	17	19	19	19	18	19	18	19
Retrieved	601	1112	1550	1291	1291	4750	4750	4750	3142	4750	3826	4750
Relevant	188	390	390	390	390	448	448	448	408	448	408	448
Rel. ret.	88	216	216	**220**	**220**	64	**282**	278	233	216	239	206
map	0.2329	**0.2392**	0.1769	0.2324	0.2324	0.0554	**0.2751**	0.26	0.2567	0.211	0.2627	0.2133
R-prec	**0.2776**	0.2747	0.2073	0.2648	0.2648	0.0697	**0.2799**	0.2692	0.2434	0.2227	0.2458	0.2196
bpref	0.4467	0.5113	0.5041	**0.5145**	**0.5145**	0.288	**0.6272**	0.6244	0.5532	0.475	0.5909	0.4853
recip_rank	0.5795	**0.6294**	0.5176	0.628	0.628	0.2032	**0.5237**	0.5147	0.496	0.365	0.5084	0.3568

Fig. 2. Interpolated precision-recall averages @ 11 standard recall levels.

5 Conclusions and Future Work

A detailed analysis of the results for the Multimedia track remains to be done. However, at this point we can conclude that despite the exploratory nature of the first year, many achievements have been realized. We have successfully acquired and exploited the Lonely Planet WorldGuide, which proved to be a useful starting point. With a minimum extension, we could successfully adopt the

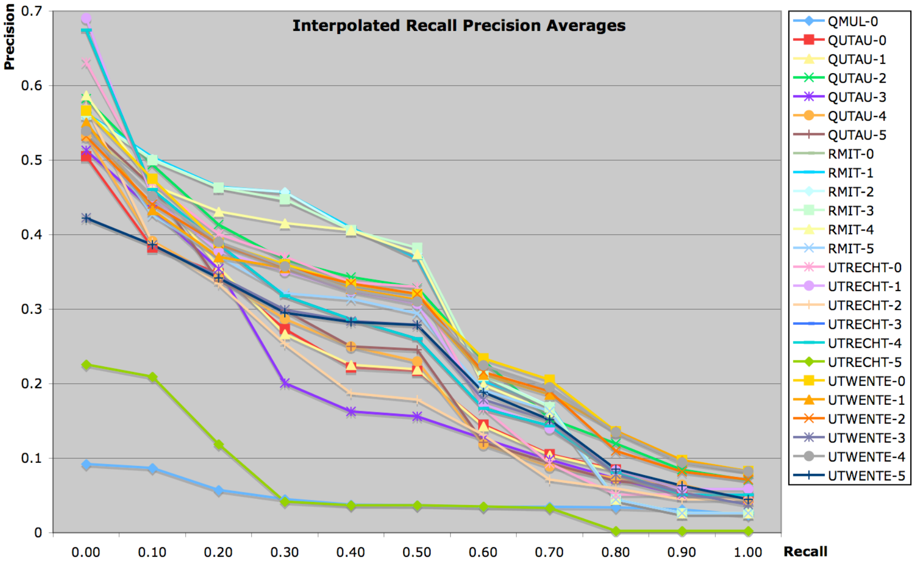


Fig. 3. Interpolated precision-recall averages @ 11 standard recall levels.

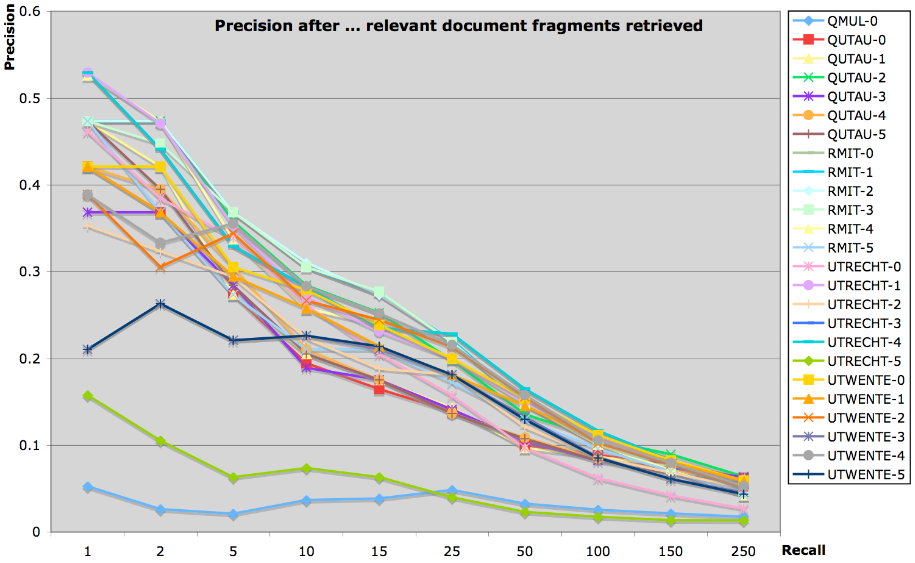


Fig. 4. Precision @ document cuttoff levels.

NEXI query language for *multimedia* structured document retrieval. A topic pool of 23 topics has been created and assessed. Five participating groups succeeded in building a multimedia retrieval system for structured documents and submitted a total of 25 runs for the evaluation.

A solid basis is created to run the multimedia track again next year. We will have to reconsider many of the choices made, such as for instance the topic formulation procedure and the evaluation metrics, which are currently based on the standard TREC methodology.

Acknowledgments

At this point we would like to thank all participants for the efforts to make the first year for the multimedia track a success. In particular we are grateful to the support we received from Dr. Benjamin Piwowarski, who helped us out with the relevance assessments. Finally, we would like to express our gratitude towards the Lonely Planet organization, for providing us with this interesting XML collection.

Integrating Text Retrieval and Image Retrieval in XML Document Searching

D. Tjondronegoro, J. Zhang, J. Gu*, A. Wardhani, S. Geva
Queensland University of Technology
2 George Street,GPO Box 2434, Brisbane, QLD 4001 Australia
{dian,jinglan.zhang, j2.gu, a.wardhani, s.geva}@qut.edu.au

Abstract: XML document format has been adopted as an industry standard by W3C. XML format contains the document content data and metadata. It marks up semantic document elements such as document, paragraph, images, maps etc. It contains structures and allows exploiting the internal structure in order to find keywords in certain document elements. The structure of XML format also supports more specific query and answers than text only documents. For example, a traditional document based query can be "return a document that contains an image like this one". An XML document can support the same query. However, it can also support other type of queries requesting more details, e.g. "return the paragraph that contains an image like this one" or "return the images that are similar to the sample image".

A picture is worth of one thousand words. Thus, many documents contain a mixture of text and images. For example, almost every webpage contains text and images so that they are more attractive and easier to understand than a page full of text. Images play an important role in webpage or article presentation. However, this information has not been explored sufficiently in traditional Information Retrieval systems. Many research issues are still open.

In information retrieval, usually the more information is used, the stricter the query is and the more precise the searching result should be. For example, "an excellent IT student" is stricter than "an IT student" which is again stricter that "a student". The text descriptions in an XML document, including the text content or the caption of the image, usually contain more precise information about an image. However, sometimes the keywords appeared in the text description are totally different from the content of an image. For example, the caption of an image may be "This place was a sea two million years ago" but the content of the image may be a mountain. We assume that if the image content is used in addition to the pure text-based retrieval, the retrieval result should be better than text-only or image-only retrieval. As an image specification in the query makes the query stricter than the query without the image, we assume image elements can be treated as if they were text elements containing ordinary keywords. This also implies that the query result should be more precise than the query result based on text only.

We test this hypothesis by doing a series of experiment using the Lonely Planet XML document collection provided by INEX2005 organizing committee. Two search engines, an XML document search engine using both content and structure and a content-based image search engine are used at the same time. Multiple test and image processing algorithms have been implemented. The results generated by these two search engines are merged together to form a new result. This paper presents our current work, initial results, some findings and vision into future work.

* supported by Jiangsu Government Scholarship of Overseas Studies.

Combining Image and Structured Text Retrieval

D.N.F. Awang Iskandar, Jovan Pehcevski, James A. Thom, and
S. M. M. Tahaghoghi

School of Computer Science and Information Technology
RMIT University, GPO Box 2476V
Melbourne 3001, Victoria, Australia
{dayang, jovanp, jat, saied}@cs.rmit.edu.au

Abstract. Two common approaches to retrieving images from a collection are retrieval by text keywords, and retrieval by visual content. However, it is widely recognised that it is impossible for keywords alone to fully describe visual content. In this paper we present our approach of combining evidence from a content-oriented XML retrieval system and a content-based image retrieval system using a linear evidence combination approach as part of the INEX 2005 Multimedia track.

1 Introduction

In large document collection it is common to find multimedia elements such as images, video, and sound. Presenting these multimedia elements in a standard way is beneficial as it might ease the retrieval process. The eXtensible Markup Language (XML) is a standard developed by the World Wide Web Consortium to describe data in a structured manner, allowing it to be stored and sent easily. Hence, description of multimedia elements can be represented in XML documents. The INitiative for the Evaluation of XML Retrieval (INEX) provides a platform for participants to evaluate the effectiveness of their XML retrieval techniques using uniform scoring procedures and a forum to compare results. INEX 2005 comprises seven tracks. The multimedia track was established with the aim of retrieving relevant XML document fragments containing various types of multimedia[1].

The RMIT group participated in the multimedia track with a fusion system that combines evidence and ranks the query results based on text and image similarity. Our motivation is to explore and analyse methods for combining evidence from content-based image retrieval (CBIR) with content-oriented XML retrieval. We explain and evaluate our approach in this paper.

The remainder of this paper is organised as follows. In Section 2 we explain the multimedia queries. We describe the approach which we use to retrieve the XML documents and the associated images based on the multimedia queries in Section 3. In Section 4, we present the analysis of our results. Related work to combination of evidence for retrieving image and text is briefly explained in Section 5. We conclude in Section 6 with a discussion of our findings and suggestions for future work.

[1] Multimedia Track @ INEX,
http://inex.is.informatik.uni-duisburg.de:2004/presentations/INEX-MM-track.pdf

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="mm6" inex_track="MM" query_type="CAS" ct_no="14">
<castitle>
//destination[about(., Europe) and about(.//culture//history, king queen)]
//images//image[about(., royal palace residence src:/images/BN7386_10.jpg)]
</castitle>
<description> From all European destinations that were ruled by either
a king or a queen in their cultural history, find images depicting a royal
palace residence. </description>
<narrative>We are a group of historians interested in royal palaces. We
want to visit destinations that contain at least one royal palace. We are
focused on European destinations that were ruled by either a king or a
queen in their cultural history. From these destinations, we want to find
images depicting a royal palace residence.</narrative>
</inex_topic>
```

Fig. 1. Example of a multimedia CAS query with one image as the source

2 Multimedia Queries

The multimedia retrieval task focuses on combination of text and images. In INEX 2005, the data collection was provided by Lonely Planet. As an initial task, INEX participants propose several topics which typically represent the information needed by the users from the collection. As an example, we proposed the topic “European destinations which have a palace” as shown in Figure 1.

Two types of queries are explored in INEX: content-only (CO) and content-and-structure (CAS). CO queries are free text queries. CAS queries contain explicit structural constraints. The INEX multimedia track uses the latter query type to represent the topic. The CAS query is contained in the `castitle` element.

The CAS query in Figure 1 can be interpreted as “find images depicting a royal palace residence from all European destinations that were ruled by either a king or a queen in their cultural history in the XML documents where the image is similar to BN7386_10.jpg”. The element to be retrieved as the answer for this topic is an image which is similar to BN7386_10.jpg.

There are 25 multimedia topics for the Lonely Planet collection. We observed that these topics belong to three topic categories. The first category includes topics that contain only text. The second category includes topics that contain a mixture of images and text, where the target element of each topic in this category is an image element. The third category is the same as the second, except that the target element of each topic is element other than image. The number of multimedia topics that belong to a category and a summary of the required results for each category is shown in Table 1.

Topic Category	Num. of Topics	Retrieval System Used	Required Query Result	Collection Involved
1	12	Content-oriented hybrid XML retrieval	Text only	XML document
2	8	Content-oriented hybrid XML retrieval and GIFT	Text and image. Image as target element	XML document and image
3	5	Content-oriented hybrid XML retrieval and GIFT	Text and image. Target element other than image	XML document and image

Table 1. Result category, retrieval system used, required query result and the collection involved.

3 Our Approach

In this experiment, we used two systems to obtain the results for the multimedia queries. Since the XML document structure serves as a semantic backbone for the retrieval of multimedia fragments, we used a content-oriented hybrid XML retrieval system [6] to retrieve the relevance documents. The GNU Image Finding Tool (GIFT)[2], a content-based image retrieval system is utilised to retrieve the results based on the visual features of a multimedia query source images.

According to Vogt et. al [9] "The chorus effect occurs when several retrieval approaches suggest that an item is relevant to a query ... this tends to be stronger evidence for relevance than a single approach doing so". In achieving the *Chorus Effect*, data fusion techniques have been employed to combine the evidence from GIFT and the content-oriented hybrid XML retrieval system. We base our retrieval process on the information retrieval data fusion process. The three following phases are involved [8]:

1. The *collection selection* phase focuses on identifying the document collection which is most likely to contain relevant documents for the user queries.
2. The *document selection* phase concerns on determining the number of relevant documents to be retrieved from the document collection.
3. The *merging* (or also known as *fusion*) phase deals with combining the evidence from multiple retrieval systems.

3.1 Phase One: Collection Selection

We view the Lonely Planet data as having three different collections which are correlated to one another. The first collection is XML documents, second is the image and third is the map. As illustrated in Table 1, the XML document collection is used to process all the queries. The image collection is used to process the queries for topic categories 2 and 3. The map collection was not used as the topic which needed the map as the target element was not assessed.

[2] http://www.gnu.org/software/gift/

3.2 Phase two: Document Selection

In this phase, each system retrieves the relevant documents or images and a list of normalised retrieval status values (RSVs) is returned as the result. We determined that the 250 top-rank documents with RSVs will be retrieved by the content-oriented hybrid XML retrieval system. Similarly with GIFT, the RSVs of all the images in the collection are returned as the result. The RSVs are ranked in descending order. The following sections explain how each system is used to generate RSVs for each multimedia query that is later used in the merging phase to produce the final run results.

Content-based Image Retrieval Content-based image retrieval aims to retrieve images on the basis of features automatically extracted from the images themselves. Prior to querying GIFT with an image, the image collection needs to be indexed. The indexing process involves image feature extraction and feature indexing using an inverted file data structure where the approach is inspired by text retrieval [7].

The colour and texture features of the images from the Lonely Planet collection are automatically extracted by GIFT. Both local and global colour features are used. As for extracting the image texture, GIFT utilises a bank of circularly symmetric Gabor filters.

GIFT's query engine was developed using Multimedia Retrieval Markup Language (MRML) [4] to evaluate and calculate the query image and the target image feature similarity based on the data from the inverted file. As a result, a ranked list of image answers based on the query image and the target image feature similarity is presented to the user. GIFT also provide the mechanism to perform relevance feedback through MRML. For this experiment, we did not perform any relevance feedback.

In retrieving the images for the multimedia topics, we presented the images which are listed in the `src` (source) element in the multimedia CAS query as the query image to GIFT. We implemented a function which enable all the images in the Lonely Planet collection to be retrieved and ranked. The default Classical IDF algorithm is used and we set the search pruning option to 100%. This allows us to perform a complete feature evaluation for the query image, even though the query processing time is longer.

Content-oriented Hybrid XML Retrieval The system we use in the INEX 2005 multimedia track follows a *hybrid* XML retrieval approach [6], combining information retrieval features from Zettair[3] (a full-text search engine) with XML-specific retrieval features from eXist[4] (a native XML database).

First, for each multimedia topic a translation module is used to automatically translate the underlying information need into a Zettair query. Terms that appear in the `castitle` part of the topic (with all structural query constraints and image references completely removed) are used to formulate the Zettair query. A list of (up to) 250 `destination` elements – presented in a descending order according to their estimated likelihood of relevance – is then returned as a resulting answer list for the INEX topic.

Second, to retrieve *elements* rather than full articles, a second topic translation module is used to formulate the eXist query. As the support elements and the target element

[3] `http://www.seg.rmit.edu.au/zettair/`

[4] `http://exist-db.org/`

of each INEX topic are strictly matched, both the terms and the structural query constraints from the topic (without the actual image references) are used to formulate the eXist query. We use the OR eXist query operator to generate the element answer list for a given topic. The answer list contains (up to) 250 matching elements, which are taken from articles that appear *highest* in the ranked list of articles previously returned by Zettair.

Last, a post-processing retrieval module that uses an XML-specific ranking heuristic (TPF) is used to rank and present the required answer elements [5].

3.3 Phase Three: Merging Evidence of CBIR and Keyword-based XML Retrieval

A simple linear evidence combination is applied to merge the RSVs from both systems. Using the following formula based on Aslandogan et. al [1], the RSVs are combined to generate a new score for the result rank list, *R*, of the multimedia queries.

$$R = \alpha \cdot S_I + (1 - \alpha) \cdot S_H$$

We let α be a parameter that influences the weighting scheme between the two retrieval systems (GIFT and hybrid XML retrieval), S_I represents the normalised RSV from GIFT and S_H is the normalised RSV from the hybrid XML retrieval system.

The rationale behind varying the weighting scheme is to investigate the effect of giving certain biases to a system. When the value of α is set to 1.0, only the RSVs from GIFT are used. On the other hand, only the hybrid XML retrieval RSVs will used when the value of α is set to 0.0. We performed six runs for each multimedia topic and the α value for each run is as illustrated in Table 2.

4 Result Analysis

This section provides an explanation of the results which are only based on 19 topics that belong to the official multimedia topic pool. We evaluated our results based on the standard recall and precision retrieval performance measures. The following measures are also used:

- Precision at cut-off (P@n): Precision after *n* document fragments retrieved. This shows precision at fixed points in the ranking.
- Mean average precision (MAP): Mean of the average precision. Average Precision is the average of the precision after each relevant document fragment is retrieved.
- R-precision: Precision value after the number of relevant topic document fragments retrieved.

The following discussion is based on Table 2. We observed the precision at cut-off 1, 5, 10 and 50. The highest precision after one document fragment retrieved is 0.5263 where the α value is 0.9, that is run RMIT-4. There is no visible change of precision values for all other runs at this cut-off value.

Run	α	P@n				MAP	R-Prec
		1	5	10	50		
RMIT-0	0.0	0.4737	**0.3684**	0.3053	**0.1484**	0.2759	**0.3267**
RMIT-1	0.1	0.4737	**0.3684**	0.3053	**0.1484**	0.2771	**0.3267**
RMIT-2	0.3	0.4737	**0.3684**	**0.3105**	**0.1484**	**0.2779**	0.3259
RMIT-3	0.5	0.4737	**0.3684**	0.3053	0.1474	0.2764	0.3259
RMIT-4	0.9	**0.5263**	0.3368	0.2579	0.1474	0.2664	0.3168
RMIT-5	1.0	0.4737	0.2737	0.2105	0.1295	0.2244	0.2525

Table 2. The alpha (α) values, P@C, MAP and R-Prec for each run.

Combining evidence from text and image at the same weight, where the value of α is 0.5 will lead to a precision value which is constantly similar for α values of 0.0, 0.1 and 0.3 after 5 document fragments retrieved. This can be seen for runs RMIT-0 to RMIT-3, which indicates that having the same amount of evidence from the CBIR system and the text retrieval system does not affect the precision performance. The precision values drop as the α value is increased.

The highest value when precision after 50 document fragments retrieved is observed for runs that have α values less than 0.3. This means that giving more weight to the evidence from the text retrieval results in better performance when retrieving a larger number of document fragments.

Run RMIT-2 with the α value of 0.3 gives the best performance for MAP. This indicates that run RMIT-2 returns more relevant documents earlier when retrieving the destination articles. As for the R-precision, run RMIT-0 and RMIT-1 has the same performance.

Based on Figure 2, runs for RMIT-0, RMIT-1 and RMIT-2 give the best overall interpolated recall precision averages performance. Run RMIT-4 performed best at low recall level. A constant performance can be seen for all the runs when the recall level is 0.8 and above.

5 Related Work

Data fusion, also known as combination of evidence, is a method of merging multiple source of evidence to form new evidence. In information retrieval, data fusion has been shown to improve the retrieval effectiveness when compared to using a single retrieval strategy [3, 8, 9]. According to Lee [3], "several researchers have investigated the effect of combining multiple representations of either queries or documents, or multiple retrieval techniques on retrieval performance because different representations or different retrieval techniques can retrieve different documents".

Multimedia retrieval using combination of evidence have been studied by Haque [2]. In his study, he compared the retrieval performance of using only image and multimedia (combination of text and image). He conducted experiments using three types of combining algorithms which are feature merging, weighted sum of ranking score and weighted sum of inverse rank position. Using combination of evidence, multimedia retrieval performs better than image retrieval. The weighted sum of inverse rank position

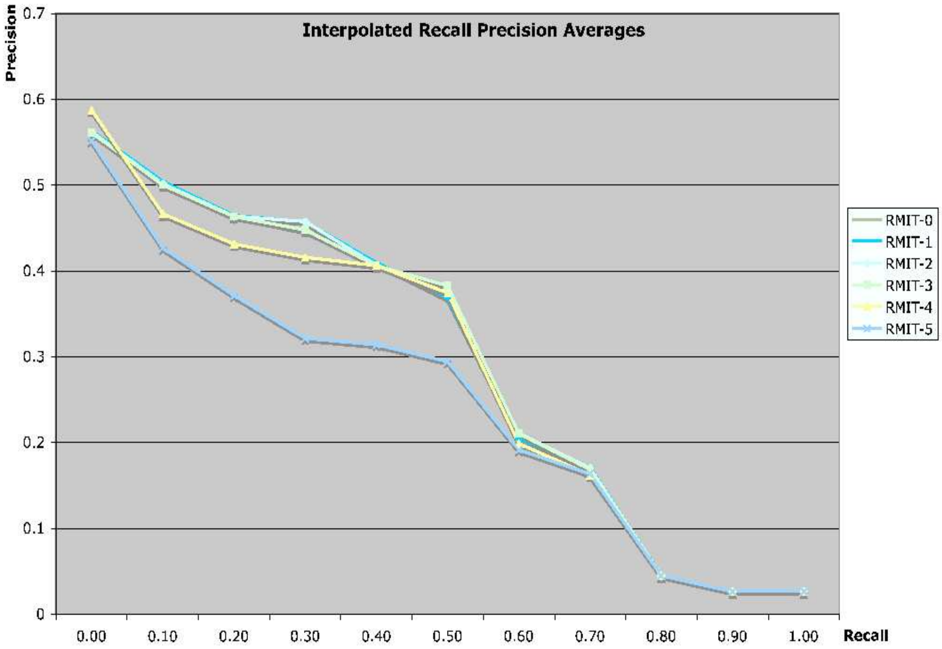


Fig. 2. Interpolated Recall Precision Averages for each run (best viewed in colour).

algorithm is shown to have the highest eleven point average precision in the multimedia retrieval. The weighted sum of ranking score algorithm performed slightly lower than the weighted sum of inverse rank position algorithm.

Aslandogan et .al [1] compared the retrieval performance of indexing person images on the web. They compared the retrieval performance on four approaches: text that is in HTML form evidence followed by face detection, face detection and recognition, linear evidence combination, and Dempster-Shafer evidence combination. The linear evidence combination and Dempster-Shafer evidence combination yield the same retrieval performance.

6 Conclusions and Future Works

We have experimented with the linear evidence combination approach in merging the normalised RSVs from two retrieval system for retrieving multimedia information from structured documents. We observed that to obtain best overall retrieval performance, an XML retrieval system needs to combine a fair amount of evidence from a CBIR system. On the other hand, to obtain best retrieval performance when ten or less elements are retrieved, a CBIR system needs to combine a little bit of evidence from an XML retrieval system.

In the future we plan to investigate different evidence combination methods.

Acknowledgements
This research was undertaken using facilities supported by the Australian Research Council and an RMIT VRII grant. We acknowledge Jonathan Yu for his assistance in proposing and assessing a multimedia topic.

References

1. Y. A. Aslandogan and C. T. Yu. Evaluating strategies and systems for content based indexing of person images on the Web. In *MULTIMEDIA '00: Proceedings of the eighth ACM international conference on Multimedia*, pages 313–321, New York, NY, USA, 2000. ACM Press.
2. N. Haque. *Image Ranking for Multimedia Retrieval*. Ph.D. thesis, School of Computer Science and Information Technology, Royal Melbourne Institute of Technology, 2003.
3. J. H. Lee. Analyses of Multiple Evidence Combination. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 267–276, New York, NY, USA, 1997. ACM Press.
4. W. Müller, H. Müller, S. Marchand-Maillet, T. Pun, D. M. Squire, Z. Pečenović, C. Giess and A. P. de Vries. MRML: A Communication Protocol for Content-Based Image Retrieval. In *Fourth International Conference On Visual Information Systems (VISual 2000)*, Lyon, France, November 2–4 2000.
5. J. Pehcevski, J. A. Thom and S. M. M. Tahaghoghi. RMIT University at INEX 2005. In *Pre-Proceedings of the Fourth INEX Workshop, Dagstuhl, Germany, November 28–30, 2005*, 2005.
6. J. Pehcevski, J. A. Thom and A-M. Vercoustre. Hybrid XML Retrieval: Combining Information Retrieval and a Native XML Database. *Information Retrieval*, Volume 8, Number 4, pages 571–600, 2005.
7. D. M. Squire, W. Müller, H. Müller and T. Pun. Content-based Query of Image Databases: Inspirations from Text Retrieval. *Pattern Recognition Letters*, Volume 21, Number 13–14, pages 1193–1198, 2000. (special edition for SCIA'99).
8. T. Tsikrika and M. Lalmas. Merging Techniques for Performing Data Fusion on the Web. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 127–134, New York, NY, USA, 2001. ACM Press.
9. C. C. Vogt and G. W. Cottrell. Predicting the Performance of Linearly Combined IR Systems. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 190–196, New York, NY, USA, 1998. ACM Press.

Multimedia Extensions for B^3-SDR, based on Principle Component Analysis

Roelof van Zwol

Utrecht University, Department of Computer Science, Center for Content and Knowledge Engineering, Utrecht, the Netherlands
roelof@cs.uu.nl

Abstract. The INEX 2005 multimedia track focusses on the retrieval of document fragments, containing both relevant text-fragments and images. In this short article we only shortly discuss the additional components that are added to the B^3-SDR system for the Multimedia track and the evaluation of the results.

1 Introduction

Structured document retrieval allows for the retrieval of document fragments, i.e. XML elements, containing relevant information. The main INEX adhoc task focusses on text-based XML element retrieval. Although text is dominantly present in most XML document collections, other types of media can also be found in those collections. Existing research on multimedia information retrieval has already shown that it is far from trivial to determine the combined relevance of a document that contains several multimedia objects. The objective here is to exploit the XML structure that provides a logical level at which multimedia objects are connected.

For the multimedia track, we started with a text-based system for structured document retrieval and a basic system for image retrieval. Using principle component analysis, we were able to merge the relevance scores calculated by the two systems, while using the underlying XML structure as a logical container. Principal component analysis (PCA) is a classical statistical method that has been widely used in data analysis and compression. In short, it provides us the means to transform an N-dimensional point set to representation in a N-1-dimensional space, while minimizing the error introduced due to the transposition. The integration of PCA for the multimedia track into our system only required transformations from a two-dimensional point set to a one-dimensional point set, i.e. text and images to a combined relevance score.

To evaluate the effectiveness of PCA to combine relevance rankings for text-fragments and images, we also submitted a text-only run, an annotation run and a combined-PCA-annotation run (mixed-cross). The annotation based run first obtains the captions that are available for the images that are specified in the information request, and then performs a text-based search. The mixed cross run first uses annotation-based enrichment of the text-based search, performs a image search, followed by a PCA merge of the relevance scores.

2 Results

Figure 1 shows the interpolated precision averages at eleven standard recall levels. From the figure, it can be clearly seen that using the image annotations (Utrecht-1) has a positive effect on the precision, when inspecting the higher recall levels. However our PCA-based runs (Utrecht-2 .. Utrecht-4) are not as successful as expected, when compared to the text-only run (Utrecht-0). Additional analysis is needed to investigate the cause, but it is likely that the poor performance of the image retrieval system is the main cause for failure.

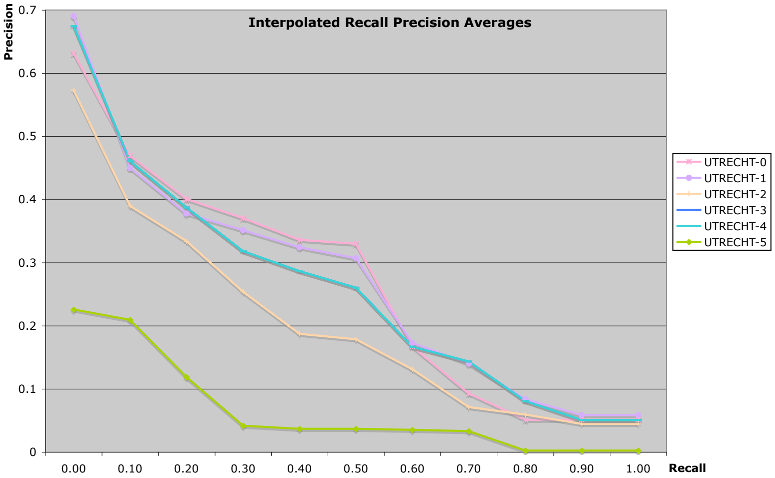


Fig. 1. Interpolated recall-precision averages.

3 Conclusions

[Due to the limited amount of time, only an extended abstract is provided for the pre-liminary proceedings]

INEX 2005 Guidelines for Topic Development

Börkur Sigurbjörnsson, Andrew Trotman, Shlomo Geva, Mounia Lalmas, Birger Larsen, and Saadia Malik

1 Aims

The aim of the INEX initiative is to provide the means, in the form of a large test collection and appropriate measures, for the evaluation of content-oriented XML element retrieval. Within the INEX initiative it is the task of the participating organizations to provide the topics and relevance assessments that will contribute to the test collection. Each participating organization, therefore, plays a vital role in this collaborative effort.

2 Introduction

Test collections, as traditionally used in information retrieval (IR), consist of three parts: a set of documents, a set of information needs called topics, and a set of relevance assessments listing (for each topic) the set of relevant documents.

A test collection for XML retrieval differs from traditional IR test collections in many respects. Although it still consists of the same three parts, the nature of these parts is fundamentally different. In IR test collections, documents are considered units of unstructured text, queries are generally treated as collections of terms and / or phrases, and relevance assessments provide judgments whether a document as a whole is relevant to a query or not. XML documents, on the other hand, organize their content into smaller, nested structural elements. Each of these elements in the document's hierarchy, along with the document itself (the root of the hierarchy), is a retrievable unit. In addition, with the use of XML query languages users of an XML retrieval system can express their information need as a combination of content and structural conditions: they can restrict their search to specific structural elements within the collection. Consequently the relevance assessments for an XML collection must also consider the structural nature of the documents and provide assessments at different levels of the document hierarchy.

This guide deals only with the topics of the test collection and provides detailed guidelines for their creation for INEX 2005.

3 Topic Creation Criteria

Creating a set of topics for a test collection requires a balance between competing interests. The performance of retrieval systems varies largely for different topics. This variation is usually greater than the performance variation of different retrieval methods on the same topic. Thus, to judge whether one retrieval strategy is (in general) more effective than another, the retrieval performance must be averaged over a large and diverse set of topics. In addition, to be a useful diagnostic tool, the average performance of the retrieval systems on the topics can be neither too good nor too bad as little can be learned about retrieval strategies if systems retrieve no, or only relevant, documents.

When creating topics, a number of factors should be taken into consideration. Topics should:

- be authored by an expert in (or someone familiar with) the subject areas covered by the collection,
- reflect real needs of operational systems,
- represent the type of service an operational system might provide,
- be diverse,
- differ in their coverage, e.g. broad or narrow topic queries,
- be assessed by the topic author.

3.1 Topic Types

A clear and precise description of the information need is required in order to unambiguously determine whether or not a given element fulfills the given need. In a test collection this description is known as the narrative. It is the only true and accurate interpretation of a user's needs. Precise recording of the narrative is important for scientific repeatability – there must exist, somewhere, a definitive description of what is and is not relevant to the user.

Many different queries could be drawn from the narrative, and some are be better than others. For example, some might contain phrases; some might contain ambiguous word; while some might even contain domain specific terms. At INEX a distinction is drawn between two types of query, those that have Content And Structure hints (CAS) and those that have Content Only (CO) hints.

Regardless of the query, the search engine results are not necessarily relevant. Even though a result might contain search terms from the query, it might not match the narrative description. Equally, some relevant documents might not be found, but they remain relevant because they are described as so by the narrative.

3.1.1 Content Only + Structure (CO+S)

The CO query simulates a user who does not know (or does not want to use) the actual structure of the XML documents in a query. This profile is likely to fit most users searching XML digital libraries.

Upon discovering their CO query returned many irrelevant hits, a user might decide to add structural hints. This is similar to a user adding + and – to a web query when too many irrelevant pages are found. At INEX, these added structural constraints (+S) are specified using the formal syntax called NEXI [1] (see the INEX 2005 website for the specification).

Example
Suppose a user wants to look at vitae of machine learning researchers. They enter the CO query:

```
vitae machine learning researcher
```

but discover that most results are about machine learning and are not vitae. They decide to add structural hints:

```
//vt[about(., machine learning researcher)]
```

restricting the results to `vt` elements only, which the user knows are vitae.

So the CO+S topic consists of three important and different versions of the information need. The narrative is the description of what is, and is not, a relevant result. The CO query is a query a user might enter into a retrieval engine. The +S query is a query the user might use when refining the query by adding structural hints, in order to express the information need more precisely.

The CO+S task is investigating relevance ranking algorithms for *ad hoc* element retrieval. This year the task is specifically investigating the usefulness of structural hints. AT INEX 2005 it will be possible to compare the performance (on the same topic) of using structural hints to that of not using structural hints.

3.1.2 Content And Structure (CAS)

A CAS query contains two kinds of structural hints: where to look (support elements), and what elements to return (target elements). In prior INEX workshops the target element hint has been interpreted either strictly or loosely (vaguely). Where to look has always been interpreted loosely.

These prior workshops have created considerable debate over how to interpret where to look. There is the database view: *all* structural constraints must be followed strictly (by exact match). Then there is the information retrieval view: an element is relevant if it satisfies the information need, irrespective of the structural constraints.

At INEX 2005 the multitude of loose and strict interpretations of topics will be compared side by side. The question being asked is this: are the structural constraints more valuable if followed exactly or are they better interpreted as structural hints?

In order to compare these interpretations it is necessary to make judgments for each *about* function of a query. Each CAS topic will therefore include a set of CAS sub-topics. These sub-topics are the constituent clauses of a topic that, when combined, constitute the actual topic.

Example
Suppose a user is interested in the work of Jiawei Han on the storage of XML documents in either a native XML, or relational, database. They might express this information need as the query:

```
//article[about(.//au, "Jiawei Han") and about(., xml storage)]//sec[about(., native
xml databases) or about(., xml relational databases)]
```

which has four about functions (sub-topics)

```
//article//au[about(.,"Jiawei Han")]
//article[about(., xml storage)]
//article//sec[about(., native xml databases)]
//article//sec[about(., xml relational databases)]
```

from this example it is clear that each *about* function can be written in the form

```
A[about(., B)]
```

where `B` is a content only query, and `A` is a navigational XPath expression constructed from the query such that in

```
P[about(Q,B)]R[about(S, C)]
```

`A` is `PQ` for `B`, and `PRS` for `C`; that is, the entire path is compounded into `A`.

The topic and each sub-topic will be human-assessed individually. From the assessments four sets of judgments will be automatically-derived (full details will be given at the assessment stage):

- VVCAS: the information retrieval assessments of the whole topic (done against the narrative).
- SVCAS: the subset of VVCAS assessments that strictly satisfy the target element constraint.
- VSCAS: those VVCAS assessments that satisfy all support element constraints, regardless of the target element constraint. Boolean operators between support elements are followed strictly.
- SSCAS: those assessments that strictly satisfy all support element constraints as well as the target element constraint.

Where, for XYCAS, X is the target element and Y is the support element, and either can be S for strict or V for vague.

By scoring each search engine against each of these interpretations it is possible to determine if the differences in the interpretations are reflected in performance. In other words, are the sets significantly different?

3.1.3 Natural Language Processing (NLP)

As an alternative to entering queries into search engines, a user might ask a librarian to find the information to satisfy their need. Such a user would give a verbal description to the librarian using a natural language. The NLP track at INEX is examining the ability of a search engine to satisfy the information need give this natural language description. Of course, NLP techniques are also used in other tracks, but using a query written in a formal query language.

Just as there are many CO queries derivable from the narrative, there are many ways to express the need in natural language. However it is expressed, it is important that it matches the narrative while at the same time it is not the narrative.

The purpose of the NLP experiments at INEX 2005 is to compare the performance of CO, CO+S and NLP techniques. To do this it is important that the same terms are used in each version of the query.

Example

Suppose a user wants to look at vitae of machine learning researchers and chose the CAS query:

```
//vt[about(., machine learning researcher)]
```

the NLP version would contain the same words and structural constraints:

```
Find me some vitae of machine learning researchers
```

which is a brief matter of fact description of the information need.

4 Topic Format

Topics are made up of several parts, these parts explain the ***same information need***, but for different purposes.

<narrative> A detailed explanation of the information need and the description of what makes an element relevant or not. The narrative should explain not only what information is being sought, but also the context and motivation of the information need, i.e., *why* the information is being sought and what work-task it might help to solve. Assessments will be made on compliance to the narrative alone; it is therefore important that this description is clear and precise.

<description> A brief description of the information need written in natural language - to be used in the NLP track. The description must be precise, concise, and as informative as the <title> and <castitle> combined. The topic description is discussed in more detail below.

<title> A short explanation of the information need. It serves as a summary of the content of the user's information need. The exact format of the topic title is discussed in more detail below. The title is present only in CO+S topics.

<castitle> A short explanation of the information need, specifying any structural requirements. The exact format of the topic title is discussed in more detail below. The castitle is optional in CO+S topics but mandatory in CAS topics.

<parent> Each CAS topic containing more than one *about* function will be submitted with a set of sub-topics describing the information need of each single *about* function. In order to match the sub-topics with the topic the parent must be identified in the sub-topic. For sub-topics, include here the castitle of the topic.

Note that the description must be interchangeable with the title and castitle. Any ambiguity or disagreement is resolved by reference to the narrative, the only accurate definition of the information need.

4.1 Topic <title>

To ensure topics are syntactically correct, a parser has been implemented in Flex and Bison (the GNU tools compatible with LEX and YACC) and is available for download or online use (see http://metis.otago.ac.nz/abin/nexi.cgi)

4.1.1 CO Topics

The topic title is a short representation of the information need. Each term is either a word or a phrase. Phrases are encapsulated in double quotes. Furthermore the terms can have either the prefix + or –, where + is used to emphasize an important concept, and – is used to denote an unwanted concept.

Example
A user wants to retrieve information about computer science degrees that are not master degrees:

```
"computer science" +degree -master
```

the + and – signs are used as hints to the search engine and do not have strict semantics. As an example the following text might be judged relevant to the information need, even though it contains the word master.

```
The university offers a program leading to a PhD degree in computer science.
Applicants must have a master degree…
```

Example
A user wants to retrieve information about IR from semi-structured documents:

```
"information retrieval" +semi-structured documents
```

As in the previous example the following text might be judged relevant, even though it neither contains the word semi-structured, nor the phrase "information retrieval".

```
The main goal of INEX is to promote the evaluation of content-oriented XML retrieval
by providing a large test collection of XML documents, uniform scoring procedures, and
a forum for organizations to compare their results…
```

Although the semantics of phrases and the + / – tokens is not strict, they may be of use to the retrieval engine. A full example of a CO topic is given in the appendix.

4.2 Topic <castitle>

Only a high level description is included here, for a more formal specification of the topic description language (NEXI) see the INEX web-site or in the proceedings of INEX 2004 [1].

To make sure that topics are syntactically correct, parsers have been implemented in Flex and Bison (the GNU tools compatible with LEX and YACC) and are available for download. An online version of the parser is also available: http://metis.otago.ac.nz/abin/nexi.cgi

The castitle is optional for CO+S topics but mandatory for CAS topics.

4.2.1 CO+S Topics

Castitles are XPath (http://www.w3c.org/TR/xpath) expressions of the form:

```
A[B]
```

or

```
A[B]C[D]
```

where `A` and `C` are navigational XPath expressions using only the descendant axis. `B` and `D` are predicates using *about* functions for text (explained below); the arithmetic operators `<`, `<=`, `>`, and `>=` for numbers; and the connectives `and` and `or`. The *about* function has the same syntax as the XPath function *contains*. Usage is restricted to the form:

```
about(.path, query)
```

where `path` is empty or contains only tag-names and descendant axis; and `query` is an IR query having the same syntax as the CO titles (i.e. query terms). The *about* function denotes that the content of the element located by the path is about the information need expressed in the query. As with the title, the castitle is only a hint to the search engine and does not have definite semantics.

Example
A user wants to know what the INEX participants said about native XML databases in 2003. The user assumes that the conference proceedings are mentioned somewhere in the front matter:

```
//article[.//fm//yr = 2003 and about(.//fm, INEX)]//*[about(., native XML databases)]
```

The user might be happy with retrieving something from the Proceedings of INEX 2003, although the proceedings were published in 2004. In the formalism expressed above,

```
A = //article
B = .//fm//yr = 2003 and about(.//fm, INEX)
C = //*
```

```
D = about(., native XML databases)
```

Example
Suppose a user wants find articles about flight simulators written in Java. The user thinks it is likely that such articles have an abstract that mentions flight simulators, and a paragraph talking about Java implementation. Thus they might write:

```
//article[about(.//abs, flight simulator) and about(.//p, java) and about(., flight
simulator java)]
```

When looking at the results the user is not likely to be picky whether the results fit the query exactly. They might, for example, be happy with articles which do not have an abstract so long as they are about Java implementation of flight simulators.

The main purpose of the INEX initiative is to build a test collection for the evaluation of content oriented XML retrieval. The most valuable part of the collection is the human made relevance assessments. Thus, each structure query **must** have at least one *about* function in the rightmost predicate.

An example CO+S topic is attached as an appendix.

4.2.2 CAS Topics

CAS topics contain a castitle written in the same XPath subset used in CO+S topics. They differ only in so far as every *about* function must also be submitted as a separate sub-topic. Each sub-topics will include a separate narrative, castitle and description – each explaining the purpose of the *about* function in the final whole topic. The sub-topics can be matched with the topic using the parent field of the sub-topic.

An example CAS topic and the constituent sub-topics are attached as an appendix.

4.2.3 Equivalent tags

In the current INEX IEEE collection there are several tags used interchangeably (for historical paper-publishing reasons). Tags belonging to the following groups are considered to be equivalent and can be used interchangeable in a query.

Paragraphs: ilrj, ip1, ip2, ip3, ip4, ip5, item-none, p, p1, p2, p3
Sections: sec, ss1, ss2, ss3
Lists: dl, l1, l2, l3, l4, l5, l6, l7, l8, l9, la, lb, lc, ld, le, list, numeric-list, numeric-rbrace, bullet-list
Headings: h, h1, h1a, h2, h2a, h3, h4

4.3 Topic <description>

The description should be precise and concise, but it must contain the same terms and the same structural requirements that appear in the <title> and the <castitle>, albeit expressed in natural language.

Example
A user wants to retrieve information about computer science degrees that are not master degrees and has chosen the title query:

```
"computer science" +degrees -master
```

for the title. From this they might choose a description of either:

```
retrieve information about degrees in computing science, but not masters degrees
```

or

```
I want descriptions of computer science degrees that are not master degrees
```

as they are equivalent, but:

```
get information about computing degrees, but not about master or PhD computing degrees
```

cannot be chosen as it expresses a different information need - there is an additional requirement that information about PhD degrees is not sought.

It is important to compare results that are based on natural language queries (the description) with queries that are based on the more formal languages (the title, and castitle.) The description must, therefore, be as informative as the title and castitle.

5 Procedure for Topic Development

Each participating group will have to submit *3 CO+S* and *1 CAS* topic (along with sub-topics) by the *6th May 2005*. Submission is done by filling in the Candidate Topic Submission Form on the INEX web site: http://inex.is.informatik.uni-duisburg.de/2005 under Tasks/Tracks → adhoc → Submit Topic.

The topic creation process is divided up into several steps. When developing a topic, use a print out of the online Candidate Topic Form to record all information about the topic you are creating.

***Step 1*: Initial Topic Statement**
Create a one or two sentence description of the information you are seeking. This should be a simple description of the information need without regard to retrieval system capabilities or document collection peculiarities. This should be recorded in the Initial Topic Statement field. Record also the context and motivation of the information need, i.e. *why* the information is being sought. Add to this a description of the *work-task*, that is, with what task it is to help (e.g. writing an essay on a given topic).

***Step 2*: Exploration Phase**
In this step the initial topic statement is used to explore the collection. Obtain an estimate of the number of relevant elements then evaluate whether this topic can be judged consistently. You may use any retrieval engine for this task, including your own or HyRex (HyRex can be accessed via the INEX website).

***Step 2a*: Assess Top 25 Results**
Judge the top 25 retrieval results. To assess the relevance of a retrieved element use the following working definition: *mark it relevant if it would be useful if you were writing a report on the subject of the topic, or if it contributes toward satisfying your information need*. Each result should be judged on it own merits. That is, information is still relevant even if it is the thirtieth time you have seen the same information. It is important that your judgment of relevance is consistent throughout this task. Using the Candidate Topic Submission Form record the number of found relevant elements and the path representing each relevant element. Then if there are:

- fewer than 2 or more than 20 relevant within the top 25, abandon the topic and use a new one,
- more than 2 and fewer than 20 relevant within the top 25, perform a feedback search (see below).

***Step 2b*: Feedback Search**
After assessing the top 25 element, you should have an idea of which terms (if any) could be added to the query to make the query as expressive as possible for the kind of elements you wish to retrieve.

Use the expanded query, to retrieve a new list of candidates. Judge the top 100 results (some are already judged), and record the number of relevant results in Candidate Topic Form. Record the expanded query in the title field of the Candidate Topic Submission Form.

***Step 3*: Write Narrative**
Having judged the top 100 results you should have a clear idea of what makes a component relevant or not. It is important to record this in minute detail as the narrative of the topic. The narrative is the definitive instruction used to determine relevance during the assessment phase (after runs have been submitted). Record not only what information is being sought, but also what makes it relevant or irrelevant. Also record the context and motivation of the information need. Include the work-task, that is: the form the information will take after having been found (e.g. written report). Make sure your description is exhaustive as there will be several months between topic development and topic assessment.

***Step 4 CO+S*: Optionally Write castitle**

Optionally re-write the title by adding structural constraints and target elements. Record this as the castitle on the Candidate Topic Submission Form. Also record why you think the structural hints might help in the narrative.

***Step 4 CAS*: Write Sub-Topics**

For CAS topics each *about* function is itself an information need that must be satisfied. Write the list of sub-topics and repeat steps 1 – 6 for each sub-topic. It is important that the topic is the sum of its sub-topics.

***Step 5*: Refining Topic Statements**

Finalize the topic title, castitle, description, and narrative. It is important that these parts all express the same information need; it should be possible to use each part of a topic in a stand-alone fashion (e.g. title for retrieval, description for filtering, etc.). In case of dispute, the narrative is the definitive definition of the information need – all assessments are made relative to the narrative and the narrative alone.

***Step 6*: Topic Submission**

Once you are finished, fill out and submit the on-line Candidate Topic Submission Form on the INEX website http://inex.is.informatik.uni-duisburg.de/2005/ under Tasks/Tracks→ Adhoc→ Topics→ Submit Topic. Make sure you submit all candidate topics no later than the *6th May 2005*.

6 Topic Selection

From the received candidate topics, the INEX organisers will decide which topics to include in the final set. This is done to ensure inclusion of a broad set of topics. The data obtained from the collection exploration phase *is* used as part of the topic selection process. The final set of topics will be distributed for use in retrieval and evaluation.

7 Acknowledgments

The topic format proposed in this document is based on the outcome of working groups set up during previous INEX workshops along with the online discussions they created. We are very grateful for this contribution. This document is a modified version of the topic development guides from previous INEX workshops.

References

[1] Trotman, A., & Sigurbjörnsson, B. (2004). Narrowed Extended XPath I (NEXI). In *Proceedings of the INEX 2004 Workshop*, (pp. 16-40).

Appendix 1: Example CO+S Topic

```
<inex_topic query_type="CO+S">
<title>formal logic reason UML diagrams</title>
<castitle>//article[about(.//bb, Rumbaugh Jacobson Booch) and about(.//abs, formal
methods)]//sec[about(., formal logic reason UML diagrams)]</castitle>
<description>I want to know about the application of formal methods and logics to
reason about UML diagrams. Relevant items probably cite Rumbaugh, Jacobson, or Booch.
</description>
<narrative>My main interest is the application of formal methods and logics in
software development. I choose to search for its application to UML diagrams because I
think it is an interesting application area. To be relevant, a document/component must
discuss the use of formal logics, such as first-order-, temporal-, or description-
logics, to model or reason about UML diagrams. I'm only interested in proper formal
logics, Business-logics and Client-logics do not have a proof system and are therefore
not considered to be formal logics. I think that sections are the most appropriate
unit of retrieval for this fairly specific topic, since I'm not really interested in
reading a lot about UML stuff in general. I want to focus in on the document parts
that talk about logic. I think it is useful for the search engine to look for citation
to the UML trio: Rumbaugh, Jacobson and Booch. Similarly think that it might be useful
to put the formal methods constraints on the abstract to stress that I'm only
interested in this particular subset of UML articles. Of course a relevant article
need not have this sort of reference or abstract, therefore the relevance of an
```

```
element will be judged on basis of how well it explains the use of formal logics to
model or reason about UML diagrams.</narrative>
</inex_topic>
```

Appendix 2: Example CAS Topic

The topic

```
<inex_topic topic query_type="CAS">
<castitle>//article[about(.//au,"Jiawei Han")]//abs[about(.,"data mining")]</castitle>
<description> a synopsis of data mining papers by Jiawei Han</description>
<narrative>I'm writing a short article about the impact of Jiawei Han on the field of
data mining. Therefore I'm interested in finding a short and concise overview of his
papers. I believe this is to be found in the abstracts of his papers. To be relevant,
the component has to be the abstract, written by Jiawei Han, about "data mining". Any
topics of data mining (e.g. association rules, data cube etc.) should be considered as
relevant.</narrative>
</inex_topic>
```

has two *about* functions (sub-topics) that must be satisfied

```
//article//au[about(.,"Jiawei Han")]
//article//abs[about(.,"data mining")]
```

each of which is submitted as a separate topic, with a parent element of

```
//article[about(.//au,"Jiawei Han")]//abs[about(.,"data mining")]
```

first,

```
//article//au[about(.,"Jiawei Han")]
```

```
<inex_topic topic query_type="CAS">
<castitle>//article//au[about(.,"Jiawei Han")]</castitle>
<parent>//article[about(.//au,"Jiawei Han")]//abs[about(.,"data mining")]</parent>
<description>Identify Jiawei Han as an author</description>
<narrative>I’m interested in works by Jiawei Han and only by Jiawei Han.  A relevant
result will contain his name as an author of a work.</narrative>
</inex_topic>
```

and second

```
<inex_topic topic query_type="CAS">
<castitle>//article//abs[about(.,"data mining")]</castitle>
<parent>//article[about(.//au,"Jiawei Han")]//abs[about(.,"data mining")]</parent>
<description>Give a synopsis of what’s happening in data mining</description>
<narrative> To be relevant, the component has to be an abstract about "data mining".
Any topics of data mining(e.g. association rules, data cube etc.) should be considered
as relevant.</narrative>
</inex_topic>
```

so this CAS topic is submitted using three Candidate Topic Submission Forms.

Appendix 3: Another Example CAS Topic

INEX topic 155 would be submitted as a whole

```
<inex_topic type="CAS">
<castitle>//article[about(.//p,"self organising feature map") and
about(.//fm//yr,2000)]//fig[about(.//fgc,"self organizing map")]</castitle>
<description>Return figures having a figure caption about the self organising map from
articles dating to about 2000 and with paragraphs about the self organising feature
map.</description>
<narrative>Looking for figures depicting the architecture or configuration of the self
organizing feature map. The figures should be from articles having elements that
discuss the map. Relevant material should be from around 2000 give or take a couple of
years. The SOM method is described or its use is an application described.</narrative>
</inex_topic>
```

and as parts for

```
//article//p[about(., "self organising feature map")]
//article//fm//yr [about(.,2000)]
//article//fig//fgc[about(.,"self organizing map")]
```

first,

```
<inex_topic type="CAS">
<castitle>//article//p[about(., "self organising feature map")]</castitle>
<parent>//article[about(.//p,"self organising feature map") and
about(.//fm//yr,2000)]//fig[about(.//fgc,"self organizing map")]</parent>
<description>paragraphs about the self organising feature maps</description>
<narrative>Looking for articles having elements that discuss the self organizing maps.
And discussion of SOM makes the result relevant.  A discussion of Kohonen’s other work
is not relevant.</narrative>
</inex_topic>
```

second,

```
<inex_topic type="CAS">
<castitle>//article//fm//yr [about(.,2000)]</castitle>
<parent>//article[about(.//p,"self organising feature map") and
about(.//fm//yr,2000)]//fig[about(.//fgc,"self organizing map")]</parent>
<description>looking for papers published during or about the year 2000</description>
<narrative>Relevant material should be from around 2000 give or take a couple of years.
Anything published between 1998 and 2002 is relevant, anything outside that range is
not of interest.</narrative>
</inex_topic>
```

and third

```
<inex_topic type="CAS">
<castitle>//article//fig//fgc[about(.,"self organizing map")]</castitle>
<parent>//article[about(.//p,"self organising feature map") and
about(.//fm//yr,2000)]//fig[about(.//fgc,"self organizing map")]</parent>
<description>Return figure captions about the self organising maps</description>
<narrative>Looking for figures depicting the architecture or configuration of the self
organizing feature map.  Table captions are relevant, but an abstract would be to big
and is therefore not relevant.</narrative>
</inex_topic>
```

INEX 2005 Retrieval Task and Result Submission Specification

Mounia Lalmas

Monday, June 20, 2005

Retrieval Task

The retrieval task to be performed by the participating groups of INEX 2005 is defined as the ad-hoc retrieval of XML elements. In information retrieval (IR) literature, ad-hoc retrieval is described as a simulation of how a library might be used, and it involves the searching of a static set of documents using a new set of topics. While the principle is the same, the difference for INEX is that the library consists of XML documents, the queries may contain both content and structural conditions and, in response to a query, arbitrary XML elements may be retrieved from the library. Within the ad-hoc retrieval task we define the following three sub-tasks:

CO sub-task: content-oriented XML retrieval using *content-only* conditions.

Queries with content-only conditions (CO queries) are requests that ignore the document structure and contain only content related conditions, e.g. only specify what an element should be about without specifying what that component is. The need for this type of query for the evaluation of XML retrieval stems from the fact that users may not care about the structure of the result components or may not be familiar with the exact structure of the XML documents. **The <*title*> part of the CO+S topics should be used as queries for the CO sub-task**.

The aim of an XML retrieval system is to find relevant elements for a given topic of request, where relevance in XML retrieval has two dimensions: *exhaustivity* and *specificity*. An element is exhaustive if the topic of request is exhaustively discussed within that element, whereas an element is specific if the element is highly focussed on the topic. The general aim of an XML retrieval system is to find the elements that are most specific and most exhaustive with respect to the topic of request.

XML retrieval systems may employ various retrieval strategies. These strategies can be viewed as how we, as system developers, assume a user will want the output of a search to be. In INEX 2005, three strategies have been defined for the CO sub-task.

- ***Focussed retrieval strategy:*** The aim of the focussed retrieval strategy is to find **the** most exhaustive and specific element in a **path**. In the case where an XML retrieval system has estimated a parent and one if its children elements to be equally exhaustive and specific for a given topic, the parent element should be returned. In addition, when a parent has been estimated as more exhaustive than one of its children elements, but that child element has been estimated as more specific than its parent, then the child element should be returned. In this way, preference for specificity over exhaustivity is given. This strategy means that a retrieval run (i.e. the retrieved elements) cannot contain any overlapping elements. This strategy is intended for approaches that are concerned with a very focussed retrieval of XML elements, i.e. aiming at targeting the appropriate level of granularity for a given topic. A CO sub-task that uses the focussed retrieval strategy is referred to as **CO.Focussed**.

- ***Thorough retrieval strategy:*** The aim of the thorough retrieval strategy is to find **all** highly exhaustive and specific elements. It will be therefore the case that, due to the nature of relevance in XML retrieval (e.g. if a child element is relevant, so will be its parent, although to a greater or lesser extent), an XML retrieval system that has estimated an element to be relevant may decide to return all its ancestor elements. This means that runs for this task may contain a large number of overlapping elements. It is however a challenge to rank these elements appropriately, as systems that rank highly exhaustive and specific elements before less exhaustive and specific ones, will obtain a higher effectiveness performance. This strategy is intended for XML retrieval approaches that do not deal with the overlapping issue from a system perspective, but that they consider it as an interface and results presentation issue. It is however still crucial that "highly" relevant elements are ranked first. A CO sub-task that uses the thorough retrieval strategy is referred to as **CO.Thorough**.

- ***Fetch and browse retrieval strategy:*** The aim of the fetch and browse retrieval strategy is to **first** identify relevant articles (the fetching phase), and **then** to identify the most exhaustive and specific elements within the fetched articles (the browsing phase). In the fetching phase, articles should be ranked according to how exhaustive and specific they are (i.e. the most exhaustive and specific articles should be ranked first). In the browsing phase, ranking should be done according to how exhaustive and specific an element in an article is when compared to other elements in the same article. This strategy is intended for XML retrieval approaches that are based on a mixture of document retrieval and element retrieval strategies. A CO sub-task that uses the fetch and browse retrieval strategy is referred to as **CO.FetchBrowse**.

+S sub-task: content-oriented XML retrieval using *additional* structural hints.

Upon discovering that a CO query returned many irrelevant hits, a user may decide to add structural hints. These structural hints are expressed in the **<castitle> part of the CO+S topics, which should be used as the query** for the +S sub-task**.** The aim of the +S sub-task is to specifically investigate the usefulness of the structural hints. The performance on the same topic when using the structural hints will be compared to that when not using structural hints (i.e. the runs for the CO sub-task).

As for the CO sub-task, there are three retrieval strategies, which are defined in exactly the same way as for the CO sub-task. A +S sub-task that uses the focussed retrieval strategy, the thorough retrieval strategy, and the fetch and browse retrieval strategy is referred to as, respectively, **+S.Focussed**, **+S.Thorough**, and **+S.FetchBrowse**.

(It should be noted that the relevance of the elements will be assessed using the <narrative> part of the CO+S topics. Runs from the CO sub-task and the +S sub-task will be merged to create the assessment pool.)

CAS sub-task: content-oriented XML retrieval based on *content-and-structure* (CAS) queries.

CAS queries are topic statements that contain explicit references to the XML structure, and explicitly specify the contexts of the user's interest (e.g. target elements) and/or the context of certain search concepts (e.g. containment conditions). More precisely, a CAS query contains two kinds of structural constraints: where to look (i.e. the *support* elements), and what to return (i.e. the *target* elements). A structural constraint can been interpreted as either strict or vague. With a strict interpretation, the structural constraints must be followed strictly, i.e. by exact match. With a vague interpretation, an element is relevant if it satisfies the information need, irrespective of the structural constraints, which are considered as structural hints. The aim of the CAS sub-task is to investigate whether the structural constraints are more valuable if they are followed strictly, or as structural hints. **The <*castitle*> part of the CAS topics should be used as queries for the CAS sub-task**.

As structural constraints can apply to support and target elements, we define four strategies for the CAS sub-task, where for XYCAS, X is the target element and Y is the support element, and either can be S for strict and V for vague:

- ***VVCAS strategy:*** where the structural constraints in both the target elements and the support elements are interpreted as vague.

- ***SVCAS strategy:*** where the structural constraints in the target elements are interpreted as strict and the structural constraints in the support elements are interpreted as vague.

- ***VSCAS strategy:*** where the structural constraints in the target elements are interpreted as vague and the structural constraints in the support elements are interpreted as strict.

- ***SSCAS strategy:*** where the structural constraints in both the target elements and the support elements are interpreted as strict.

In the above four strategies, the aim is to retrieve the most exhaustive and specific elements with respect to the topic of request. By evaluating each XML retrieval approach against each of these interpretations, it will be possible to determine if the difference in interpretations has an effect on performance.

The guidelines for the NLP retrieval task will be given separately.

Result Submission

For the CO and +S sub-tasks and their corresponding strategy, e.g. CO.Focussed, CO.Thorough, … +S. Thorough, +S.FetchBrowse, up to 3 runs may be submitted. The results of one run must be contained in one submission file (i.e. up to 18 files can be submitted in total). A submission may contain up to **1500** retrieval results for each of the INEX topics included within that sub-task (e.g. for the +S.Focussed strategy of the +S sub-task only submit the search results obtained using the <*castitle*> part of the CO+S topics).

For the Fetch and Browse strategy, for both the CO and +S sub-tasks, the retrieved elements (obtained through the browse phase) of the top ranked article (obtained through the fetch base) should be ranked first, then the retrieved elements of the second ranked article should be ranked, etc. The ranking of the fetched articles will be derived automatically.

It should be noted that although the motivation for the CO+S is to specifically investigate the usefulness of the structural hints, the INEX organisers will only report effectiveness performance of each individual run. Pairwise comparison between e.g. a CO.Focussed run and its respective +S.Focussed run will have to carried out by the participants themselves, to be reported in the INEX 2005 workshop. In addition, for participants that are only interested in the CO sub-task, there are no requirements to submit runs for the +S sub-task, and vice versa.

Please note that not all CO+S topics have a <*title*> and <*castitle*> parts. This means that there were no additional structural constraints, so the +S and CO runs are the same. For pairwise (CO vs. +S) comparison purpose, these topics should be ignored.

For the CAS sub-task, a retrieval run is composed of the elements that have been estimated as relevant to the CAS topic. For the CAS sub-task and their corresponding strategy, e.g. VVCAS, VSCAS, up to 2 runs may be submitted. At least 1 run must be submitted for the VVCAS strategy. A submission may contain up to **1500** retrieval results for each of the CAS topics. Although performance will be calculated on the parent-topics, participants are asked to submit runs for all CAS topics (including the sub-topics). The results from the sub-topics can be thought of as partial "evidence" used by the retrieval approach to help rank results for the parent-topic. ***Note that the structural constraints for the sub-topics should be interpreted as vague***.

Submission format

For relevance assessments and the evaluation of the results we require submission files to be in the format described in this section. The submission format for the CO, +S, and CAS sub-tasks is defined in the following DTD:

```
<!ELEMENT inex-submission    (description, collections, topic+)>
<!ATTLIST inex-submission
   participant-id     CDATA    #REQUIRED
   run-id             CDATA    #REQUIRED
   task    ( CO.Focussed | CO.Thorough | CO.FetchBrowse |
             +S.Focussed | +S.Thorough | +S.FetchBrowse |
             VVCAS | VSCAS | SVCAS | SSCAS)    #REQUIRED
   query   (automatic | manual)    #REQUIRED
>
<!ELEMENT description    (#PCDATA)>
```

```
   <!ELEMENT topic     (result*)>
   <!ATTLIST topic
      topic-id    CDATA    #REQUIRED
   >

   <!ELEMENT collections (collection+)>
   <!ELEMENT collection (#PCDATA)>

   <!ELEMENT result    (in?,file, path, rank?, rsv?)>
   <!ELEMENT in      (#PCDATA)>
   <!ELEMENT file    (#PCDATA)>
   <!ELEMENT path    (#PCDATA)>
   <!ELEMENT rank    (#PCDATA)>
   <!ELEMENT rsv     (#PCDATA)>
```

Each submission must contain the participant ID of the submitting institute (available at http://inex.is.informatik.uni-duisburg.de/2005/inex05/ShowParticipants05.jsp), a run ID (which must be unique for the submissions sent from one organisation – also please use meaningful names as much as possible), the identification of the task (e.g. CO.Focussed, +S.FetchBrowse, etc), and the identification of whether the query was constructed automatically or manually from the topic. Please note that **at least one of the runs for each sub-task must be with the use of automatic queries**. Furthermore each submitted run must contain a description of the retrieval approach applied to generate the search results. A submission contains a number of topics, each identified by its topic ID (as provided with the topics).

For compatibility with the heterogeneous collection track, the <*collections*> tag is mandatory. There should be with <*collections*> at least one <collection> tag, which is by default set to “ieee” for the ad hoc track. The <*in*> tag is optional for the ad hoc track (<*in*> states from which collection each result comes from).

For each topic a maximum of **1500** result elements may be included per sub-task (i.e. CO.Focussed, CO.Thorough, …, +S.Thorough, +S.FetchBrowse, VVCAS, …, SSCAS). A result element is described by a file name and an element path, and it may include rank and/or retrieval status value (rsv) information. For the ad hoc retrieval task, `<collection>` is set to “ieee”. Here is a sample submission file for the focussed retrieval strategy of the CO sub-task:

```
<inex-submission participant-id="12" run-id="VSM_Aggr_06" task=”CO.Focussed”
query=”automatic”>
    <description>Using VSM to compute RSV at leaf level combined with
          aggregation at retrieval time, assuming independence and using
          augmentationweight=0.6.
    </description>
    <collections>
        <collection>ieee</collection>
    </collections>
    <topic topic-id="01">
        <result>
             <file>tc/2001/t0111</file>
             <path>/article[1]/bm[1]/ack[1]</path>
             <rsv>0.67</rsv>
        </result>
        <result>
             <file>an/1995/a1004</file>
             <path>/article[1]/bdy[1]/sec[1]/p[3]</path>
             <rsv>0.1</rsv>
        </result>
        [ ... ]
    </topic>
    <topic topic-id="02">
        [ ... ]
    </topic>
    [ ... ]
</inex-submission>
```

Regarding the Fetch and Browse retrieval strategy, the most relevant elements of the most relevant article should be ranked first, then the most relevant elements from the second most relevant article should then be ranked, etc. An example is given below.

```
<topic topic-id="01">
       <result>
              <file>tc/2001/t0111</file>
              <path>/article[1]/bm[1]/ack[1]</path>
              <rank>1</rank>
       </result>
       <result>
              <file>tc/2001/t0111</file>
              <path>/article[1]/sec[1]/</path>
              <rank>2</rank>
       </result>
       [ ... ]
       <result>
              <file>xg/2005/t0135</file>
              <path>/article[1]/bm[1]/ack[1]</path>
              <rank>3</rank>
       </result>
       <result>
              <file>xg/2005/t0135</file>
              <path>/article[1]/sec[1]/</path>
              <rank>4</rank>
       </result>
       [ ... ]
</topic>
```

An article element can be returned only if the article itself has been estimated as the most relevant element (with respect to the article). Finally note that a submission run ha a maximum of 1500 elements, thus one challenge in the Fetch and Browse retrieval strategy is to determine the number of relevant elements to return for each fetched (relevant) article.

Rank and RSV

The rank and rsv elements are provided for submissions based on a retrieval approach producing ranked output. The ranking of the result elements can be described in terms of:

- Rank values, which are consecutive natural numbers, starting with 1. Note that there can be more than one element per rank.
- Retrieval status values (RSVs), which are positive real numbers. Note that there may be several elements having the same RSV value.

Either of these methods may be used to describe the ranking within a submission. If both rank and rsv are given, the rank value is used for evaluation. These elements may be omitted from a submission if a retrieval approach does not produce ranked output.

File and path

Since XML retrieval approaches may return arbitrary XML nodes from the documents of the INEX collection, we need a way to identify these nodes without ambiguity. Within INEX submissions, elements are identified by means of a file name and an element (node) path specification, which must be given in XPath syntax.

File names must be given relative to the INEX collection's `xml` directory (excluding the `xml` directory from the file path). The file path should use '/' for separating directories. Note that <u>only article files</u> (e.g. no volume.xml files) can be referenced here. The extension `.xml` must be left out. Example:

```
an/1995/a1004
```

Element paths are given in XPath syntax. To be more precise, only fully specified paths are allowed, as described by the following grammar:

```
Path           ::=  '/' ElementNode Path | '/' ElementNode '/' AttributeNode | '/' ElementNode
ElementNode    ::=  ElementName Index
AttributeNode  ::=  '@' AttributeName
Index          ::=  '[' integer ']'
```

Example:

```
/article[1]/bdy[1]/sec[4]/p[3]
```

This path identifies the element which can be found if we start at the document root, select the first “article” element, then within that, select the first “bdy” element, within which we select the fourth “sec” element, and finally within that element we select the third “p” element.

Important: XPath counts elements starting with 1 and takes into account the element type, e.g. if a section had a title and two paragraphs then their paths would be given as: `../title[1]`, `../p[1]` and `../p[2]`.

A result element may then be identified unambiguously using the combination of its file name and element path. Example:

```
<result>
    <in>ieee</in>
    <file>an/1995/a1004</file>
    <path>/article[1]/bdy[1]/sec[1]/p[3]</path>
</result>
```

An application that can be used to check the correctness of a given path specification is available at http://inex.is.informatik.uni-duisburg.de/2005/browse.html. Note that this application requires the input of a file name and element path. If these are correctly given, the specified XML element within its container article element will be displayed.

Result Submission Procedure

To submit a run, please use the following link:

http://inex.is.informatik.uni-duisburg.de/2005/

Then go to Tasks/Tracks->Adhoc->Submissions. The online submission tool will be available soon.

Acknowledgments

The sub-tasks and the strategies are based on the outcome of the INEX 2004 workshop. These were also further extensively discussed and debated on the INEX 2005 organisers list, and the INEX 2005 discussion list. These discussions and debates are greatly appreciated and acknowledged. Finally, special thanks go to Shlomo Geva, Gabriella Kazai, Yosi Mass, Tassos Tombros and Andrew Trotman for their comments and clarifications on this document.

INEX 2005 Relevance Assessment Guide

1. Introduction

During the retrieval runs, participating organisations evaluated the 87 INEX 2005 topics (40 content + structure (CO+S) and 47 content-and-structure (CAS) queries) against the IEEE Computer Society document collection and produced a list (or set) of document components (XML elements[1]) as their retrieval results for each topic. The top 1500 components in a topic's retrieval results were then submitted to INEX. The submissions received from the different participating groups have now been pooled and redistributed to the participating groups (to the topic authors whenever possible) for relevance assessment. Note that the assessment of a given topic should not be regarded as a group task, but should be provided by one person only (e.g. by the topic author or the assigned assessor).

The aim of this guide is to outline the process of providing relevance assessments for the INEX 2005 test collection. This requires first a definition of relevance (Section 2), followed by details of what (Sections 3) and how (Section 4) to assess. Finally, we describe the on-line relevance assessment system that should be used to record your assessments (Section 5).

2. Relevance in INEX

Relevance in INEX is defined according to the following two dimensions:

- **Exhaustivity (E)**, which describes the extent to which the document component discusses the topic of request.
- **Specificity (S)**, which describes the extent to which the document component focuses on the topic of request.

In order to decrease assessment effort, a highlighting procedure is used in INEX 2005, leading to the following process for assessment (more details in Sections 4 and 5):

• In the first pass, assessors highlight text fragments that contain only relevant information

• In the second pass, assessors judge the exhaustivity level of any elements that have highlighted parts.

As a result of this process, any elements that have been fully highlighted will be automatically labelled as fully specific. The main advantage of this highlighting approach is that assessors will now only have to judge the exhaustivity level of the elements that have highlighted parts (in the second phase). The specificity of any other (partially highlighted) elements will be calculated automatically as some function of the contained relevant and irrelevant content (e.g. in the simplest case as the ratio of relevant content to all content, measured in number of words or characters).

An exhaustivity level is therefore requested for all document components that contain some relevant information, and can be any of the following values:

🟩🟩 **Highly exhaustive (HE):** the component discusses most or all aspects of the topic of request. In the relevance assessment system, E2 is represented as two green squares.

🟩⬜ **Partially exhaustive (PE)**: the component discusses only few aspects of the topic of request. In the relevance assessment system, E1 is represented as two squares, one green and the other white.

▭ **Too small (TS):** the component contains some relevant material, but the relevant fragment is too small to be assessed. In the relevance assessment system, TS is a pale green rectangle.

Within the relevance assessment system, a component – that contains some relevant information and has not yet been assessed has an unknown exhaustivity. The corresponding icon is 🟩🟥 (a green and a red squares). All other elements will be – automatically - assumed as **Not Exhaustive (NE)**.

[1] The terms document component and XML element are used interchangeably.

3. What to judge

Depending on the topic, a pooled result set may contain initially around 500 articles.

Traditionally, in evaluation initiatives for information retrieval, like TREC or CLEF, relevance is judged on document level, which is treated as the atomic unit of retrieval. In XML retrieval, the retrieval results may contain document components of varying granularity, e.g. paragraphs, sections, articles, etc. Therefore, to provide comprehensive relevance assessment for an XML test collection, **it is necessary to obtain assessment for all components at the different levels of granularity that contain any relevant information.**

This means that if you find, say, a section of an article relevant to the topic of the request (i.e. containing highlighted text), you will then need to provide assessment with regards to exhaustivity for the found relevant component, for all its ascendant elements until you reach the article component (unless this can be automatically inferred, e.g. the parent of a highly exhaustive (HE) element will be itself highly exhaustive (HE)), and for all its descendant elements that contain relevant information (i.e. containing highlighted text) until you have identified all relevant sub-components.

Such comprehensive assessments are necessary as it is demonstrated by the following example. Consider the XML structure in Figure 1. Let us say that you judged the marked sec element relevant to the topic, as partially exhaustive (PE, denoted by 🟩□, see Section 2). Given this single assessment, it would not be possible to deduce the exhaustivity level of the ascending or descending elements. For example, both bdy and article may be judged either highly (HE) or partially exhaustive (PE) depending on the volume of additional relevant information contained within the other sections and in the fm and bm components. Looking at the sub-components of our sec element, it is clear that no conclusions can be drawn from the assessment score assigned to our sec element regarding the exhaustivity level of its sub-components; for example, one of the paragraphs of the second ss2 element may be too small (TS) while the other may be partially exhaustive (PE).

```
[article]
 [fm]
  ...
 [bdy]
  [🟩□sec]
   [ss1]
    [ip1]
    [ss2]
     [p]
    [p]
    [ss2 ]
     [ip1]
     [p]
     [lc]
      [li]
       [p]
       [p]
      [li]
       [p]
   [ss1]
   [ss1]
  [sec]
    ...
 [bm]
  ...
```

Figure 1. Example XML structure

As a general rule, it can be said that the exhaustivity level of a parent element is always equal to or greater than the exhaustivity level of its children elements. This is due to the cumulative nature of exhaustiveness. For example, the parent of a highly exhaustive (HE) element will always be highly exhaustive (HE), since the child element already discusses all or most aspects of the topic. However, besides this general rule, no specific rules exist that would automate all the exhaustivity assessment of ascendant and descendant elements of relevant components. Therefore, **you will need to explicitly judge the exhaustivity level of all elements that contain relevant information**. This is the only way to ensure both comprehensive and consistent relevance assessments.

4. How to judge

As described in Section 2, the assessment process is to be done in two phases.

- In the first pass, assessors highlight text fragments that contain only relevant information. A vital consideration is that the highlighting must be based solely on the specificity dimension (e.g. ignoring exhaustivity in the first phase). Assessors should be made aware not to highlight larger contexts because these are more exhaustive, if at the same time they are less specific (i.e. contain irrelevant fragments). It is important that only purely relevant information fragments get highlighted. To decide which text to highlight, you should skim-read the whole article (that a result element is a part of - even if the result element itself is not relevant!) and identify any relevant information as you go along. The on-line system can assist you in this task by highlighting keywords (that are chosen using the interface) and pool elements (elements retrieved by participating systems) within the article (see Section 5).
- In the second phase, you should assess the exhaustivity of the components that intersect with any of the highlighted passages (i.e. identified in the first phase). The on-line assessment system (see Section 5) will identify for you all elements that have to be assessed for phases 2.

During the relevance assessment of a given topic, all parts of the topic specification should be consulted in the following order of priority: narrative, topic description, and topic title. The narrative should be treated **as the most authoritative description of the user's information need**, and hence it serves as the main point of reference against which relevance should be assessed. In case there is conflicting information between the narrative and other parts of a topic, the information contained in the narrative is decisive. Note that it is not because that a term listed within the topic is not present in an element that the element is not relevant. It may be that a component contains some or maybe all the terms, but is irrelevant to the topic of the request. Also, there may be components that contain none of the terms yet are relevant to the topic.

For both the CO+S and CAS topics, the topic titles (may) contain structural constraints in the form of XPath expressions. These structural conditions should be ignored during your assessment. This means that you should assess the elements returned for a CO+S and CAS topic as whether they satisfy your information need (as specified by the topic) **with respect to the content criterion only**.

Note that some result elements may be related to each other (ascendant/descendant), e.g. an article and some sections or paragraphs within the article. This should not influence your assessment. For example if the pooled result contains Chapter 1 and then Section 1.3, you should not assume that Section 1.3 is more relevant than Sections 1.1, 1.2, and 1.4, or that Chapter 1 is more relevant than Section 1.3 or vice versa. Remember that the pooled results are the product of different retrieval engines, which warrants no assumptions about the level of relevance based on the number of retrieved related components!

You should judge each document component on its own merits! That is, a document component is still relevant even if it the twentieth you have seen with the same information! It is imperative that you maintain consistency in your judgement during assessment. Referring to the topic text from time to time will help you maintain judgement consistency.

5. Using the on-line assessment system (X-Rai)

There is an on-line relevance assessment system (XML Retrieval Assessment Interface) provided at:

https://inex.lip6.fr/2005/xrai

which allows you to view the pooled result set of the topics assigned to you for assessment, to browse the IEEE-CS document collection and to record your assessments. Use your INEX username and password to access this system.

The assessment tool works with opera and recent "gecko" browsers: we highly recommend you to use Opera (version 8 or up only) available at http://www.opera.com. Other compatible browsers are:

- **Mozilla** (version 1.7 or up) at http://www.mozilla.org/products/firefox/.
- **Firefox** (version 1 and up) at http://www.mozilla.org/products/mozilla1.x/.

Note that **JavaScript must be enabled** for the assessment tool to work and that **the assessment tool is not compatible with Internet Explorer. Any bug report should be submitted using the project homepage (https://developer.berlios.de/projects/x-rai/) using the link in the "Links" menu of the interface (Figure 2).**

5.1. Home page

After logging in, you will be presented with the Home page (see Figure 2) listing the topic ID numbers of the topics assigned to you for assessment (under the title "Choose a pool"). This page can always be reached by clicking on the **"X-Rai"** link of the menu bar on any subsequent pages.

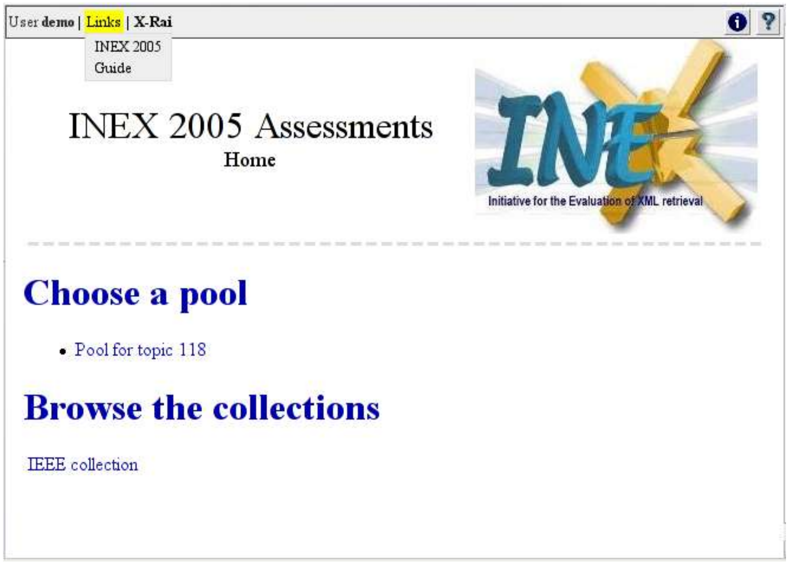


Figure 2: Home page and menu bar

In the "Links" menu

- **INEX 2005**: link to the official INEX web site.
- **X-Rai project:** link to the development web site of X-Rai where you can submit bug reports or/and feature requests.
- **Guide**: the latest version of this assessment guide.

Each X-Rai page is composed of the following components:

- The menu bar, which is itself composed of four parts:
  1. The login name (e.g. "demo" in Figure 2),
  2. A list of menu items, which can be accessed by holding the mouse over the menu label (e.g. "**Links**" in Figure 2.),
  3. The location within X-Rai, where each location step is a hyperlink (in Figure 2, we are at the root of the web site, so the only component of the location is "**X-Rai**", which is a link to the home page),
  4. The menu bar may also contain a number of icons (displayed on the right hand side, see Figure 3a). Click on one of these icons to display (or hide):

     ⓘ Information about X-Rai.

     ? Toggle the help
- The main window.
- An optional status bar (see Figure 5), displayed only when assessing a pool, i.e. in pool, sub-collection or article view (see relevant sections below) appears at the bottom of the window and shows the number of unknown assessments you have to judge before completing assessing the document (in Figure 5, there is only one unknown assessment).

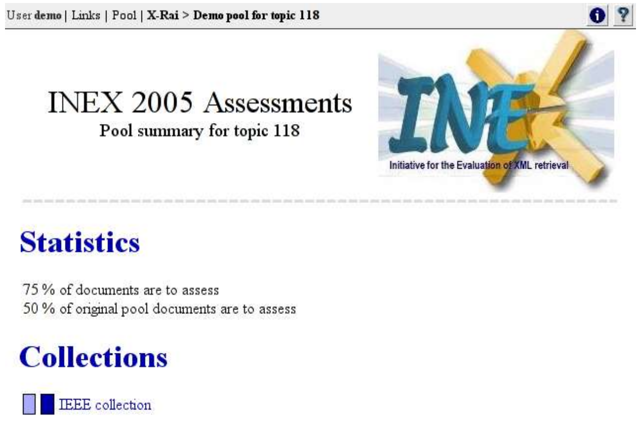
- In the status bar, three arrows (⇦, ⇧ and ⇨) may be used to navigate quickly between the elements to be assessed. You may also use the shortcut keys of 1 (left), 2 (up) and 3 (right). The up arrow enables you to move to a level up in the hierarchy, e.g. from an article or a collection part to its innermost enclosing part of the collection (you move in the opposite direction by selecting a sub-collection or an article). The left arrow can be used to go to the previous element to be assessed, while the right arrow to go to the next element to be assessed.

The on-line assessment system provides three main views (Sections 5.2 to 5.4):

1. Pool view,
2. Sub-collection view, and
3. Article view

5.2. Pool view

Clicking on a topic ID will display the Pool main page for that topic (see Figure 3a).



User demo | Links | Pool | X-Rai > Demo pool for topic 118

INEX 2005 Assessments

Pool summary for topic 118

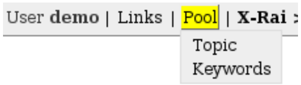
Statistics

75 % of documents are to assess
50 % of original pool documents are to assess

Collections

IEEE collection

Figure 3a. Pool view



User demo | Links | Pool | X-Rai :
Topic
Keywords

Figure 3b. Pool menu:

Topic: displays topic statement.
Keywords: to manipulate list of words and phrases to highlight.

Here, a new menu item, “**Pool**”, appears on the menu bar at the top of the window.

Within the “Pool” menu (Figure 3b), with the “**Topic**” submenu item you can display the topic statement in a popup window. This is useful as it allows you to refer to the topic text at any time during your assessment.

The “**Keywords**” submenu item allows you to access a feature, where you can specify a list of words or phrases to be highlighted when viewing the contents of an article during assessment. These cue words or phrases can help you in locating potentially relevant texts within an article and may aid you in speeding up your assessment (so add as many relevant cue words as you can think of!). You may edit, add to or delete from your list of keywords at any time during your assessment (remember, however, to refresh the currently assessed article to reflect the changes). You may also specify the preferred highlighting colour for each and every keyword. After selecting the “**Keywords**” menu item, a popup window will appear showing a table of coloured cells. A border surrounding a cell signifies a colour that is already used for highlighting some keywords. Move the mouse over a coloured cell to display the list of keywords that will be highlighted in that colour. To edit the list of words or phrases for a given colour, click on the cell of your choice. You will be prompted to enter a list of words or phrases (one per line) to highlight. You can choose three different highlighting modes using the drop-down menu: using coloured fonts, drawing a border around the phrase or using a background colour. Note that the words or phrases you specify will be matched against the text in the assessed documents in their exact form, *i.e.* no stemming is performed.

Under the title “Collections” is the list of collections to be assessed. In INEX 2005 (ad hoc task) there is only one such collection, the IEEE collection.

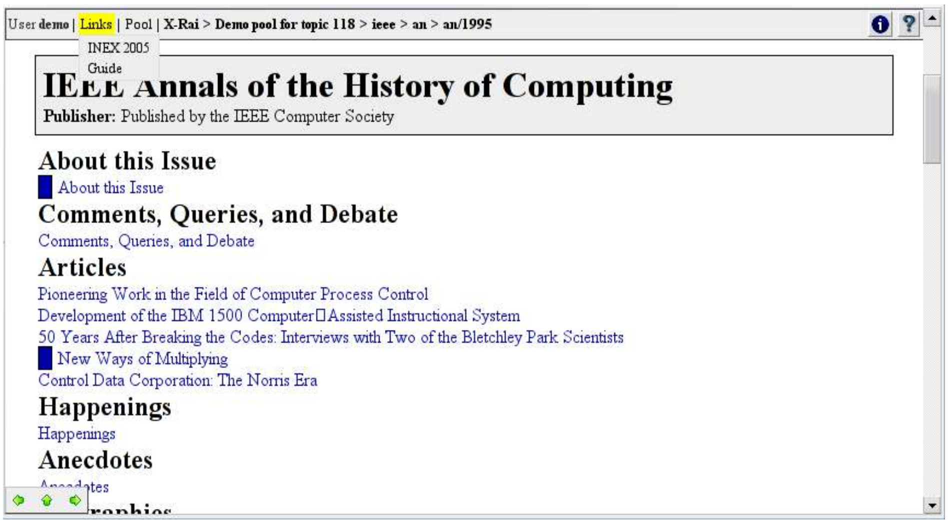
The left or right arrows on the status bar move the focus to the previous or next collection, where there is at least one element to assess (since there is only one collection, so no change will occur).

Clicking the hyperlink of "IEEE collections" will take you into the sub-collection view.

5.3. Sub-collection view

The sub-collection views allow you to browse the different sub-collections within the IEEE collection, i.e. volumes, years within a given volume (see Figure 4), the collection of articles within a given volume and year. Note that this view will show all elements within a sub-collection, i.e. all articles within a given volume and year, and not only the ones that need to be assessed. For each possible sub-collection, there is an indication on the number of documents to be assessed in it (if this number is greater than 0), both for documents that were initially in the pool and for documents you choose to assess.

The left or right arrows on the status bar move the focus to the previous or next sub-collection, where there is at least one document to assess. You can also directly click on a link to a sub-collection.



User **demo** | Links | Pool | **X-Rai** > **Demo pool for topic 118** > **ieee** > **an** > **an/1995**

IEEE Annals of the History of Computing

Publisher: Published by the IEEE Computer Society

About this Issue

About this Issue

Comments, Queries, and Debate

Comments, Queries, and Debate

Articles

Pioneering Work in the Field of Computer Process Control
Development of the IBM 1500 Computer□Assisted Instructional System
50 Years After Breaking the Codes: Interviews with Two of the Bletchley Park Scientists
New Ways of Multiplying
Control Data Corporation: The Norris Era

Happenings

Happenings

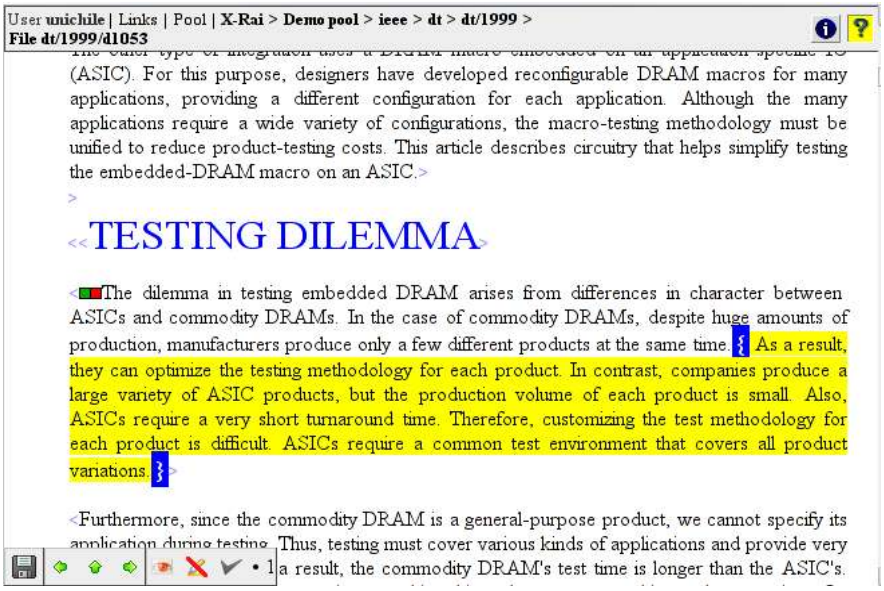
Anecdotes

Anecdotes

Figure 4. Sub-collection view

5.4. Article view

It is in this article view that elements can be assessed. The article view (see Figure 5) displays all the XML elements of an article together with their content. There are two types of objects within an article view: XML elements and passages. The latter are defined by the assessor while highlighting whereas the former are predefined by the XML file. XML elements boundaries are denoted by < and > (less/greater in pale blue). { A passage in the interface has a yellow background and is enclosed within two braces on a blue background like this sentence }. For each element, the assessment value is displayed at its start, that is after the "<" for an XML element.



User unichile | Links | Pool | X-Rai > Demo pool > ieee > dt > dt/1999 >
File dt/1999/d1053

(ASIC). For this purpose, designers have developed reconfigurable DRAM macros for many applications, providing a different configuration for each application. Although the many applications require a wide variety of configurations, the macro-testing methodology must be unified to reduce product-testing costs. This article describes circuitry that helps simplify testing the embedded-DRAM macro on an ASIC.>

>

<<TESTING DILEMMA>

<The dilemma in testing embedded DRAM arises from differences in character between ASICs and commodity DRAMs. In the case of commodity DRAMs, despite huge amounts of production, manufacturers produce only a few different products at the same time. As a result, they can optimize the testing methodology for each product. In contrast, companies produce a large variety of ASIC products, but the production volume of each product is small. Also, ASICs require a very short turnaround time. Therefore, customizing the test methodology for each product is difficult. ASICs require a common test environment that covers all product variations.>

<Furthermore, since the commodity DRAM is a general-purpose product, we cannot specify its application during testing. Thus, testing must cover various kinds of applications and provide very
• 1 a result, the commodity DRAM's test time is longer than the ASIC's.

Figure 5. Article view

Highlighting

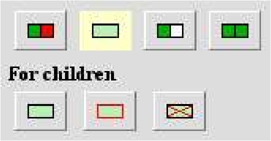
You are in the highlighting mode when the marker icon in the status bar is ✎. During the highlight phase, you should identify only relevant (i.e. totally specific) passages by highlighting them. Passages can span over XML element boundaries. The passage limits are predefined by a pre-processing of XML files and correspond “more or less” to sentence boundaries. A consequence of this is that you should highlight the smallest passage that encloses the only relevant information if the predefined boundaries do not correspond exactly to the totally specific fragment.

To highlight a passage, select it with the mouse as you would do in any word processor or text editor, and click on the square with the yellow background (or press “h”).

If you make an error, you can unhighlight it by selecting the non relevant passage and clicking on the square with the white background (or press “u”).

Assessing

Once you have finished to highlight relevant passages, you may switch to the assessing mode by clicking on the yellow marker. You can also switch by pressing the “m” (for **m**ode) key. Once you have switched, it is not possible to (un)highlight passages any more and the icon in the status bar should look like a crossed marker (✗). To assess an element, simply click on the assessment icon. An assessment panel will then appear:



The first line shows the list of possible assessments for the current assessed element. The second line contains shortcuts to set or remove “too small” judgements for some or all of the children - and, as a consequence, to all the descendants: the first button assesses **all the non judged** children as “too small”; the second one assesses **all** children as “too small”, and the third one resets the assessments to “unknown” for children judged “too small”. The process is recursive: if a value is changed for a child, then the same operation is applied for its own children. For example, judging a child as “too small” will judge the child children as TS, etc.

You should always give the exhaustivity value of the **strictly** contained data, that is without taking into account the component context (i.e. parent or sibling component). Any XML element that contains at least a passage cannot be judged as “too small”. The interface will prevent you from doing so (in the above Figure, the “too small” assessment is disabled).

In order to assess an XML element, you can have a quick look at its boundaries by putting the mouse pointer over an element assessment: the content of the XML element will be bordered by a red line. You can also change the background colour of an XML element by clicking on its assessment while pressing the shift key. You can switch back to the normal display by clicking again on its assessment while pressing the shift key. This is useful when assessing big elements (like a section, etc.) so you can inspect their full content before judging them.

Judging an element (or its descendants) “too small”

If the assessed XML element intersects with one or more passages but does not contain any one passage completely, it is possible to assess it as “too small”. This will also automatically assess its descendants as TS. When the element - denoted X in the following - is not too small itself but its descendants (or a part of them) are, then it is possible to judge all of them as “too small”. The procedure to follow is:

1. assess explicitly all the descendants which are not “too small”

2. click on the X assessment icon, and press on the icon “assess the remaining children as too small” (the icon 🟩). Note that the interface will automatically judge as TS any descendant of a too small element.

It is also possible to judge all the descendants as “too small”, overriding their values by clicking on the TS icon with a red border (🟩). If you made a mistake and want to reset the assessments of the descendants, you can also remove all the TS judgements of the descendants by clicking on the crossed pale rectangle icon (☒).

Assessment consistency

Contrarily to last years, there is no pre-checking on the allowed exhaustivity values. If there is a conflict after the user assessed an element, then the conflicting assessment value(s) is/are reset to "unknown" so the judge has to reassess it: for example, let a section be assessed as HE and its only paragraph as HE. If the user changes the paragraph assessment to PE or unknown/too small, then the section assessment is set to unknown.

Another example of consistency check is when you change an assessment from “too small” to another value: the descendants, which were previously assessed as “too small”, will be reset to “unknown”.

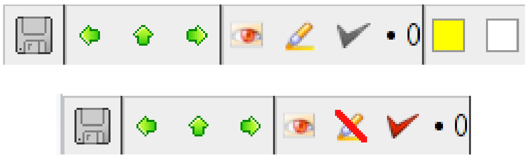


Figure 6. Status bar (article view only): highlighting mode (top) and assessing mode (bottom)

The disk icon (here disabled): saving your assessments

The left/right arrows: going to the previous/next element to judge
The up arrow: going to the sub-collection view that contains the article

The eye: shows or hides the pool elements

The yellow marker: switch between the highlighting and the assessing view.

The mark reflects the status of the document: completely assessed and validated (green), completely assessed but not validated (red), and not completely assessed and not validated (grey). You can validate a document (i.e., mark it as finished) only if the mark is red.

The number indicates the number of elements to be assessed.

The yellow/white square (when in highlighting mode) permits to (un) highlight the selected passage.

5.6. Saving your assessments

!

The assessment tool this year does not automatically save the assessments, but you NEED TO SAVE YOUR RELEVANCE ASSESSMENTS by clicking on the disk icon:

💾

The icon is disabled (grey shade) when all assessments are saved.

Be warned that Opera doesn't provide a way to prevent from exiting a page without saving assessments. PLEASE ONLY USE THE INTERFACE TO NAVIGATE INTO THE SITE as this is the only way to prevent you from leaving a page with non-saved assessment(s).

Icon	*Shortcut*	*Action description*
All views within a pool		
⬅	1	Highlight the previous element, (sub)collection or document to assess
⬆	2	Go to the container (sub-collection for an article, etc.)
➡	3	Highlight the next element, (sub)collection or document to assess
Article view		
💾	control+s	Save the current assessment
👁		Hide the pool elements
❌		Show the pool elements
Article view - highlighting mode ✏		
🟨	h	Highlight the currently selected passage.
□	u	Unhighlight the currently selected passage
✏	m	Finish highlighting and switch to the assessing mode
Article view - assessing mode ✏		
✓		Mark the article as finished
✓		Mark the article as not finished
✏	m	Go back to the highlighting mode

INEX 2005 Evaluation Metrics

Gabriella Kazai

Mounia Lalmas

1. INTRODUCTION

This document describes the official INEX 2005 metrics, which are the eXtended Cumulated Gain (XCG) Metrics.

2. RELEVANCE ASSESSMENTS

Relevance assessments are given according to two relevance dimensions: *exhaustivity* and *specificity*. In INEX 2005, exhaustivity is measured using 3 + 1 levels: highly exhaustive ($e = 2$), somewhat exhaustive ($e = 1$), not exhaustive ($e = 0$) and "too small" ($e =?$). Specificity is measured on a continuous scale with values in $[0, 1]$, where $s = 1$ represents a fully specific component (i.e. one that contains only relevant information).

We denote the relevance degree of an assessed component, given by the combined values of exhaustivity and specificity, as (e, s), where $e \in ?, 0, 1, 2$ and $s \in [0, 1]$. For example, $(2, 0.72)$ denotes a highly exhaustive component, 72% of which is relevant content.

An important property of the exhaustivity dimension is its propagation effect, reflecting that if a component is relevant to a query then all its ascendant elements will also be relevant. Due to this property, all nodes along a relevant path[1] are always relevant (with varying degrees of relevance), hence resulting in a recall-base[2] comprised of sets of overlapping elements.

2.1 Relevance assessments for the CAS tasks

This year there are four sets of CAS judgments, one for each of the four CAS interpretations - each derived from the same initial set of judgments.

2.1.1 VVCAS

The assessments as done by the assessors (against the narrative); i.e. no change is made to the original assessment set.

2.1.2 SVCAS

Those VVCAS judgments that strictly satisfy the target element requirement. This set of judgments was computed by taking the VVCAS judgments and removing all judgments that do not satisfy the target element. This is a simple matching process in all except topic 260 in which the target element is specified as //bdy//*, in which case all descendants of //bdy (excluding //bdy) are target elements.

2.1.3 VSCAS

A relevant element is not required to satisfy the target requirement, however the document must satisfy all other requirements specified in the query. In all except two cases, this requirement is that for a judgment of the parent topic to be relevant, it must come from a document that also has SVCAS judgments for all its children. In one exception (topic 247), this conjunction is replaced with a disjunction. In the other exception (topic 250) there are (presently) no judgments.

2.1.4 SSCAS

Those VSCAS judgments that satisfy the target element requirement. The are computed from the VSCAS judgments in the same way that SVCAS judgments are computed from VVCAS judgments.

Note that all CAS tasks were evaluated according to the Thorough strategy (i.e. "overlap=off"), see Section 4.2.1.

3. DEFINITION OF AN IDEAL RECALL-BASE

An ideal recall-base is defined to evaluate the CO.Focussed task.

An ideal recall-base is a subset of the full recall-base, where overlap between relevant reference elements is removed so that the identified subset represents the set of ideal answers, i.e. those elements that should be returned to the user. Given a set of preference relations among the (e, s) value pairs, those components representing the best element for the user are selected. This is done through the definition of a preference function on the possible (e, s) pairs and a methodology for traversing an XML tree and selecting ideal nodes based on their relative preference relations to their structurally related nodes.

[1] A relevant path is a path in an article file's XML tree, whose root node is the article element and whose leaf node is a relevant component (i.e. $(e > 0, s > 0)$) that has no or only irrelevant descendants.

[2] The term recall-base refers to the collection of assessments within the test collection that forms the ground-truth for evaluation experiments.

Quantisation functions are used for modelling various sets of possible user preferences, adjustable to a given user model. The following two functions are used in INEX'05: $quant_{strict}$ (Equation 1) and $quant_{gen}$ (Equation 2). The strict function models a user for whom only fully specific and highly exhaustive components are considered worthy. The generalised (gen) function credits document components according to their *degree* of relevance, hence allowing to model varying levels of user satisfaction gained from not perfect, but still relevant components or near-misses[3].

$$quant_{strict}(e,s) : \begin{cases} 1 & \text{if} \quad e = 2 \quad \text{and} \quad s = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

$$quant_{gen}(e,s) := e * s \tag{2}$$

The following methodology is adopted to traverse the XML trees and select the ideal nodes. Given any two components on a relevant path, the component with the higher quantised score is selected. In case two components' scores are equal, the one higher in the tree is chosen (i.e. parent/ascendant). The procedure is applied recursively to all overlapping pairs of components along a relevant path until one element remains. After all relevant paths have been processed, a final filtering is applied to eliminate any possible overlap among ideal components, keeping from two overlapping ideal paths the shortest one.

The use of an ideal recall-base supports the evaluation viewpoint (needed for the CO.Focussed task) whereby components in the ideal recall-base *should* be retrieved, while the retrieval of near-misses *could* be rewarded as partial successes, but other systems *need not* be penalised for not retrieving such near-misses.

4. OFFICIAL METRICS: EXTENDED CUMULATED GAIN (XCG)

The XCG metrics are a family of metrics that are an extension of the cumulated gain (CG) based metrics of [1] and which aim to consider the dependency of XML elements (e.g. overlap and near-misses) within the evaluation. The extension of these metrics to INEX lies in the way 1) the relevance score for a given document component is calculated via the definition of relevance value (RV) functions, and 2) the definition of the ideal recall-bases (see previous section).

The XCG metrics include the user-oriented measures of normalised extended cumulated gain ($nxCG$) and the system-oriented effort-precision/gain-recall measures (ep/gr).

4.1 xCG and normalised xCG ($nxCG$)

The xCG metric accumulates the relevance scores of retrieved documents along the ranked list. Given a ranked list of document components, xCG, where the element IDs are replaced with their relevance scores, the cumulated gain at rank i, denoted as $xCG[i]$, is computed as the sum of the relevance scores up to that rank:

$$xCG[i] := \sum_{j=1}^{i} xG[j] \tag{3}$$

For example, the ranking $xG_q = <3,1,0,0,1,3,2,2,0,0>$ produces the cumulated gain vector of $xCG = <3,4,4,4,5,8,10,12,12,12>$.

For each query, an ideal gain vector, xI, can be derived by filling the rank positions with the relevance scores of all documents in the recall-base (or as in the case of the CO.Focussed task, with the relevance scores of all elements in the ideal recall-base) in decreasing order of their degree of relevance. The corresponding cumulated ideal gain vector is referred to as xCI. For our toy example, the ideal gain vector may be $xI = <3,3,3,3,2,2,2,1,1,0,...>$, for which we obtain $xCI = <3,6,9,12,14,16,18,19,20,20,...>$.

A retrieval run's xCG vector can then be compared to this ideal ranking by plotting both the actual and ideal cumulated gain functions against the rank position.

By dividing the xCG vectors of the retrieval runs by their corresponding ideal xCI vectors, we obtain the normalised xCG ($nxCG$) measure:

$$nxCG[i] := \frac{xCG[i]}{xCI[i]} \tag{4}$$

For a given rank i, the value of $nxCG[i]$ reflects the relative gain the user accumulated up to that rank, compared to the gain he/she could have attained if the system would have produced the optimum best ranking. For our example gain vector xG, we obtain $nxCG = <1,0.67,0.44,0.33,0.36,0.5,0.56,0.63,0.6,0.6>$. For any rank the normalised value of 1 represents ideal performance.

Systems may be compared at several cutoff values, e.g. $nxCG[1]$ or $nxCG[100]$. In addition, we may average $nxCG$ scores up to a given rank as:

$$MAnxCG[i] := \frac{\sum_{j=1}^{i} nxCG[j]}{i} \tag{5}$$

For example, we obtain $MAnxCG[6] = 0.55$ for our xG vector.

4.2 Calculating a component's relevance value

The definition of the xCG and $nxCG$ metrics is based on the gain value, $xG[i]$, that a user obtains when examining a returned result component. In this section we detail how this gain is calculated.

We define a *relevance value (RV) function*, $r(c_i)$, as a function that returns a value in $[0,1]$ for a component c_i in a ranked result list, representing the component's relevance or gain value to the user. The meaning of such a value may be compared to the notion of utility, reflecting the worth that a retrieved component holds for the user. A score of 0 reflects no relevance or utility, 1 is highest relevance score and values in between represent various gain levels.

[3] Note that the calculation of the generalised quantisation score by multiplying the exhaustivity and specificity values is somewhat ad-hoc and may be revised in the future.

Such a relevance value will depend on a number of factors, like the component's exhaustivity and specificity degree (i.e. its (e, s) values). When retrieval results are assumed independent, the gain value of a document can be obtained as a direct function of these two parameters. When results are dependent, a component's gain value will likely be influenced by additional aspects, such as what the user may have seen at earlier ranks or what additional elements he/she can access from the current returned component. A wide variety of RV functions may be defined, each capturing a different type of user behaviour. Currently XCG supports a simple model of a user, taking into account overlap and near-misses.

4.2.1 Overlap

To reflect the viewpoint of a user for whom any already viewed components become irrelevant, the possible overlap of result elements with already seen components must be considered when calculating the relevance value of a given component. We define the following result-list dependent relevance value (RV) function:

$$rv(c_i) := f(quant(assess(c_i))) \tag{6}$$

where $assess(c_i)$ is a function that returns the assessment value pair (e, s) for the i-th component in the ranking if it is given within the recall-base and $(0, 0)$ otherwise. The function $quant(\cdot)$ is a quantisation function (e.g. Equation 1 and Equation 2).

The $f(\cdot)$ function can take several forms depending on the kind of dependency that exist among result elements within the ranking. We only consider the dependency of a given component to already retrieved components (i.e. those earlier in the ranking). We have three possibilities: 1) a component may be returned for the first time (i.e. no related nodes have been seen by the user before), 2) it may have already been seen by the user in full (e.g. if a container section of the current paragraph result was returned at an earlier rank), 3) some parts of it may have been seen before (e.g. if a paragraph of the current section result has already been returned). We define three versions of $f(\cdot)$ for each of these cases.

For a not-yet-seen component, we set $f(x) = x$. The RV function then returns the quantised assessment value pair $quant(assess(c_i))$:

$$rv(c_{not\_yet\_seen}) := quant(assess(c_i)) \tag{7}$$

For a component that has been fully seen already by the user, we define $f(x) = (1-\alpha)\cdot x$, and hence the RV function as:

$$rv(c_{fully\_seen}) := (1-\alpha)\cdot quant(assess(c_i)) \tag{8}$$

The weighting factor of α represents how much frustration a user is willing to tolerate when accessing redundant components or component-parts. For example, setting $\alpha = 1$ reflects a user who does not tolerate already viewed components. In this case, the RV function returns 0 for a fully seen, and hence redundant, component, reflecting that it represents no value to the user any more. In INEX'05, α is set to 1.

Finally, if a component has been seen only in part before, then its relevance value is calculated as:

$$rv(c_{partially\_seen}) := \alpha \cdot \frac{\sum_{j=1}^{m}(rv(c_j)\cdot\ |c_j|)}{|c_i|} + (1-\alpha)\cdot quant(assess(c_i)) \tag{9}$$

where m is the number of c_i's child nodes and $|\cdot|$ is the length of an element (in characters).

Overlap can be ignored (e.g. for the Thorough tasks) within the evaluation by simply adopting Equation 7 regardless if a result element overlaps with other results or not. We will refer to this mode of evaluation as "overlap=off", while the result-list dependent functions defined above as "overlap=on".

4.2.2 Near-misses

To consider the retrieval of near-misses within the evaluation, we reward a partial score for the retrieval of non-ideal elements that are structurally related to ideal components. This year, only those relevant elements of the full recall-base are considered near-misses which are not included in the ideal recall-base[4].

Given this set of near-misses and the ideal recall-base, the XCG metrics are applied such that the ideal gain vector of a query, xI, is derived from the ideal recall-base, and the gain vectors, xG, corresponding to the system runs under evaluation are based on the full recall-base. The relevance score of a near-miss component is calculated by Equation 6.

Before the final gain value can be assigned to $xG[i]$, we need to apply a dependency normalisation function, which ensures that the total score for any sub-tree of an ideal node cannot exceed the maximum score achievable when the ideal node itself is retrieved. For example, an ideal node may have a large number of relevant child nodes whose total RV score may exceed that of the ideal node. The dependency normalisation function $(rv_{norm}(c_i))$ safeguards against this by ensuring that for any $c_j \in S$:

$$\sum_{c\in S} rv(c) \leq rv(c_{ideal}) \tag{10}$$

where S is the set of retrieved descendant nodes of the ideal node c_{ideal}, and where c_{ideal} is on the same relevant path as c_j. The final normalised relevance score provides the gain value result components: $xG[i] = rv_{norm}(c_i)$.

4.3 Effort-precision: ep

The cumulated gain based measures described so far provide a recall-oriented view of effectiveness at fixed rank positions. We may ask what is the amount of effort required of the user to reach a given level of cumulated gain when scanning a given ranking. Furthermore, we may wish to measure the amount of required effort compared to an ideal ranking. The horizontal line drawn at the cumulated gain value of r, shown in Figure 1, illustrates this view. Based on this, and analogue to the definition of $nxCG$, we define a precision-oriented XCG measure, effort-precision ep as:

$$ep[r] := \frac{e_{ideal}}{e_{run}} \tag{11}$$

[4] Note that $e =?$ are currently considered as $e = 0$

where e_{ideal} is the rank position at which the cumulated gain of r is reached by the ideal curve and e_{run} is the rank position at which the cumulated gain of r is reached by the system run. A score of 1 reflects ideal performance, where the user need to spend the minimum necessary effort to reach a given level of gain.

Effort-precision, *ep*, is calculated at arbitrary gain-recall points, where gain-recall is calculated as the cumulated gain value divided by the total achievable cumulated gain [3]:

$$gr[i] := \frac{xCG[i]}{xCI[n]} = \frac{\sum_{j=1}^{i} xG[j]}{\sum_{j=1}^{n} xG[j]} \tag{12}$$

where n is the total number of relevant documents. The meaning of effort-precision at a given gain-recall value is the amount of relative effort (where effort is measured in terms of number of visited ranks) that the user is required to spend when scanning a system's result ranking compared to the effort an ideal ranking would take in order to reach a given level of gain relative to the total gain that can be obtained.

This method follows the same viewpoint as standard precision/recall, where recall is the control variable and precision the dependent variable. In our case, the gain-recall is the control variable and effort-precision the dependent variable. As with precision/recall, interpolation techniques are necessary to estimate effort-precision values at non-natural gain-recall points, e.g. when calculating effort-precision at standard recall points of $[0.1, .., 1]$, denoted as e.g. *ep*@0.1. Given the nature of our xCG curve, a simple linear interpolation method is used.

As with standard precision/recall, the uninterpolated mean average effort-precision, denoted as *MAep*, is calculated by averaging the effort-precision values obtained for each rank where a relevant document component is returned. For non retrieved relevant document component a precision score of 0 is assigned. We also calculate an average over the interpolated effort-precision values at standard recall points, which we refer to as *iMAep*. Analogue to recall/precision graphs, we plot effort-precision against gain-recall and obtain a detailed summary of a system's overall performance.

4.4 The Q and R metrics

A criticism of the $nxCG$ metrics is that they do not average well across topics [2] (in [4]). The reason for this is that as the total number of relevant documents differs across topics, so does the upperbound performance at fixed ranks.

A solution has been suggested in [4] in the form of the following measures, where the explicit incorporation of the rank position in the denominator ensures that performance is calculated against an always increasing ideal value:

$$Q - measure = \frac{1}{R} \sum_{j=1}^{i} isrel(d_j) \frac{cbg(j)}{cig(j) + j} \tag{13}$$

where R is the total number of relevant documents, d_j is the document retrieved at rank j, $isrel(\cdot)$ is a binary function that returns 1 if the document is relevant (to any degree) and 0 otherwise. The function $cbg(\cdot)$ is a so-called cumulated bonus gain function, which is defined as $cbg(i) := bg(i) + cbg(i-1)$, where $bg(i) := g(i) + 1$ if $g(i) > 0$ and $bg(i) := 0$ otherwise, and $g(i)$ is the gain value at rank i. The function $cig(\cdot)$ is the cumulated bonus gain derived for the ideal vector (analogue to $cbg(\cdot)$).

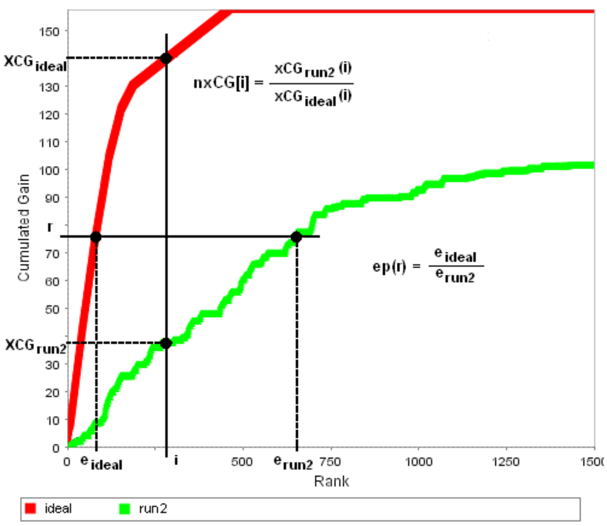


Figure 1: Calculation of $nxCG$ and effort-precision (ep)

$$R - measure = \frac{cbg(R)}{cbg(R) + R} \tag{14}$$

We employ versions of the above measures, adapted to XML and referred to as Q and R.

5. WHAT RESULTS ARE REPORTED?

The results of the following metrics are reported:

- Effort-precision/gain-recall (ep/gr) at standard gain-recall points (e.g. 0.1,0.2,..,1) (e.g. *ep*@0.1: effort-precision at 0.1 gain-recall).
- Non-interpolated mean average effort-precision ($MAep$), calculated as the average of effort-precision values at each natural gain-recall point (i.e. whenever a relevant XML element is found).
- Interpolated mean average effort-precision ($iMAep$), calculated as the average of effort-precision values at the standard gain-recall points.
- Normalised xCG ($nxCG$) at various rank positions (e.g. $nxCG[30]$ denoted as $nxCG@30DCV$: $nxCG$ at rank 30, and $nxCG[0.1rank\%]$, denoted as $nxCG@0.1rank\%$: nxCG at 10% of the length of the output list, e.g. 1500).
- Mean average $nxCG$ at rank 1500 ($MAnxCG[1500]$), calculated as the average of $nxCG[i]$ values for $i = 1$ to 1500.
- Q and R.

The official system-oriented evaluation will be based on the ep/gr measures, with $MAep$ being the main overall performance indicator. The official user-oriented evaluation will be based on the $nxCG$ measures.

All results can be accessed at https://inex.lip6.fr/2005/metrics/ using your INEX login. The official rankings will be posted on the INEX web site.

6. EVALJ

All metrics have been implemented within a single Java project: the EvalJ evaluation package, which can be downloaded from https://sourceforge.net/projects/evalj/.

To download the code, please follow the instructions on https://sourceforge.net/cvs/?group_id=136430 and use:

```
cvs -d:pserver:anonymous@cvs.sourceforge.net:
/cvsroot/evalj login
cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:
/cvsroot/evalj co -P EvalJ
```

Alternatively, installer files can be accessed from http://evalj.sourceforge.net/. These should just install directly.

There is a README included within EvalJ, detailing how to run the various metrics.

7. EVALUATING DIFFERENT TASKS

The XCG metrics take several parameters, which define how e.g. overlap is to be handled. These parameters are read at run time from a config file. A config file, `inex2005.prop` is provided within EvalJ, which contains the official settings for the parameters and their explanations. These are as follows.

7.1 CO.Focussed and COS.Focussed

These tasks are evaluated using the overlap=on option, which means that overlap and near-misses are considered within the evaluation. The ideal recall-base is generated automatically within the evaluation based on the selected quantisation functions and the methodology described in section 3. The assessment pool IDs are given to filter the total set of assessments, which is necessary in order to select the official pools, e.g. for multi-assessed topics.

```
TASK: CO.Focussed #or COS.Focussed
METRICS: nxCG, ep/gr, q
ALPHA: 1.0
OVERLAP: on
QUANT_FUNCTIONS: gen, strict
DCV: 1, 2, 3, 4, 5, 10, 15, 25, 50, 100, 500, 1000,
1500
ASSESSMENTS_DIR: $apath/adhoc2005/official/CO+S/*/
QUERY_TYPE: CO+S
SUBMISSIONRUNS_DIR: $apath/inex2005_runs/*/
```

7.2 CO.Thorough and COS.Thorough

These tasks are evaluated using the overlap=off option, which mean that overlap is tolerated within the evaluation. This means that no ideal recall-base is generated and the gain value of a component is only a function of its exhaustivity and specificity values, regardless if it overlaps or not with a previously returned element. The assessment pool IDs are given to filter the total set of assessments, which is necessary in order to select the official pools, e.g. for multi-assessed topics.

```
TASK: CO.Thorough #or COS.Thorough
METRICS: nxCG, ep/gr, q
ALPHA: 1.0
OVERLAP: off
QUANT_FUNCTIONS: gen, strict
DCV: 1, 2, 3, 4, 5, 10, 15, 25, 50, 100, 500, 1000,
1500
ASSESSMENTS_DIR: $apath/adhoc2005/official/CO+S/*/
QUERY_TYPE: CO+S
SUBMISSIONRUNS_DIR: $apath/inex2005_runs/*/
```

7.3 SSCAS, SVCAS, VSCAS and VVCAS

These tasks are evaluated based on the Thorough task assumption, with overlap=off.

```
TASK: SSCAS #or SVCAS, VSCAS, VVCAS
METRICS: nxCG, ep/gr, q
ALPHA: 1.0
OVERLAP: off
QUANT_FUNCTIONS: gen, strict
DCV: 1, 2, 3, 4, 5, 10, 15, 25, 50, 100, 500, 1000,
1500
ASSESSMENTS_DIR: $apath/official/SSCAS/ #or .../SVCAS/,
.../VSCAS/, .../VVCAS/
QUERY_TYPE: CAS
SUBMISSIONRUNS_DIR: $apath/inex2005_runs/*/
```

7.4 CO.FetchBrowse and COS.FetchBrowse

The evaluation methodology for this task is different from all other tasks in that two separate evaluation scores are calculated: an article-level and an element-level score.

The article-level score regards a system's ability to find relevant documents in the first place. To obtain this score, we first filter the recall-base to only contain article nodes. The ideal gain vector is obtained by sorting the filtered set by quantised score. We compare this ideal gain vector to the list of article nodes *derived* from a system run. To derive the list of articles from a run, we reduce each XML element in the run to its article root and keep from any two duplicate entries the first occurrence (e.g. from $< a1/e1, a1/e2, a2 >$ we derive $< a1, a2 >$).

The element-level score reflects a system's ability to locate the relevant elements within an article document. We report an average of the scores calculated for each cluster (where a cluster is made up of the list of elements returned within a given article). The recall-base for a given cluster consists of the relevant elements within the given article (the article element itself removed), where elements are ordered in decreasing quantised value. The list of elements in a cluster in a run (again with article nodes removed) is then compared against the cluster's recall-base directly. The individual cluster-scores are then averaged over all clusters and then over all queries.

We only report effort-precision/gain-recall measures for the FetchBrowse tasks, as the selection of an appropriate document cutoff value for nxCG is an open question (due to the small number of relevant elements within each cluster-recall-base, i.e. article). In addition, we only use the overlap=off option as overlap=on requires further considerations.

```
TASK: CO.FetchBrowse #or COS.FetchBrowse
METRICS: nxCG, ep/gr, q
ALPHA: 1.0
OVERLAP: off
QUANT_FUNCTIONS: gen, strict
DCV: 1, 2, 3, 4, 5, 10, 15, 25, 50, 100, 500, 1000,
1500
ASSESSMENTS_DIR: $apath/adhoc2005/official/CO+S/*/
QUERY_TYPE: CO+S
SUBMISSIONRUNS_DIR: $apath/inex2005_runs/*/
```

8. WHAT ELSE IS TO COME?

We will publish results where the too small elements are considered as relevant and will also evaluate the FetchBrowse tasks using overlap=on. These, however, require changes to the EvalJ code, which will be distributed under a new version number after the workshop, with possible additional changes that may result from discussions at the workshop. Please also let us know of any problems and bugs you may find so that these can also be fixed. Thanks.

9. ACKNOWLEDGMENTS

We are very grateful for the work of Benjamin Piwowarski, who has provided the metrics web interface (at https://inex.lip6.fr/2005/metrics/) and has also implemented the inex-2002 (inex-eval) and inex-2003 (inex-eval-ng) metrics in EvalJ, the results of which are reported alongside the official metrics.

We would like to thank Andrew Trotman for providing the CAS assessment pools for the different CAS tasks and for contributing section 2.1 to this paper.

Special thanks to Arjen de Vries and Saadia Malik for their invaluable help with hands-on work as well as for their many useful comments and email discussions.

Many thanks to all members of the organisers mailing list for their helpful comments and special thanks to all participants for all their efforts and hard work.

10. REFERENCES

[1] K. Järvelin and J. Kekäläinen. Cumulated Gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (ACM TOIS)*, 20(4):422–446, 2002.

[2] N. Kando, K. Kuriyama, and M. Yoshioka. Information retrieval system evaluation using multi-grade relevance judgements - discussion on averageable single-numbered measures (in japanese). Technical report, 2001.

[3] J. Kekäläinen and K. Järvelin. Using graded relevance assessments in IR evaluation. *Journal of the American Society for Information Science and Technology*, 53(13):1120–1129, 2002.

[4] T. Sakai. New performance metrics based on multigrade relevance: Their application to question answering. In *NTCIR Workshop 4 Meeting Working Notes*, June 2004.

INEX 2005 Heterogeneous Track Tasks and Result Submission Specification

Heterogeneous Track Tasks

The Heterogeneous track has two tasks this year “HET.CO” and “HET.CAS”. The first task focuses on content-oriented XML retrieval and second attempts to use structural clues in retrieval. For the HET.CO task the goal is to retrieve elements from the various collections based on their content (or implied content in the case of references for articles, books, etc.), therefore the issue for retrieval is mapping the content specification of the topic titles across the collections regardless of structure. For HET.CAS the retrieval must map not only content (and implied content) but also structure, or *implied* structure. For example if a query specifies that an “article” is to be about a topic, then the results should rank articles (whether full text or bibliographic entries) ahead of other elements or document types (such as books, PhD Theses, web pages, etc.)

Topics

There are also two types of topics this year: Those selected from the Adhoc Track, drawn from the CO+S adhoc topics, and those created especially for the track. Topics will include both a text title and the NEXI castitle elements. (Topics will be announced on Nov. 1)

Collections

The specific collections for the Heterogeneous track are available on the INEX participants web site. These are available as compressed tar files:

1. Berkeley2.tar.gz
2. bibdbpub2.tar.gz
3. CompuScience2.tar.gz
4. dblp2.tar.gz
5. hcibib2.tar.gz
6. qmuldcsdbpub2.tar.gz
7. zdnet-xml.tar.gz

In addition, the IEEE collection used in the Adhoc track and the LonelyPlanet collection used in Multimedia track are available (Use of the LonelyPlanet data is optional and requires a separate user agreement like those used for the IEEE data -- see the Multimedia track section of the INEX participants web site for details.)

Result Submission

For each topic up to 6 runs may be submitted, 3 for the HET.CO task and 3 for the HET.CAS task. The results of one run must be contained in one submission file (e.g. up to 3 files can be submitted for each task). A submission may contain up to 1500 retrieval results for each of the topics.

Submission format

For relevance assessments and the evaluation of the results we require submission files to be in the same format as used for the adhoc tasks, as described in this section. The overall submission format is defined in the following DTD:

```
<!ELEMENT inex-submission (description, collections, topic+)>
<!ATTLIST inex-submission
participant-id CDATA #REQUIRED
run-id CDATA #REQUIRED
task (HET.CO | HET.CAS) #REQUIRED
query (automatic | manual) #REQUIRED
>
```

```
<!ELEMENT description (#PCDATA)>
<!ELEMENT topic (result*)>
<!ATTLIST topic
topic-id CDATA #REQUIRED
>
<!ELEMENT collections (collection+)>
<!ELEMENT collection (#PCDATA)>
<!ELEMENT result (in?,file, path, rank?, rsv?)>
<!ELEMENT in (#PCDATA)>
<!ELEMENT file (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT rank (#PCDATA)>
<!ELEMENT rsv (#PCDATA)>
```

Each submission must contain the participant ID of the submitting institute (available on the INEX participants' site), a run ID (which must be unique for the submissions sent from one organisation – also please use meaningful names as much as possible), the task which should be HET.CO or HET.CAS and an indication of whether the query was constructed automatically or manually from the topic.

Furthermore each submitted run must contain a description of the retrieval approach applied to generate the search results. A submission contains a number of topics, each identified by its topic ID (as provided with the topics).

For each submission the collections used in the run must be specified. A maximum of 1500 result elements may be included for each topic. A result element is described by a collection name in which the result element has been found (using the optional "in" tag), by a file name, and an element path. It may include rank and/or retrieval status value (rsv) information.

Before detailing the various elements of the DTD, here is a sample submission file:

```
<inex-submission participant-id="3" run-id="HET_CO_RUN1" task=”HET.CO”
query=”automatic”>
<description>Using Logistic Regression to match titles and abstracts
for all databases
</description>
<collections>
<collection>ieee</collection>
<collection>Berkeley2</collection>
<collection>bibdbpub2</collection>
<collection>CompuScience2</collection>
<collection>dblp2</collection>
<collection>hcibib2</collection>
<collection>qmuldcsdbpub2</collection>
<collection>ZDNet</collection>
</collections>LonelyPlanet</collection>
<topic topic-id="01">
<result>
<in>Berkeley2</in>
<file>Berkeley2/95/09/CUBGLAD84227121S</file>
<path>/USMARC[1]/VarFlds[1]/VarDFlds[1]</path>
<rsv>0.67</rsv>
</result>
<result>
<in>dblp2</in>
<file>dblp2/2003/08/26/journals/jcc/FraderaS02</file>
<path>/entry[1]/article[1]</path>
<rsv>0.1</rsv>
</result>
[ ... ]
</topic>
<topic topic-id="02">
[ ... ]
</topic>
[ ... ]
</inex-submission>
```

Rank and RSV

The `rank` and `rsv` elements are provided for submissions based on a retrieval approach producing ranked output. The ranking of the result elements can be described in terms of

- Rank values, which are consecutive natural numbers, starting with 1.
- Retrieval status values (RSVs), which are positive real numbers. Note that there may be several elements having the same RSV value.

Either of these methods may be used to describe the ranking within a submission. If both rank and rsv are given, the rank value is used for evaluation. These elements may be omitted from a submission if a retrieval approach does not produce ranked output.

In, File and path

Since XML retrieval approaches may return arbitrary XML nodes from the documents of the INEX Het Track collection, we need a way to identify these nodes without ambiguity. This year the structure of the collections has been changed to enable support by the XRai evaluation tool. This means that instead of single (or a few) files for each Het collection, there is now a directory structure for each collection with each individual file representing a single xml document (as has always been the case in the IEEE collection). Now for INEX Het Track submissions, elements are identified by means of an “in” element (which includes the name of the collection for a given result), the file name (which also includes the collection name as well as the file path and file name for each individual document file), and element (node) path specification, which must be given in XPath syntax. File names for each collection (except IEEE) are rooted in the top-level collection directory, which is named for the collection and excludes higher-level directories in your local file path. The file path should use '/' for separating directories. The 2 exceptions are the file names for the IEEE collection (which should be in the same form used in the adhoc tasks) and the ZDNet collection which should have the form “xml/articles/21…” Note also that only article files for the IEEE collection should be used (e.g. no volume.xml files). The extension “.xml” must be left out for ALL file names.

Element paths are given in XPath syntax. To be more precise, only fully specified paths are allowed, as described by the following grammar:

```
Path           ::=  '/' ElementNode Path
                  | '/' ElementNode '/' AttributeNode
                  | '/' ElementNode

ElementNode    ::=  ElementName Index

AttributeNode  ::=  '@' AttributeName

Index          ::=  '[' integer ']'
```

Example:

```
/article[1]/bdy[1]/sec[4]/p[3]
```

This path identifies the element which can be found if we start at the document root, select the first “article” element, then within that, select the first “bdy” element, within which we select the fourth “sec” element, and finally within that element we select the third “p” element.

Important: XPath counts elements starting with 1 and takes into account the element type, e.g. if a section had a title and two paragraphs then their paths would be given as: `../title[1]`, `../p[1]` and `../p[2]`.

A result element may then be identified unambiguously using the combination of its file name and element path.

Example:

```
…
<result>
   <file>qmuldcsdbpub2/1996/inproceedings/Doc_711</file>
   <path>/DOCUMENT[1]/TITLE[1]</path>
</result>
…
```

Result Submission Procedure

The run submission link will be set up as soon as possible. Details will be sent to the Het track mailing list.

October 25, 2005
Ray R. Larson

INEX 2005 Multimedia Track - Working Document

Roelof van Zwol Mounia Lalmas Gabriella Kazai

October 4, 2005

1 Introduction

This guide serves as the working document for the INEX 2005 Multimedia Track and will accumulate all necessary guides.

1.1 Objective

The main objective of the INEX 2005 multimedia track is to provide an evaluation platform for structured document retrieval systems that do not only include text in the retrieval process. Many structured document collections today also contain other types of media, such as images, speech, and video. To include these media types into the retrieval process and to produce a meaningful ranking is far from trivial.

Using the structure of the document as a semantic/logical backbone for the retrieval of multimedia document fragments will allow us to investigate this problem from a new perspective. In the first year of the multimedia track, we plan to provide an evaluation platform for the retrieval of multimedia structured document fragments, rather similar to the methodology used for the INEX Ad Hoc track.

1.2 Task description

The task for the multimedia track is to retrieve relevant document fragments based on an information need with a structured multimedia character. A structured document retrieval approach in that case should be able to combine the relevances of the different media types into a single (meaningful) ranking that is presented to the user. The INEX multimedia track differs from other approaches in multimedia information retrieval, in the sense that it focuses on using the structure of the document to extract, relate and combine the relevances of different multimedia fragments.

The focus for 2005 will be on the combination of text and image retrieval. For this purpose we will use the Lonely Planet document collection, which can be explored at: `http://contentlab.cs.uu.nl/lonelyplanet/Collections/LonelyPlanet/`. The required login and password for this web-site is sent to the participating groups, once the data-handling agreement for the Lonely Planet collection is received by the organization.

2 Topic development guide

This topic development guide is heavily based on the developement guide used for the INEX 2005 Adhoc track[1] Here we will redefine the Topic format and Topic formulation procedure. For clarity we

[1] http://inex.is.informatik.uni-duisburg.de/2005/internal/pdf/TD05.pdf

have inserted a part of the section about topic creation criteria.

2.1 Topic creation criteria

Creating a set of topics for a test collection requires a balance between competing interests. The performance of retrieval systems varies largely for different topcis. This variation is ussually greater than the performance variation of different retrieval methods on the same topic. Thus, to judge whether one retrieval strategy is (in general) more effective than another, the retrieval performance must be averaged over a large and diverse set of topics. In addition, to be a useful diagnostic tool, the average performance of the retrieval systems on the topics can be neither too good nor too bad as little can be learned about retrieval strategies if systems retrieve no, or only relevant, documents.When creating topics, a number of factors should be taken into consideration. Topics should:

- be authored by an expert in (or someone familiar with) the subject areas covered by the collection,
- reflect real needs of operational systems,
- represent the type of service an operational system might provide,
- be diverse,
- differ in their coverage, e.g. broad or narrow topic queries,
- be assessed by the topic author.

In addition we add for the multimedia track that the topics should:

- have a clear multimedia character, which should be clearly specified in the topic description and narrative.

Within the multimedia track we'll focus on the content and structure topics, as these allow explicit formulation of the multimedia character in the information request, e.g. NEXI-CAS query.

Consider for example the topic based on the Lonely planet collection below:

Example 1
<inex_topic inex_track="MM" query_type="CAS" topic_id="0">
<description>Find images depicting scuba diving activities for destinations with a tropical climate and that discuss exploring the beautiful underwater nature by snorkeling and diving activities</description>
<castitle>//destination[about(.//weather,tropical climate) and about(.//activities, beautiful "underwater nature" snorkeling and diving)]//images//image[about(., scuba diving)]</castitle>
<narrative>We want to go on a scuba diving trip to a destination with a tropical climate and beautiful underwater scenery (nature). Therefore we want to find some images on this subject, so that we can have some pre-holiday fun.</narrative>
</inex_topic>

The `description` and `narrative` specify the information need and its explanation. Based on this information need, we can formulate a NEXI-query as given in the `title` section of the topic definition. Our example query above includes both textual and image components of the information need, e.g. `about(.//weather,tropical climate)` is a condition that the weather textual element of a destination should talk about tropical climate, while the `//image[about(., scuba diving)` asks for images that depict scuba diving scenes.

Although the target elements of the above example are images, so far, simple textual retrieval approaches may be sufficient to produce the required output by searching image captions. However, a combination of text and image retrieval systems is encouraged within the track as these may in fact produce better results.

Consider next the following example topic:

Example 2
<inex_topic inex_track="MM" query_type="CAS" topic_id="0">
<description>Find images depicting scuba diving activities, like in BN5970_6.jpg, for destinations with a tropical climate and that discuss exploring the beautiful underwater nature by snorkeling and diving activities</description>
<castitle>//destination[about(.//weather,tropical climate) and about(.//activities, beautiful "underwater nature" snorkeling and diving)]//images//image[about(., scuba diving) and about(., src: /image/BN5970_6.jpg)]</castitle>
<narrative>We want to go on a scuba diving trip to a destination with a tropical climate and beautiful underwater scenery (nature). Therefore we want to find some images on this subject, so that we can have some pre-holiday fun.</narrative>
</inex_topic>

In this example, an instance of the *about* clause, `about(., src:/image/BN5970_6.jpg)`, uses a slightly different syntax to the usual two arguments of a path directive and a textual description of the information need. Here the second argument contains a reference to a multimedia object, in this case, the name and path of an image file. This extension of the *about* clause allows for querying by sample.

By expressing both the content and image components of the information need within the same *about* clause, we are effectively overloading its meaning, leaving it to the retrieval system to decide if a text or image search (or both) is required. The reason for doing so is to emphasise the multimedia nature of the track. Using the extended *about* clause, we can specify query constraint for a document fragment (which may be pure text, image, or a combination of multiple media) using a textual descriptions (e.g. `about(//image, scuba diving)`, `about(//destination, scuba diving)`) or using similar images (e.g. `about(//image, src:/image/BN5970_6.jpg)`, `about(//destination, src:/image/BN5970_6.jpg)`).

Various combinations of query conditions will require different strategies where text and image retrieval can be combined. The track's focus is on the combination of the two techniques and therefore we encourage the submission of topics that force systems to implement content-based image retrieval (e.g. `about(//destination, src:/image/BN5970_6.jpg)`).

2.2 Topic format

The topic format for the multimedia track consists of the following fields: a **description**, **castitle**, and **narrative**. The following information should be contained in each of these fields:

- `<description>` A brief description of the information need, specifying any structural, textual, and visual requirements/composition on the content. The description must be precise, concise, and informative, but it must contain the same terms and the same structural requirements that appear in the `castitle`, albeit expressed in natural language.
- `<castitle>` A valid NEXI expression based on the Lonely Planet document collection that contains at least one *about* clause containing at least one image component. The expression is of the form *//A[B]* or *//A[B]//C[D]*.
- `<narrative>` A detailed explanation of the information need and the description of what makes and element relevant or not. The narrative should explain not only what information is being sought, but also the context and motivation of the information need, i.e. *why* the information is being sought and what purpose it may serve. Assessments will be made on compliance to the narrative alone; it is therefore important that this description is clear and precise.

The additional constraints, as defined in the Topic development Guide for the INEX Ad Hoc track apply, for so far they are applicably and not overruled in this additional guide.

2.3 Topic development guidelines

Each participating group will have to submit 3 (CAS) topics by July 15, 2005 using the on-line form provided.

The topic creation process is divided in several steps.

Step 1: Initial topic statement

Create a one or two sentence description of the information you are seeking. This should be a simple description of the information need without regard to retrieval system capabilities or document collection perculiarities. This should be recorded in the Initial Topic Statement field. Record also the context and motivation of the information need, i.e. why the information is being sought.

Step 2: Exploration phase

In this step the initial topic statement is used to explore the collection. Obtain an estimate of the number of relevant elements, then evaluate whether this topic can be judged consistently. We have two search engines available, a text-based system and an image-based system, which can be used by the participants for topic development. These are reachable from: `http://contentlab.cs.uu.nl/~lonelyplanet/`.

Step 2a: Assess the top 25 text fragments

You have to judge the relevancy of the retrieved text fragments (using binary relevance only). Each result should be judged on its own merits. Abandon a search, if there are fewer than 2 or more than 20 relevant text fragments in the result list.

Step 2b: Assess top 25 images

Since most participants will not have an off-the-shelves system available for the multimedia track, we have chosen to do a separate scan for the relevance of the image component. Therefore you'll have to assess the top 25 images and judge their relevance (using binary relevance only). Each result should be judged on its own merits. Abandon the search, if there are fewer than 2 or more than 20 relevant images in the result list.

Step 2c: Inspect document matching

To assure that the document collection has a reasonable chance of completely fulfilling the text and image-based constraints of the information need a check at document level is needed. In this step, you have to count the number of documents that both satisfy the textual conditions and the image conditions. I.e. if a relevant text fragment was found, and there is an image that belongs to the same XML document, a match is found that corresponds to the information need. Abandon the topic if less than 3 matches are found over the top 25 results for both components.

Step 3: Write description, title, and narrative

During this step you're asked to complete the topic definition by writing the description, title, and narrative. Please obey the instruction that are given for the topic development procedure, as described in the INEX 2005 Guidelines for Topic Development.

Step 4: Topic submission

Once you are finished, fill out the on-line Candidate Topic Submission Form on the INEX website at http://inex.is.informatik.uni-duisburg.de/2005 under Tasks/Tracks → Multimedia → Topics → Submit Topic. Make sure you submit all your candidate topics no later than 15 July, 2005. Thank you.

3 Relevance assessments guidelines

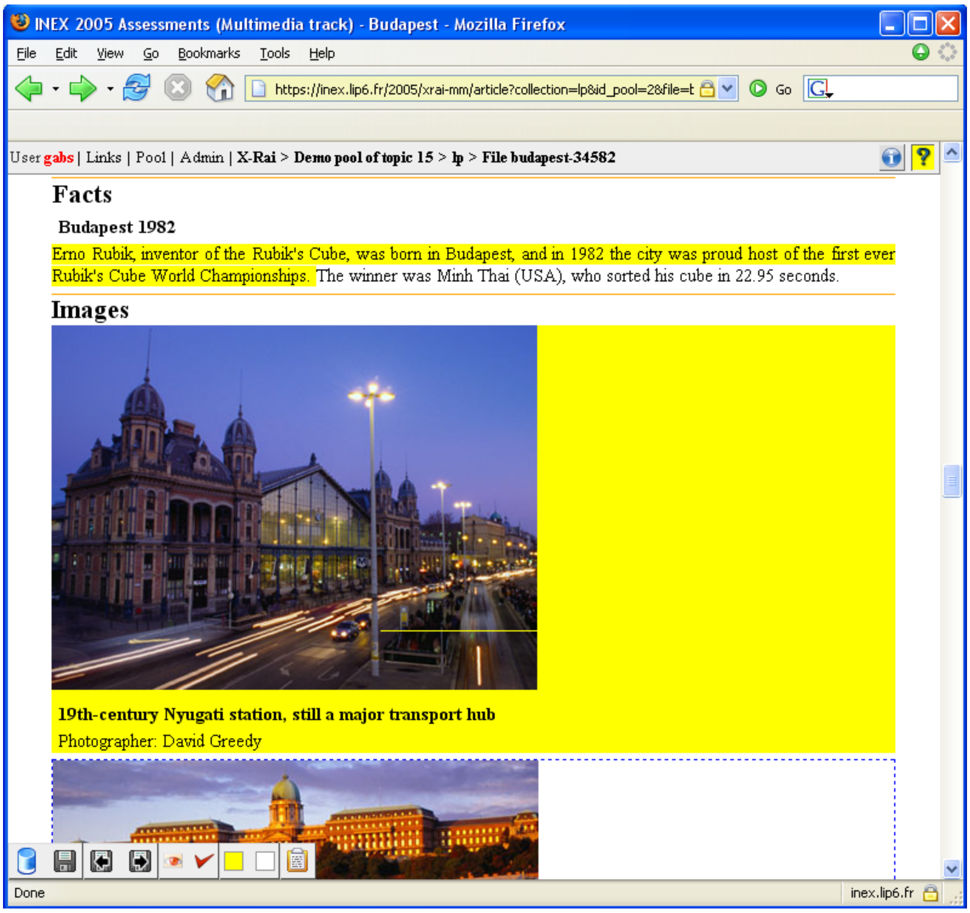
We base the definition of relevance on the definition employed in the INEX ad-hoc track with the exception that we measure exhaustivity only on a binary scale. In addition, reflecting the SSCAS task, we only consider an XML element relevant if it strictly matches the structural conditions specified within the query, i.e. only target elements may be relevant and only if they are contained in an XML document that satisfies the query's containment constraints.

Therefore, a given multimedia fragment is said to be relevant if it "discusses" (or depicts) the topic of request to any degree and if it strictly adheres to the structural conditions requested by the user.

Similarly to the ad-hoc track, the assessment procedure follows the highlighting approach. However, given the binary nature of relevance, the assessment procedure for the multimedia track consists only of a single pass. During this single pass assessors are to highlight multimedia fragments that contain only relevant content, i.e. relevant content that contains no (or only minimal) non-relevant content. In the case of textual content, only relevant text fragments, e.g. words or sentences, should be highlighted. In the case of images, since currently it is not possible to highlight only a part of an image, the whole image should be highlighted if it contains relevant content (regardless of how much of the image may be non-relevant).

The on-line assessment interface can be accessed at https://inex.lip6.fr/2005/xrai-mm/ using your multimedia track login. The assessment system is described in the INEX 2005 Relevance Assessment Guide[2]. The only difference is that the Toolbar has been simplified. Firgure 1 shows a document of the Lonely Planet collection being assessed. The toolbar can be seen in the bottom left corner. The icons from left-to-right are: remove assessments, save assessments, save assessments and move to the previous document, save assessments and move to the next document, show or hide support elements(i.e. retrieval results), set document as assessed, highlight selected text fragment, unhighlight

[2] http://inex.is.informatik.uni-duisburg.de/2005/internal/pdf/Relevance_Assessment2005.pdf


INEX 2005 Assessments (Multimedia track) - Budapest - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
https://inex.lip6.fr/2005/xrai-mm/article?collection=lp&id_pool=28&file=t
Go
User gabs | Links | Pool | Admin | X-Rai > Demo pool of topic 15 > lp > File budapest-34582
Done
inex.lip6.fr

Facts

Budapest 1982

Erno Rubik, inventor of the Rubik's Cube, was born in Budapest, and in 1982 the city was proud host of the first ever Rubik's Cube World Championships. The winner was Minh Thai (USA), who sorted his cube in 22.95 seconds.

Images

19th-century Nyugati station, still a major transport hub

Photographer: David Greedy

Figure 1: The on-line assessment system and toolbar

selected text fragment and passage information. Please make sure you set a document as assessed and save it before moving onto the next document to assess.

All highlighted multimedia fragments will be saved as exhaustive, fully specific and strictly satisfying the structural constraints.

Given the small collection and the strict structural match we expect that assessment work will be light. However, the strict relevance criteria may bring up unforeseen questions that we would like to hear about. We would like to ask assessors to keep a log book and record any questions or uncertainties that arised during assessment or any situations when you felt you had to make perhaps not-so-obvious decisions. For each issue or question, please record the topic id and the particular XML element that sparked the question. To help you with this reporting task, the assessment system provides an easily accessible passage information facility (the right-most button of the Toolbar). Simply highlight a passage and click the clipboard button to display all the reference information that you need to include in a log entry. Please then submit your log to us or even make it available to everyone by emailing the list of multimedia participants. Thank you.

4 Acknowledgements

This guide is heavily based on the Topic Development Guide for the INEX 2005 Ad hoc track. We are thankful for the guidelines provided there and thankfully adopt them, for as far as applicable, for the INEX 2005 Multimedia track.

We would like to express our gratitude for the work of Benjamin Piwowarski for providing the necessary modifications to the on-line assessment system.

5 Example

Based on the example introduced earlier, the following should be submitted:

1) Affiliation: Utrecht University, the Netherlands
2) Author (name, e-mail): Roelof van Zwol, roelof@cs.uu.nl
3) Initial topic statement:
We want to go on a scuba diving trip to a destination with a tropical climate and beautiful underwater scenery (nature). Therefore we want to find some images on this subject, so that we can have some pre-holiday fun.
4) Number of relevant text-fragments in TOP 25: 14
5) Top 25 retrieved text fragments (path, document):

```
file:  comoros-and-mayotte-755.xml path:/destination[1]/images[1]
file:  bonaire-3807.xml path:  /destination[1]/images[1]
...
file:  vancouver-18170.xml path:/destination[1]/activities[1]
```

6) Number of relevant image fragments in TOP 25: 7
7) Top 25 retrieved images (image name or path, document):

```
file:  australia-245.xml image:  /images/BN918_1.jpg
file:  belize-590.xml image:  /images/BN9002_8.jpg
...
file:  comoros-and-mayotte-755.xml image:  /images/BN5970_6.jpg
```

8) Matching documents: 4
9) Candidate topic:

```
<inex_topic inex_track=‘‘MM’’ query_type="CAS">
<description>Find images depicting scuba diving activities for destinations with
a tropical climate and that discuss exploring the beautiful underwater nature by
snorkeling and diving activities.</description>
<castitle>//destination[about(.//weather,tropical climate) and about(.//activities, beautiful ”underwater nature” snorkeling and diving)]//images//image[about(., scuba diving) and about(., src:/image/BN5970_6.jpg)]</castitle>
<narrative>We want to go on a scuba diving trip to a destination with a tropical
climate and beautiful underwater scenery (nature).  Therefore we want to find some
images on this subject, so that we can have some pre-holiday fun.</narrative>
</inex_topic>
```

INEX iTrack Daffodil

11th October 2005

1 Introduction

DAFFODIL[1] is a retrieval system aiming at providing strategic support during the information seeking process in Digital Libraries. A custom-fitted version of Daffodil has been setup as the baseline system for this year's Interactive Track. This version includes the following tools: Search tool, Progress tool and Related Term tool. For the search tool at the backend the *Hyper-media Retrieval Engine for XML (HyREX)*[2] has been used.

2 Logging in

In the Login screen (Figure 1), the experimenter has to fill in information relating to:

- Login name and password,
- Task, possible values are A and B
- Topic Id, the unique id assigned to a topic
- Rotation, indication of the order in which the current test person performs the tasks.

Please consult the Track Guidelines document for the meanings of these identifiers, and details about how to log in when doing experiments.

Once you have logged in please minimise or close the Personal Library on the right, as it is not to be used in the experiments.

3 Search Tool

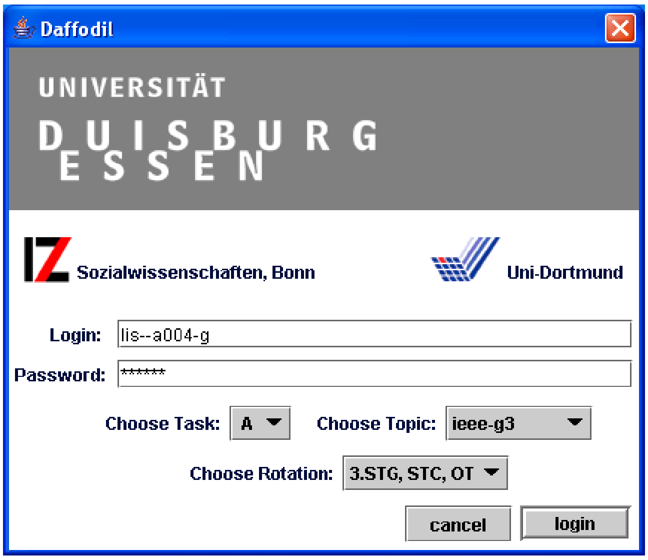
The search tool (Figure 2) view is divided into two parts. The first part is the *Query Form*. Depending on the document collection, different search fields are made available. For the IEEE collection, the available fields are title, author, publication year and freetext (for fulltext search). The second part is the *Result List* where the search hits will be presented (see below).

3.1 Performing a Search

If you have entered a query, e.g. "GPS mobile phones", in the freetext field and hit the search button, the search is performed in the full text of the selected collection. The current query is presented in the frame below, and can be edited there.

[1] http://www.is.informatik.uni-duisburg.de/projects/daffodil

[2] http://www.is.informatik.uni-duisburg.de/projects/hyrex/



Login: lis--a004-g

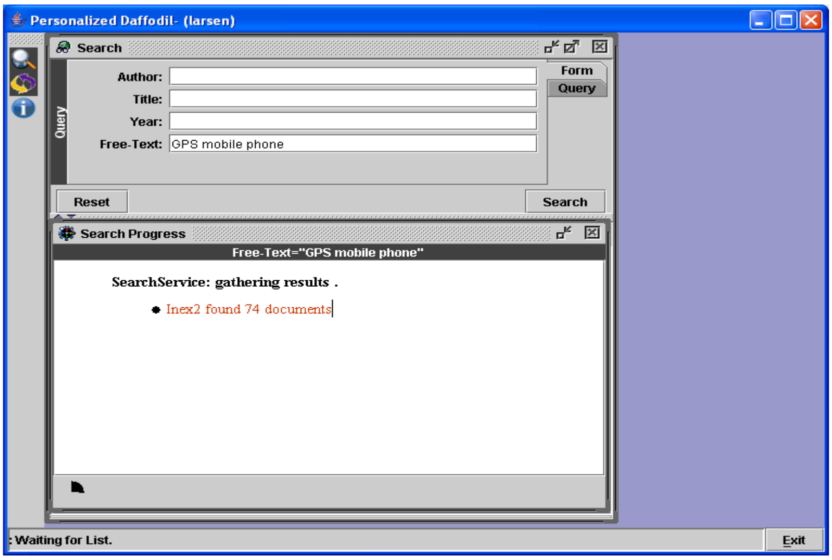
Password: ******

Choose Task: A

Choose Topic: ieee-g3

Choose Rotation: 3.STG, STC, OT

Figure 1: Login screen



Author:

Title:

Year:

Free-Text: GPS mobile phone

Free-Text="GPS mobile phone"

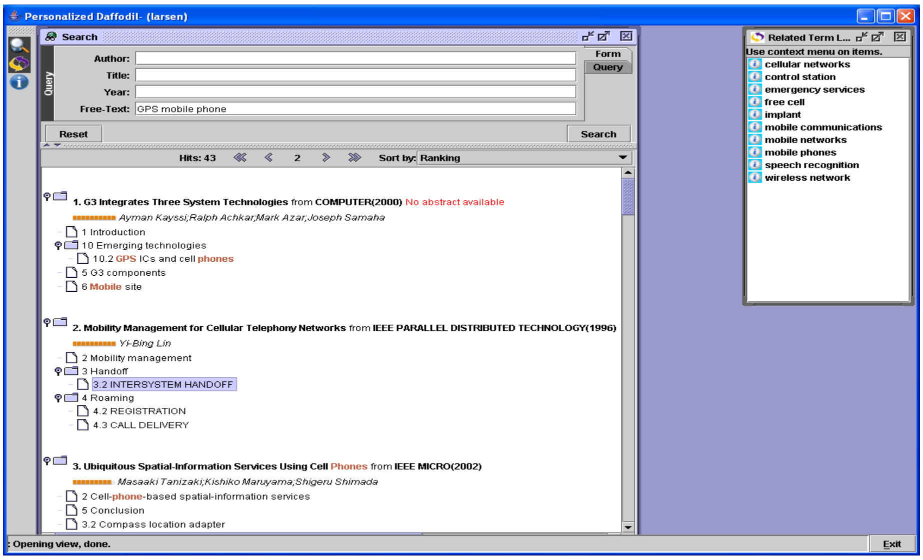
SearchService: gathering results .

- Inex2 found 74 documents

: Waiting for List.

Figure 2: Search in progress

A progress tool (Figure 2 lower left) will open up and show some information about the state of the search, and how many objects are found in the digital library.



Personalized Daffodil- (larsen)

Author:
Title:
Year:
Free-Text: GPS mobile phone

Hits: 43 « ‹ 2 › » Sort by: Ranking

1. G3 Integrates Three System Technologies from COMPUTER(2000) No abstract available
Ayman Kayssi;Ralph Achkar;Mark Azar;Joseph Samaha
- 1 Introduction
- 10 Emerging technologies
  - 10.2 GPS ICs and cell phones
- 5 G3 components
- 6 Mobile site

2. Mobility Management for Cellular Telephony Networks from IEEE PARALLEL DISTRIBUTED TECHNOLOGY(1996)
Yi-Bing Lin
- 2 Mobility management
- 3 Handoff
  - 3.2 INTERSYSTEM HANDOFF
- 4 Roaming
  - 4.2 REGISTRATION
  - 4.3 CALL DELIVERY

3. Ubiquitous Spatial-Information Services Using Cell Phones from IEEE MICRO(2002)
Masaaki Tanizaki;Kishiko Maruyama;Shigeru Shimada
- 2 Cell-phone-based spatial-information services
- 5 Conclusion
- 3.2 Compass location adapter

Use context menu on items.
- cellular networks
- control station
- emergency services
- free cell
- implant
- mobile communications
- mobile networks
- mobile phones
- speech recognition
- wireless network

: Opening view, done.

Figure 3: Result list

3.2 Related Terms

The terms found related by the Related Term Tool to the given query terms are also presented on the right side. The context menu, or dragging, can be used to modify query terms as depicted in Figure 3.

3.3 Result List

The result list is presented in a hierarchy. At the first level whole documents are presented, while elements from the documents are presented at subsequent levels as depicted in Figure 3. The presented elements are the most highly ranked elements from the document. By grouping the result list in this way overlapping results are avoided.

Each first level entry in the result list is preceded by a bar that gives information about its relevance to the query. It also shows the name of the author(s), the title, publication year and journal name of the document. Along with this all terms matching the query are highlighted in a special color.

3.3.1 Sorting

The result list is ordered by the Retrieval Status Value (RSV) by default. It is possible to choose a different sorting order from the "Sort by" drop down list. Possible options are

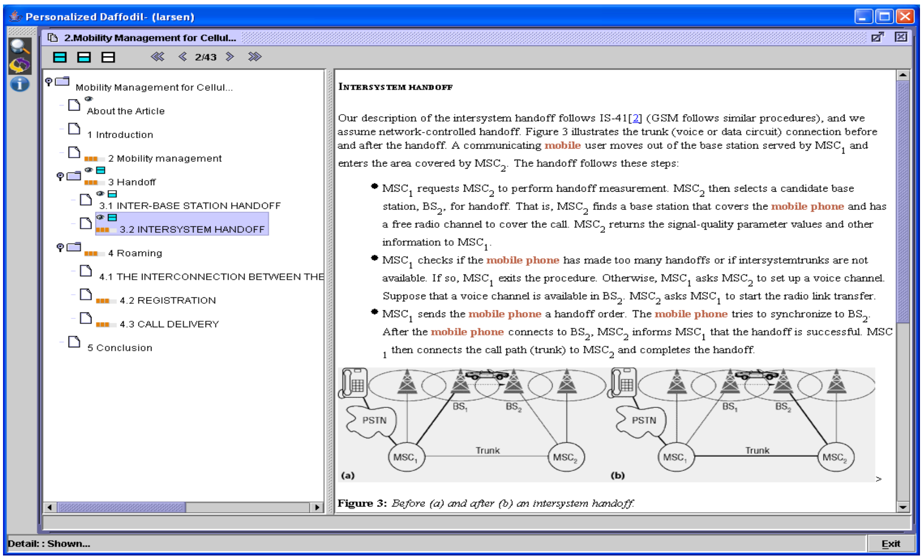
- Authors
- Title
- Year

3.3.2 Navigation

Navigation is possible either by using the scrollbar or by using the arrow icons at the top of the result list. Double left arrow («) moves the selection to the first document in the result list. Single left arrow (<) moves the selection to the previous document in the result list. Single right arrow (>) moves the selection to the next document in the result list. Double right arrow (») moves the selection to the last document in the result list.

3.3.3 Viewing Details (Full text)

The full text of a document or an element can be viewed either by double clicking or by using the context menu in the Result list (Figure 4).



Mobility Management for Cellul...
- About the Article
- 1 Introduction
- 2 Mobility management
- 3 Handoff
  - 3.1 INTER-BASE STATION HANDOFF
  - 3.2 INTERSYSTEM HANDOFF
- 4 Roaming
  - 4.1 THE INTERCONNECTION BETWEEN THE
  - 4.2 REGISTRATION
  - 4.3 CALL DELIVERY
- 5 Conclusion

INTERSYSTEM HANDOFF

Our description of the intersystem handoff follows IS-41[2] (GSM follows similar procedures), and we assume network-controlled handoff. Figure 3 illustrates the trunk (voice or data circuit) connection before and after the handoff. A communicating **mobile** user moves out of the base station served by MSC$_1$ and enters the area covered by MSC$_2$. The handoff follows these steps:

- MSC$_1$ requests MSC$_2$ to perform handoff measurement. MSC$_2$ then selects a candidate base station, BS$_2$, for handoff. That is, MSC$_2$ finds a base station that covers the **mobile phone** and has a free radio channel to cover the call. MSC$_2$ returns the signal-quality parameter values and other information to MSC$_1$.
- MSC$_1$ checks if the **mobile phone** has made too many handoffs or if intersystemtrunks are not available. If so, MSC$_1$ exits the procedure. Otherwise, MSC$_1$ asks MSC$_2$ to set up a voice channel. Suppose that a voice channel is available in BS$_2$. MSC$_2$ asks MSC$_1$ to start the radio link transfer.
- MSC$_1$ sends the **mobile phone** a handoff order. The **mobile phone** tries to synchronize to BS$_2$. After the **mobile phone** connects to BS$_2$, MSC$_2$ informs MSC$_1$ that the handoff is successful. MSC $_1$ then connects the call path (trunk) to MSC$_2$ and completes the handoff.

Figure 3: *Before (a) and after (b) an intersystem handoff.*

Figure 4: Document details

The Detail window is split into two sub windows and a top bar. The left window contains the table of contents of the document. Here the RSV of any retrieved elements within the document are indicated by a RSV bar similar to those in the Result list. The right window displays the details of the currently selected document element.

In order to move back and forth within the document, single arrow buttons can be used while double arrow buttons can be used to move to the next and previous enteries in the result list. Along with the navigational buttons, the top toolbar also contains icons for giving a relevance assessment of the currently viewed element. The possible values are Relevant (fully coloured), Partially Relevant (partially coloured) and Not Relevant (no colours). For a selected element, the assessment is given by clicking one of these three icons. Already viewed items and those which have been assessed are iconzed by a small eye and feedback icon respectively as shown in Figure 4.

4 Installation

4.1 Requirements

Install Java 1.4.2 http://java.sun.com/j2se/1.4.2/download.html] on your machine.

4.2 iTrack Daffodil

Then use you browser to fetch the following URL http://www.is.informatik.uni-duisburg.de/projects/daffodil/InexDaffodil.jnlp

This will download the Daffodil application to your machine and begin installation. During installation you will be asked to accept a number of certificates, and if you want desktop integration. Choosing the latter will provide you with a Daffodil desktop icon providing easy access to the system. Before you log on the first time please create a personilazation file as detailed below.

If you are behind a Firewall or a Proxy, please refer to the information at http://www.is.informatik.uni-duisburg.de/wiki/index.php/FAQs#We_have_a_firewall_and_I_cannot_connect_to_Daffodil._But_we_do_have_a_proxy.

4.3 Personalization

Daffodil needs to be personalized with setting specific to the interactive track. Create a file called "daffuser.properties" in your home dir, and include in this file the settings found at: http://www.is.informatik.uni-duisburg.de/wiki/index.php/Daffuser.properties_for_inex_users

Please note that this will personalize the system for the user currently logged onto your machine. If you want to log om the machine with a different login, you need to copy the daffuser.properties file to the home dir of this users account.

The home dir for windows users is typically the same as the folder containing the "My Documents" folder, but you may wish to check with your system administrator if you are unsure. For example:

C:\Documents and Settings\larsen, for the user larsen

5 Logging

Please refer to the Track Guidelines for details on the logging procedures. A log viewer is available at http://inex.is.informatik.uni-duisburg.de/2005/inex05/LogViewer/testuser.jsp

Logging guidelines

In this year's INEX2005 interactive track local systems can be tested against a baseline system provided by the track organisers (The B task – see the Interactive Track Guidelines). The purpose of these guidelines is to ensure that the logs of local systems are comparable to that of the baseline system. In addition we hope that the experiences gained from these tests can inform the development of a Standardized XML schema for Digital Library Logging as initiated by DELOS[1].

The log data will comprise of one *session* for each topic the test person searches. Each session will be recorded in a log that records the *events* in the session, both the actions performed by the test person and the responses from the system. Table 1 below contains a list of suggestions of the type of events that can be logged for each session; items in **bold** are suggested as a minimum set of events to log.

An example of the logs files from this year's baseline system is given in the Appendix below. Please note that the local logs do not have to follow this exact format, but should be in a format that can be transformed into a similar format or a common XML format.

Table 1. Events to log for each session. Suggested **minimum** requirements in **bold**.

For each session • **Session ID** • **Login time** • **Logout time** • **Test Person ID** • **Collection (IEEE, LP)**
For each event • Begin and End **Time stamps** for every action (**year-month-day ; hh:mm:ss**:ms) • **Session ID** (Session ID) added to every action • **Eventtype,** as stated below
Types of events logged • **Submitted queries** (query type (e.g. free-text, use of fields, NEXI, Bricks) ; **Exact query terms as inputted by the test person**) • **Query results** (**number of retrieved elements/documents** ; rank/RSV ; DocID/elementID for all retrieved documents) • **Any viewed hits** and how the user got there (**DocID/elementID** ; Directly from hitlist/From browsing) • Any **use of interface functionalities** (e.g., Next 10 hits, hit page 7, Next hit. Sort results etc.) • Any **browsing within documents** (elementID ; where the user came from) • **Relevance assessments** (docID/elementID ; assessment)

[1] see http://www.is.informatik.uni-duisburg.de/wiki/index.php/JPA_2_-_WP7

Appendix

Sample log from the 2005 baseline system with comments.

Walkthrough
Various data is contained in the section element. This followed by the events caused by the test person.

1. The test person logs in (or rather the experimenter does it for her to ensure that everything is done correctly, e.g., selection of task, collection and topic)
2. She searches in a free-text field supporting Boolean queries. After a couple of seconds 69 elements are retrieved
3. She sorts the results by year
4. She sort the results by RSV (Retrieval Status Value)
5. She chooses to look at the full-text (‘detailquery’) of the element ranked first (a subsection)
6. She assesses the element with the value ‘2’
7. She browses to the previous subsection in the same document by pressing the ‘prev’ button
8. After pressing ‘prev’ again the containing section is displayed
9. She assesses this section with the value ‘1’
10. She then chooses the view the abstract from a different document from the result list (the system displays the whole front matter)
11. She logs out.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE session SYSTEM "session.dtd">
  <session id="c2.134a7d8:10677fcc8a8:-7f3f@UA"
    userid="unidu_searcher_001"
    timestamp="2005/09/23 09:45:31:99"
    duration="6 m 7 s 970 ms "
    colletion="IEEE"
    task="A"
    topicid="IEEE-G-003">

    <events>

    <!-- This is an example log, converted and simplified to present the
    idea that how a log should look like so that we can do comparisons
    across systems -->
    <!-- The log consists of events with a set of minimal recommended
    elements, and a system-specific element, where everybody can put
    the real system log, that fires an event -->

      <event id="5a.-2d3babc:10681f1aaf4:-7ffc@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="login"
      beginTimestamp="2005/09/23 09:45:33:334"
      endTimestamp="2005/09/23 09:45:34:000">
      <!-- This is the actual login event by the user -->
      <!-- Minimal needed info: userid, timestamp -->
       <userid>unidu_searcher_001</userid>
    </event>
      <event id="6a.-2d3babc:10681f1aaf4:-7ffc@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="search"
      beginTimestamp="2005/09/23 09:45:33:334"
      endTimestamp="2005/09/23 09:47:01:737">
      <!-- This is a normal search for metadata -->
      <!-- Minimal needed info: querytype, query, timestamp -->
      <!-- Querytype has somehow to be defined: e.g. boolean, NEXI,
      BRICKS, etc. -->
      <!-- The query should be given exactly as inputted by the user -->
       <querytype>boolean,free-text</querytype>
       <query>GPS, mobile phones</query>
       <numberOfFoundDocs>69</numberOfFoundDocs>
       <results>
        <result rank="1" file="co/2000/rx107" path="article[1]/bdy[1]/sec[10]
        /ss1[2]" rsv="0.16628836582695"/>
       <result rank="2" file="co/1998/ry053" path="/article[1]" rsv="
       0.164833086254384"/>
       <result rank="3" file="co/2000/rx107" path="/article[1]/bdy[1]/sec[10]"
       rsv="0.16032668116438"/>
       <result rank="4" file="tc/1995/t1169" path="/article[1]/bdy[1]/sec[3]"
       rsv="0.159319299371904"/>
       <result rank="5" file="td/2000/l0444" path="/article[1]/bdy[1]/sec[4]"
       rsv="0.158636569166586"/>
         <!—Only 5 are shown here, but all retrieved elements should be
         given in the log. This will enable us to study characteristics
         of both what the user chose and what they didn't choose -->
       </results>
    </event>

      <event id="62.-2d3babc:10681f1aaf4:-7ff4@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="filter"
      beginTimestamp="2005/09/23 09:47:45:42"
      endTimestamp="2005/09/23 09:47:45:42" >
       <type>sort</type>
       <attribute>year</attribute>
       <predicate>greater-then</predicate>

    </event>

    <event id="65.-2d3babc:10681f1aaf4:-7ff1@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="filter"
      beginTimestamp="2005/09/23 09:48:11:962"
      endTimestamp="2005/09/23 09:48:11:962">
       <type>sort</type>
       <attribute>rsv</attribute>
       <predicate>greater-then</predicate>

    </event>

    <event id="66.-2d3babc:10681f1aaf4:-7ff0@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="detailquery"
      timestamp="2005/09/23 09:48:33:570" >
       <coming-from>resultlist</coming-from>
       <docid>2602</docid>
       <file>co/2000/rx107</file>
       <path>/article[1]/bdy[1]/sec[10]/ss1[2]</path>
       <rank>1</rank>
    </event>
```

```
    <event id="6c.-2d3babc:10681f1aaf4:-7fea@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="relevance-assessment"
      timestamp="2005/09/23 09:49:16:237" >
       <docid>2602</docid>
       <file>co/2000/rx107</file>
       <path>/article[1]/bdy[1]/sec[10]/ss1[2]</path>
       <assessment>2</assessment>

      </event>

      <event id="6f.-2d3babc:10681f1aaf4:-7fe7@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="detailbrowsing"
      timestamp="2005/09/23 09:49:31:618" >
       <coming-from>prev</coming-from>
       <docid>2602</docid>
       <file>co/2000/rx107</file>
       <path>/article[1]/bdy[1]/sec[10]/ss1[1]</path>

    </event>

    <event id="72.-2d3babc:10681f1aaf4:-7fe4@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="detailbrowsing"
      timestamp="2005/09/23 09:50:01:808" >
       <coming-from>prev</coming-from>
       <docid>2602</docid>
       <file>co/2000/rx107</file>
       <path>/article[1]/bdy[1]/sec[10]</path>

    </event>

    <event id="74.-2d3babc:10681f1aaf4:-7fe2@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="relevance-assessment"
      timestamp="2005/09/23 09:50:06:692" >
       <docid>2602</docid>
       <file>co/2000/rx107</file>
       <path>/article[1]/bdy[4]/sec[10]</path>
       <assessment>1</assessment>

    </event>

    <event id="76.-2d3babc:10681f1aaf4:-7fe0@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="abstractquery"
      timestamp="2005/09/23 09:50:21:264" >
       <coming-from>resultlist</coming-from>
       <rank>10</rank>
       <docid>6679</docid>
       <file>pd/1996/p4065</file>
       <path>/article[1]/fm[1]</path>

    </event>

    <event id="86.-2d3babc:10681f1aaf4:-7fd0@ea127.0.0.1:5678"
      sessionid="c2.134a7d8:10677fcc8a8:-7f3f@UA"
      eventType="logout"
      timestamp="2005/09/23 09:51:39:69" >

    </event>
  </events>
</session>
```

Introduction to the experiment

Thank you for agreeing to participate in the experiment.

All the collected data will be treated as confidential and it will not be possible to identify you as person with the data after the experiment has ended.

The purpose of this document is to inform you about how to prepare for the experiment.

The overall goal of the experiment is to investigate the performance of a new system for information retrieval. In response to your queries the system presents articles to you, and also attempts to indicate which parts of the articles that may be most closely related to your search.

On the day of the experiment, you will be asked to carry out a number of search tasks in the system. Two tasks will be supplied by the experimenter. In addition, we would like you to search for information on a topic relating to an interest of your own. This could, for instance, be related to your work or a topic of general interest to you. The collection you will be searching consists of articles from computer science journals covering a broad range of topics in fields related to computers and computing, and the topic you search for should therefore relate to this subject area. In order to maximise the chances that your own topic is covered by the collection, please email *two* topics of interest to you to the experimenter before the day of the experiment. The experimenter will then do a preliminary search of the collection to examine their coverage in the collection, and inform you which one to search on when the experiment starts.

Please record your topics in the forms found in **Appendix B**. The description of a topic should include:

A. A short description of *what* you are looking for, i.e., what you will ask the system to find.

B. A description of the motivation for the topic, i.e., *why* you are looking for this information or what *problem* you intend to solve with the information. Such details of the *context* of your topic are useful in giving the experimenter a better understanding of your topic.

C. A reflection on the ideal answer you would like to be presented with, i.e. what a perfect answer would look like to satisfy your topic.

An example of such a topic is given below. An overview of the collection can be found in **Appendix A** which you may skim read to help you in formulating your topics.

Please email the two topics to your experimenter by the time given in the email which contained this document.

Topic example

A. What are you looking for (*what* do you want the system to find)?

> *I want information about conferences and workshops in the multimedia field. This may be found in articles titled "call for papers" or "upcoming events"*

B. What is the motivation of the topic (*why* are you looking for this, what *problem* can be solved with the information, and in what *context* did the problem arise)?

> *I am starting to organize a workshop on multimedia and I am collecting information about other multimedia events (mainly conferences and workshops) in order to decide the most appropriate topics and dates for my workshop.*

C. What would an ideal answer look like (what should a perfect answer contain to solve your problem)?

> *Relevant items mention conferences or workshops in a multimedia related field with their topics of interest and dates. Items that only mention the name of the conference are somewhat relevant because I can look for further information myself. Items mentioning conferences outside the multimedia field or information about multimedia are not relevant*

Appendix A – overview of the document collection

The document collection that you can search contains articles published by the IEEE (Institute of Electrical and Electronics Engineers), and contains material published between 1995 and 2004. There are about 14,000 articles in the collection.

The broad area covered by the articles is Computer Science. The material is intended both for a computer science and a non-specialist, more general audience.

One part of the collection contains publications designed to inform readers about trends and news in a number of broad fields related to computers and computing (e.g. software, graphics, Internet computing, applications of computing in science and engineering, intelligent systems, distributed systems, microcomputer & microprocessor design, etc.). Apart from technical issues, these magazines also touch on topics such as the history of computing. These publications typically cover material that practitioners, researchers, and managers can use to keep up to date with current issues and developments in the wider field of computing.

The other part contains publications that are scholarly journals designed to inform readers on the state of the art in a number of specialized fields related to computers and computing. The areas covered in these journals range from software engineering, knowledge and data engineering and mobile computing, to intelligent systems and AI, graphics and visualisation and computational biology and bioinformatics.

The collection can only be searched in its entirety, that means, you can not specify which of the two parts you wish to search. The results returned to your search can contain information from any of the publications included in the collection that best match your search.

Appendix B – forms for recording topics

Own topic no. 1

A. What are you looking for (*what* do you want the system to find)?
B. What is the motivation of the topic (*why* are you looking for this, what *problem* can be solved with the information, and in what *context* did the problem arise)?
C. What would an ideal answer look like (what should a perfect answer contain to solve your problem)?

Own topic no. 2

A. What are you looking for (*what* do you want the system to find)?
B. What is the motivation of the topic (*why* are you looking for this, what *problem* can be solved with the information, and in what *context* did the problem arise)?
C. What would an ideal answer look like (what should a perfect answer contain to solve your problem)?

Instructions to searchers

Thank you for agreeing to participate in the experiment. All the collected data will be treated as confidential and it will not be possible to identify you as person with the data after the experiment has ended.

The purpose of this document is to give you information about the experiment and your role in it.

You will be asked to search on a number of tasks on a single search system. You will first be given a practice run with the system. In this run you will be given a training task. During this practice run you can ask any questions about any of the features of the system. The collection you will be searching consists of full text articles from the journals published by the IEEE Computer Society. The broad topic area is computer science with a focus on hardware and software development. The time span covered by the collection is 1995 to 2004.

In this experiment you will be asked to search on three tasks: two from two task categories as well as one of your own tasks. The experimenter has chosen that of your tasks that best matches the content of the collection. For the two other tasks, you will have a choice of one from three available tasks. For each task you can take a maximum of 20 minutes. If you feel you have completed the task before the 20 minute period you can notify the experimenter and the session will be stopped.

The overall goal of the experiment is to investigate the performance of a new system for information retrieval. In response to your queries the system presents articles to you, and also attempts to indicate which parts of the articles may be most closely related to your search. Feel free to browse the results returned by the system and to submit as many different queries to the system as you like for each task. It is an important aspect of the experiment to collect your assessments of the relevance of the information presented to you by the system. To help us collect this data, please select an assessment score *for each viewed piece of information*. More information about the system and its features can be found below.

During the experiment you will also be asked to complete several additional questionnaires:

- Before the experiment (measuring search experience)
- Before each task (measuring task familiarity)
- After each task
- After the experiment (collecting experiment feedback)

The experiment will conclude with an interview.

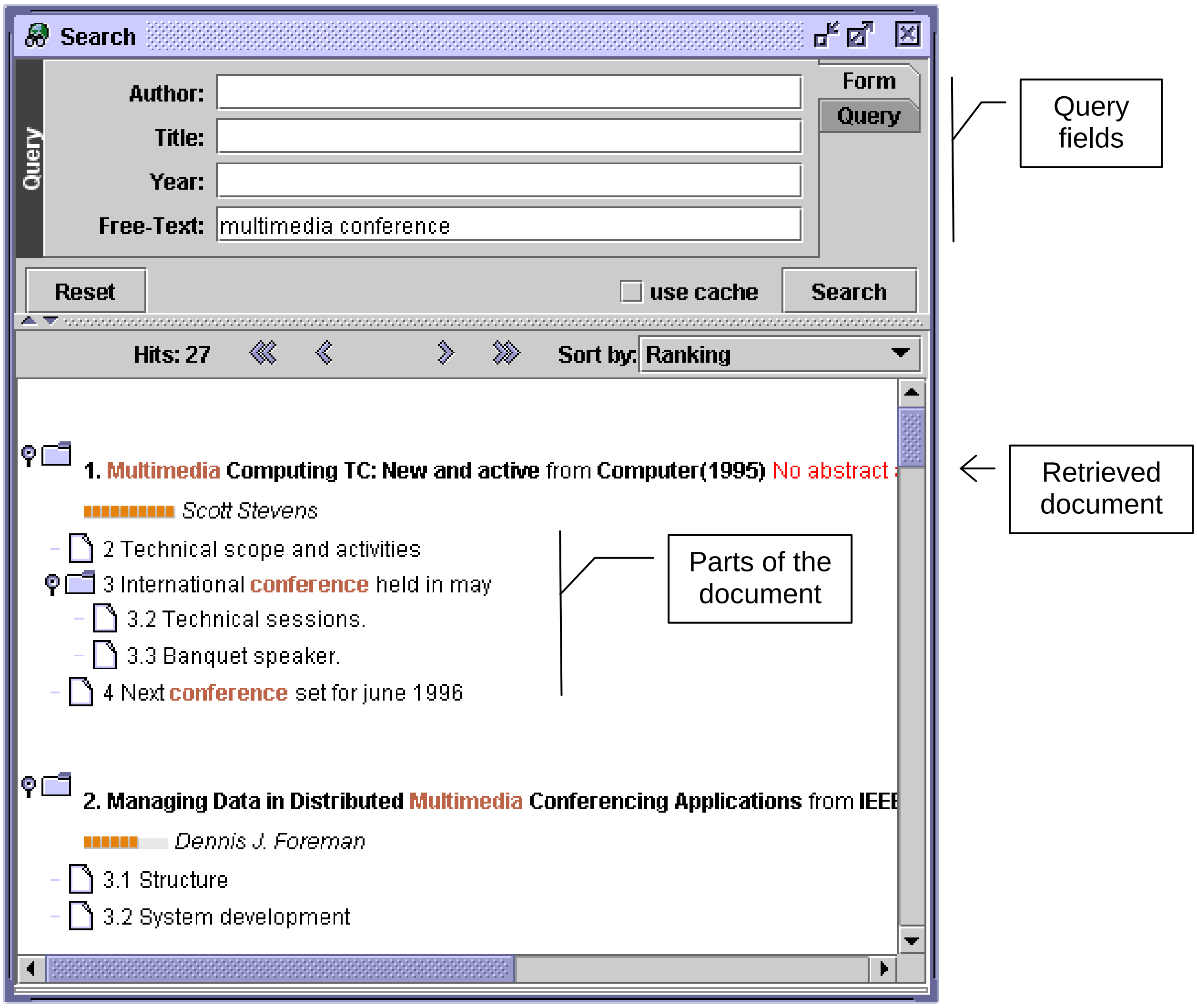
Thank you for your help!

System features

The experimenter will log into the system for you; once for each task.

After login you will see a form for query formulation. A number of query fields are offered: Author, Title, Year and Free-text. The Free-text field allows searching in the full text of the documents. Holding your mouse over a field offers search tips for that field.

After entering a query and pressing “Search” a search progress indicator will inform you about the number documents found. A related term list will also appear, suggesting alternative search terms. The results are presented as documents and in some cases the system also attempts to indicate which parts of the articles that may be most closely related to your search.



Author:
Title:
Year:
Free-Text: multimedia conference

□ use cache

Hits: 27 Sort by: Ranking

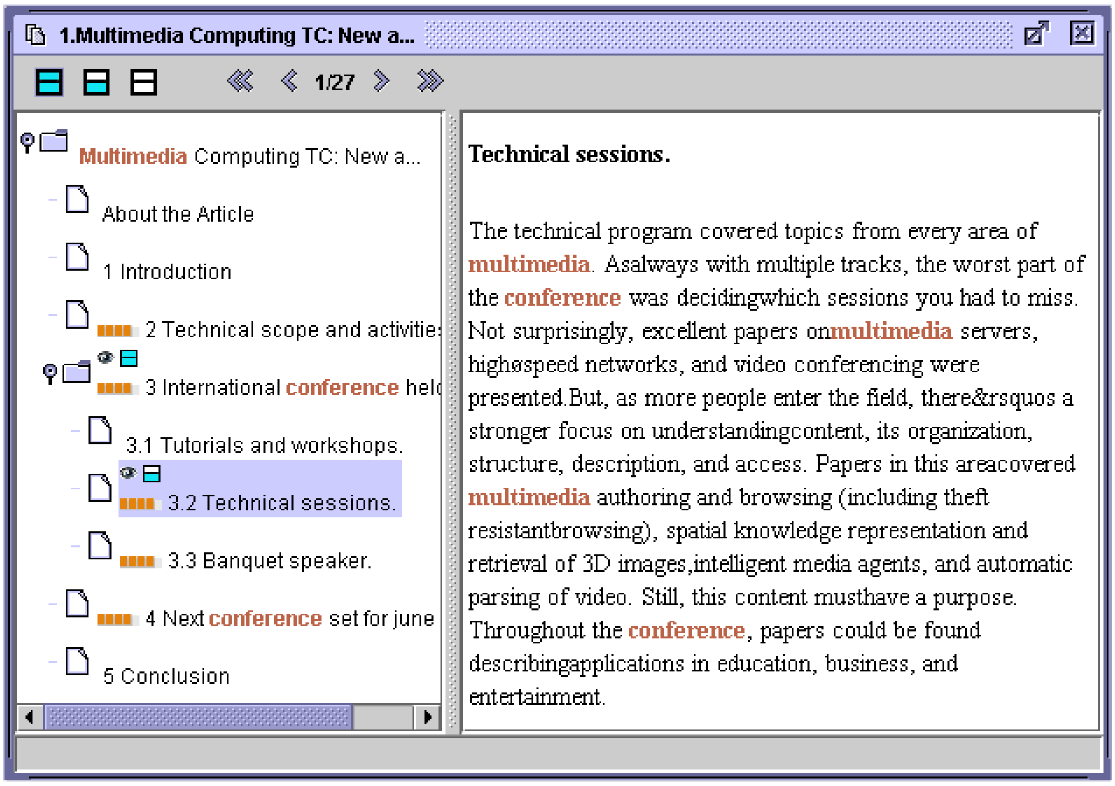
1. Multimedia Computing TC: New and active from Computer(1995) No abstract
Scott Stevens
- 2 Technical scope and activities
- 3 International conference held in may
  - 3.2 Technical sessions.
  - 3.3 Banquet speaker.
- 4 Next conference set for june 1996

2. Managing Data in Distributed Multimedia Conferencing Applications from IEEE
Dennis J. Foreman
- 3.1 Structure
- 3.2 System development

Double-clicking a document or part of it will open the part in a new window. This is split in two panes: one with a Table of Contents of the whole document, and one with the selected part of the document.

Assessment scores. Click to assess the current part of the document

Click here to close the document



1.Multimedia Computing TC: New a...

1/27

- Multimedia Computing TC: New a...
  - About the Article
  - 1 Introduction
  - 2 Technical scope and activitie:
  - 3 International conference helc
    - 3.1 Tutorials and workshops.
    - 3.2 Technical sessions.
    - 3.3 Banquet speaker.
  - 4 Next conference set for june
  - 5 Conclusion

Technical sessions.

The technical program covered topics from every area of multimedia. Asalways with multiple tracks, the worst part of the conference was decidingwhich sessions you had to miss. Not surprisingly, excellent papers onmultimedia servers, highøspeed networks, and video conferencing were presented.But, as more people enter the field, there&rsquos a stronger focus on understandingcontent, its organization, structure, description, and access. Papers in this areacovered multimedia authoring and browsing (including theft resistantbrowsing), spatial knowledge representation and retrieval of 3D images,intelligent media agents, and automatic parsing of video. Still, this content musthave a purpose. Throughout the conference, papers could be found describingapplications in education, business, and entertainment.

Currently viewed part of the document. Click to view another part.

Any part of the document which you have already viewed is indicated with a small eye (👁). Please remember select an assessment score *for each viewed piece of information* that reflect the usefulness of the seen information in solving the task. Three different scores are available at the right-hand top of the page:

▣ - Relevant

⊟ - Partially Relevant

□ - Not Relevant

The symbols are also used in the in the Table of Contents for already assessed parts.

The currently viewed part of the document is highlighted in the Table of Contents. Other parts of the document can be viewed by clicking a different heading in the Table of Contents. Once you feel that there is no more relevant information in the document, please close it by pressing ☒ - this will return you to the result list where you may choose to examine another document, or to perform a new search.

Please do NOT press the “Exit” button at any time as this will terminate your session. Exit

Training task

Task ID: Training

You have become involved in the organization of a workshop on multimedia at your department. You would like to collect information about other multimedia events (conferences, workshops, etc.) in order to decide the most appropriate topics and dates for the workshop at your department.

Find, for instance, pieces of information that include "Call for papers" or "Upcoming Events" in multimedia related research areas.

Please select *one* of the following tasks:

Task ID: G1

New anti-terrorism laws allow intelligence agencies like the FBI (Federal Bureau of Investigation) and CIA (Central Intelligence Agency) to monitor computer communications to spot suspected criminals and terrorists. You would like to find information about how this affects your own and other people's privacy and to know what concerns have been raised.

Find, for instance, information that discusses the Carnivore or Echelon projects or other similar surveillance of computer communication.

Task ID: G2

Your department has produced a Linux-program and it is being discussed whether to release it under a public license such as GNU or GPL (General Public License). Therefore, you have been asked to find information about the implications of releasing the code under a public license as an open source program.

Find, for instance, information that discusses different licensing schemes or articles about the impact of open source programs.

Task ID: G3

Video games are being played by an ever increasing number of people of all ages, and the game industry is becoming a major economic player. You would therefore like to find non-technical information about how video games have affected people's lives as well as how the games have changed the entertainment industry.

Find, for instance, information discussing the concerns that playing video games may lead to a rise in violent behaviour, or information about the effect of video games on the film industry.

Please select *one* of the following tasks:

Task ID: C1

One of your friends has recently bought a small handheld Global Positioning System (GPS) unit, and the possibilities offered by this technology have caught your interest. You would like to explore new killer applications for mobile devices. Therefore, you are looking for examples and descriptions of applications that use GPS, for devices such as mobile phones, PDAs (Personal Desktop Assistants) and other wireless and mobile devices.

Find, for instance, information that discusses examples of how applications that use GPS can be used to accomplish new tasks or provide new services.

Task ID: C2

In your daily work you sign on to a range of different systems both locally and remotely. On many of them you have different user IDs and different passwords, and you find it annoying to have to verify your identity again and again. In addition, you find it demanding to maintain all these IDs and passwords and to keep them secure.
You have heard about LDAP (Lightweight Directory Access Protocol) and other single sign-on procedures, and wish to learn more about them to assess the potentials for creating a single sign-on procedure for your local network (with both Unix, Linux, PC and Mac platforms).

Find, for instance, information that discusses single sign-on procedures, or state of the art user-authentication methods.

Task ID: C3

Data security and authenticity is an important issue at your work place. One approach to ensure data authenticity is the so-called “steganography” where data is embedded in various media files like images, sound files, video files and so on. A commonly used data embedding technique is Watermarking where data can be effectively hidden in a file without the changes being visible to the common person. You want to learn more about Watermarking as a technique for data embedding that will enable you to verify the authenticity of a file.

Find, for instance, information that discusses the use of Watermarking techniques to hide information that will allow later validation of a files authenticity.

To be filled in by the experimenter

Participating site:	*Searcher ID:*
Rotation: 1 2 3 4 5 6	

Before-experiment Questionnaire

1. Initials:

2. Age:

3. Gender (Please circle)

   Male / Female

4. What is your first language?

5. Current occupation:

6. What university degrees, minor or majors do you have or plan to take in the near future (if any)?

   Degree/major Year

   ____ _______

   ____ _______

   ____ _______

   ____ _______

7. Have you participated in previous on-line searching studies, as

   Experimenter □ Yes Test person □ Yes
   □ No □ No

8. Overall, how many years have you been doing on-line searching? ________ years

More questions on the next page →

Please, circle the number closest to your experience:

How much experience have you had	**No experience**		**Some experience**		**A great deal of experience**
9. Searching on computerised library catalogues either locally (e.g. your library) or remotely (e.g. Library of Congress)	1	2	3	4	5
10. Searching on digital libraries of scientific articles (e.g. ACM Digital Library)	1	2	3	4	5
11. Searching on WWW search engines	1	2	3	4	5
12. Searching on other systems, please specify the system(s) on the lines below: ____________________ ____________________	1	2	3	4	5
13. Reading or accessing journals and magazines published by the Institute of Electrical and Electronics Engineers (IEEE)	1	2	3	4	5

Please circle the number most appropriate to your searching behaviour:

	Never	**Once or twice a year**	**Once or twice a month**	**Once or twice a week**	**One or more times a day**
14. How often do you perform a search on any kind of system?	1	2	3	4	5

Please circle the number that best indicates to what extent you agree with the following statement:

	Strongly disagree	**Disagree**	**Neutral**	**Agree**	**Strongly agree**
15. I enjoy carrying out information searches	1	2	3	4	5

To be filled in by the experimenter

Participating site:	*Searcher ID:*
Rotation: 1 2 3 4 5 6	*Task: G1 G2 G3 C1 C2 C3 Own*

Before-each-task Questionnaire

Please circle the number that best indicates your perception of the task you have chosen:

	Not at all		**Somewhat**		**Extremely**
1. Are you familiar with the topic of the given task?	1	2	3	4	5
2. Do you think it will be easy for you to search on this task?	1	2	3	4	5

Please circle the number that best indicates your perception of the task you have chosen:

	Long, e.g., a whole article		**Medium, e.g., a section in an article**		**Short, e.g., a single paragraph**
3. How large would you expect an ideal answer to be?	1	2	3	4	5

4. Do you expect that a single answer/piece of information/..? will be enough for your task?

- ❑ Yes, the answer can probably be found within a single piece of information.
- ❑ No, I expect that I will have to combine pieces of information from many sources to solve the task.

To be filled in by the experimenter

Participating site:	*Searcher ID:*
Rotation: 1 2 3 4 5 6	*Task: G1 G2 G3 C1 C2 C3 Own*

After-each-task Questionnaire

Please circle the number which best corresponds to your opinion:

	Not at all		**Somewhat**		**Extremely**
1. Was it easy to get started on this search?	1	2	3	4	5
2. Was it easy to do the search on the given task?	1	2	3	4	5
3. Are you satisfied with your search results?	1	2	3	4	5
4. Do you feel that the task has been fulfilled?	1	2	3	4	5
5. Do you feel that the search task was clear?	1	2	3	4	5
6. Was the search task interesting to you?	1	2	3	4	5
7. Did you know a lot about the topic of the task in advance?	1	2	3	4	5
8. Did you have enough time to do an effective search?	1	2	3	4	5

Please circle the number which best corresponds to the searching experience you just had:

	Not at all		**Somewhat**		**Extremely**
9. How well did the system support you in this task?	1	2	3	4	5

More questions on the next page →

Please circle the number which best corresponds to your views on the information presented to you by the system:

	Not at all		**Somewhat**		**Extremely**
10. On average, how relevant to the search task was the information presented to you?	1	2	3	4	5
11. Did you in general find the presentation in the result list useful?	1	2	3	4	5
12. Did you find the parts of the documents in the result list useful?	1	2	3	4	5
13. Did you find the Table of Contents in the Full Text view useful?	1	2	3	4	5

14. Was a single answer/piece of information/..? enough to solve your task?

- ❑ Yes, the task could be solved with a single piece of information.
- ❑ No, I had to combine pieces of information from many sources to solve the task.

15. In what ways (if any) did you find the system interface useful in this task?

16. In what ways (if any) did you find the system interface *not* useful in this task?

Please continue overleaf if necessary →

To be filled in by the experimenter

Participating site: *Searcher ID:*

Rotation: 1 2 3 4 5 6

Post-experiment Questionnaire

1. Please rank the three tasks you have worked in relation to their difficulty:

   - Most difficult:
   - Middle difficult:
   - Least difficult:

Please circle the number better corresponding to your view on the questions:

	Not at all		**Somewhat**		**Extremely**
2. How understandable were the tasks?	1	2	3	4	5
3. To what extent did you find the tasks similar to other searching tasks that you typically perform?	1	2	3	4	5
4. How easy was it to learn to use the system?	1	2	3	4	5
5. How easy was it to use the system?	1	2	3	4	5
6. How well did you understand how to use the system?	1	2	3	4	5

More questions on the next page →

7. What did you like about the search system?

8. What did you dislike about the search system?

9. Do you have any general comments?

Thank you for your help!!!

Please continue overleaf if necessary →

Guidelines for Post-experiment Interviews

General information

The type of interview will be semi-structured. General guidelines and a "script" for the interview will be provided in this document; experimenters are also to probe the searchers for further clarifications/information where appropriate, depending on the response.

Recording interviews

The minimum requirement will be to write down the answers to questions as fully as possible. The interview transcripts will need to be sent back to the experimenters in electronic format, so experimenters will be responsible for putting the responses in appropriate electronic format. The track organisers will provide appropriate template spreadsheets to make data input as convenient and uniform as possible. To facilitate this, please use the numbers provided in the script below (e.g. b-1i) when recording the answers in your transcripts. These numbers will be used for input in the provided spreadsheets.

Preferably, an audio recording of each interview should be performed, and the transcript made at a later time with no risk of loss of information from the interview.

Interview script

In general, we are trying to investigate the following issues:

a. Interface issues
b. Issues related to the tasks
c. Retrieval granularity (Document vs. element retrieval, Passage vs. element retrieval)
d. Applications for element retrieval
e. Other issues

a. Interface Issues

1. Did you find the ranked results presentation useful?
2. Do you have any comments about the way we present documents in the ranked list (i.e., present a document and its top ranked elements together)?
3. Did you like the use of the table of contents in the Document View?
    i. If yes, why did you find it useful?
4. Are there any other features that you would like to see incorporated at the interface?
    i. If yes, why would you find them useful?

b. Issues related to the tasks

1. For each of the simulated tasks you searched (note which):
   i. Did you find it realistic?
   ii. Did you find it interesting?
   iii. Why/why not?
   iv. Did the search tip help (the last sentence of the simulated task: “Find, for instance, …”)?
   v. How did you use the search tip when searching?
   vi. How did you use the main part of the tasks description?
2. For your own task:
   i. Did you find it difficult to formulate a task?
   ii. Did you need more information to do so, or access to the collection?
   iii. After having worked with your own task, is there any information you would like to add to the description?

c. Retrieval granularity

For the task that you were asked to formulate yourselves:

1. What was generally the length of the best answers? Document / section / paragraph level? Something else?
   i. If you thought elements were more useful, did you normally have to combine multiple elements to find solutions to tasks?
2. Did you find that the task could be answered from the information contained within a single document?
3. Did you find the structural breakdown of documents useful?
4. Did you think that the breakdown to structural elements (sections etc.) corresponded to items containing answers to your tasks?
   i. Did you think that answers were mainly contained within one or more elements?
   ii. Did you think that answers typically spanned over more than one element (or element part)?
   iii. Did you think that answers typically were included in a smaller part of a single element?
   iv. Did you think that a breakdown of documents based on topics would be preferable to the structural one?

d. Applications of element retrieval

1. Did you like the idea of a retrieval system that takes into account the structural breakdown?
2. Did you find it well suited to your own task?
3. Do you think that such a retrieval system can have applications to other tasks?
   i. If yes, can you name some?

e. Other issues

1. Any other expectations that you may have from such a system?
2. Are there any other comments that you want to make?

Mining XML documents
Bridging the gap between Machine Learning and Information Retrieval

Joint Challenge

In cooperation with the INEX Initiative (INitiative for the Evaluation of XML Retrieval) from the DELOS Network of Excellence and the PASCAL Network of Excellence.

Web site: http://xmlmining.lip6.fr

Contact: xmlmining@lip6.fr

1 Overview

The objective of the challenge is to develop machine learning methods for structured data mining and to evaluate these methods for XML document mining tasks. The challenge is focused on classification and clustering for XML documents. Datasets coming from different XML collections and covering a variety of classification and clustering situations are provided to the participants. The challenge will run in two rounds: the first round corresponds to the present call and will run between July and November 2005 with results presented at the INEX workshop at the end of November 2005 (November 28-30). A second round will be organized between January and March 2006 with a presentation of the participants' results at a later workshop (e.g. the Pascal challenge workshop on April 2006 in Venice, or a joint workshop INEX - Pascal). Participation is open to all.

2 General description of the challenge

2.1 Context and Motivations

This challenge aims at gathering machine learning (ML), Information Retrieval (IR) and XML researchers in order to:

- Define the new key problems for structured data mining with ML techniques.
- Identify and assess the potential of ML techniques for dealing with generic ML tasks such as classification and clustering in the structured domain.
- Build XML document collections, define evaluation methodologies and develop software which will be used for the evaluation of structured classification and clustering.
- Compare existing methods on different structured datasets corresponding to actual XML document collections.

Learning in structured domains is a recent research direction in the ML field. Structured data do appear in many different domains. For this challenge, we will focus on structured textual data and more precisely on XML document collections. There already exists a very active community in the IR/ XML domain which has started to work on XML search engines and XML textual data. This community is mainly organized since 2002 around the INEX initiative (INitiative for the Evaluation of XML Retrieval) which is funded by the DELOS network of excellence on Digital Libraries. INEX has already gathered large XML textual collections. The challenge is a joint event co-organized by the two networks (PASCAL and DELOS).

2.2 Key issues to be investigated for structured document classification and clustering

Among the many open problems for handling structured data, we will focus in this challenge on the two generic tasks of *classification* and *clustering*. The goal of the challenge is therefore to explore algorithmic, theoretical and practical issues regarding the classification and clustering of structured data. The challenge is also aimed as a discussion forum for defining new ML problems specific to XML documents.
Dealing with XML document collections is a particularly challenging task for ML and IR. XML documents are defined by their logical structure and their content (hence the name semi-structured data). Both types of information should be addressed in order to effectively mine XML documents.

Structure document information is described through an ordered labelled tree where labels correspond to XML tags which may or may not carry semantic information. In document collections, content information is composed of text and images, but for this challenge, only the textual part is considered. The textual content is usually contextually dependent of the logical structure XML documents usually come in large collections; this means that the scalability issue is fundamental here.
Depending on the application context, it may be relevant to consider the structure information alone or both structure and content of the documents.
We will then consider two different sets of tasks corresponding respectively to *Structure Only* and *Structure and Content* classification and clustering.

The challenge will explore classification and clustering tasks for XML data through a series of tests performed on collections with different characteristics, like their size, content, complexity or structural organization, in order to test the different methods in different applicative contexts. These collections will be generated using existing XML collections, (see the collections description). For both classification and clustering, the goal will be to identify either a known the information source or the thematic of documents. Either structure, content or both might be relevant for these tasks.

Structure Only tasks

In the following, *Structure* Only tasks correspond to classification or clustering using the structural description of XML documents alone, i.e. the XML ordered labelled tree providing the relations between the document elements. The input space in this case corresponds to the tag alphabet which is usually of limited size and to the relations between these tags.
Dealing with structure alone measures the ability to identify information sources in the context of XML data. Note that in many other domains (e.g. biology) data also come as

ordered labelled trees so that investigating classification and clustering methods for such trees has many applications outside the field of XML.

Structure and Content tasks

Structure and content tasks correspond to classification or clustering of whole XML documents, i.e. the XML tree and the textual content associated to the nodes. In this case the input space is much larger than for the structure only task since both the tag space and the textual content have to be considered.

The structure and content tasks are more challenging than structure alone and encompass different generic tasks. For our purpose XML documents gathered from heterogeneous sources may be characterized through two dimensions: their structure and their thematic. The structure may correspond to a specific DTD or schema (e.g.: for the Inex collection this might be the format of a specific journal or the specific DTD used by an editor for a series) and the thematic may be associated to the general scientific area of the document (e.g. vision, virtual reality, etc). Depending on the application, both dimensions may be involved in the characterization of a class or cluster.

Consider these two dimensions, structure (S) and thematic (T). Figure 1 and 2 illustrate two examples where each class (resp. cluster) corresponds to a given colour. Each class may be considered as a mixture of one or more themes and structures. The different sources and themes may also overlap in different proportions leading to classification (clustering) tasks with different complexity.

Figure 1 illustrates a simple case where classes correspond exactly to structures (S_i) or sources. Even in this simple case, where each category is associated to a structure, content too may be useful when the statistics of the different thematic vary among classes.

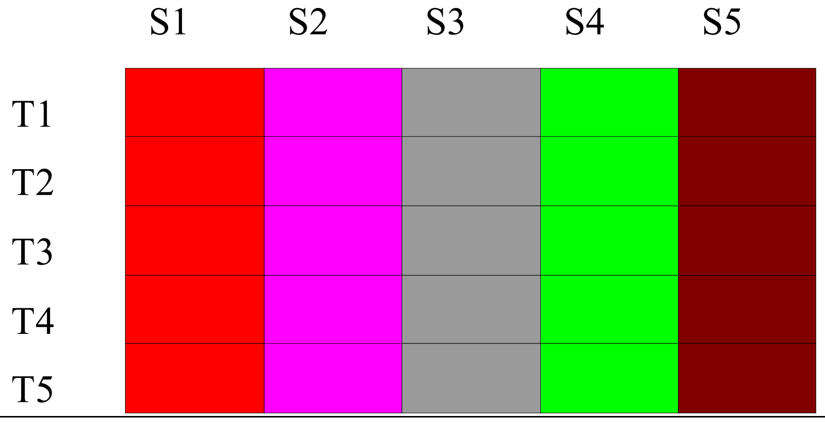


Figure 1 : each source corresponds to a specific structure (5 sources here – corresponding to columns) – the different thematic (rows) will appear in different proportions in each source. Each color corresponds to a class.

Figure 2 illustrates a more complex problem where each class corresponds to a mixture of structural (S) and content (T) information.

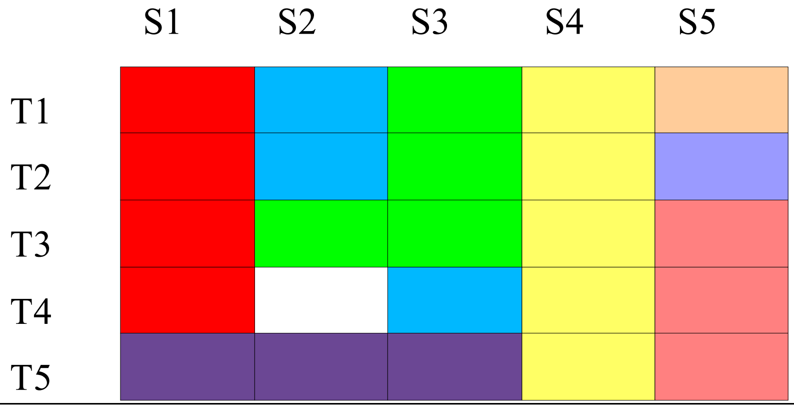


Figure 2: the different structural and thematic classes are visualized horizontally and vertically (there are 5 structures sources and 5 thematic for this example). The different information sources are identified by different colors. 9 classes are thus identified in this figure. Each class corresponds to a mixture of thematic (T) and structure (S) information.

Depending on the practical application in mind, one may distinguish different generic problems like:

- identify common structures across different content types or themes (*structure oriented classification and clustering*) - this is illustrated in figure 1 where each class corresponds to a specific structure and deals with all the collection themes.
- identify different content types across structures (*content oriented classification and clustering*). Classes would then correspond to the horizontal lines in figure 1.
- identify homogeneous classes for both structure and content information (*Content and Structure clustering and classification*) - this corresponds to figure 2. This is the more general task and encompasses many different situations. Classes correspond to colours in figure 2.

Guidelines for the XML Document Mining Tasks

1 Description of the challenge

1.1 Tasks

The challenge will focus on classification and clustering for the two following tasks:

Structure Only: clustering and classification of XML documents using only the structure information present in the documents. The goal of this task is to find the different structural families of documents.
For this task, we consider that each XML document has a single label corresponding to the structural source the document comes from. We aim at finding this structure label for each document of the corpus. Note that this task corresponds to clustering and classification of ordered labelled trees.

Structure and content: clustering and classification of XML documents using structure and content information. Both structure and content may be useful for characterizing the different classes in these tasks. This generic task is representative of many different practical situations where structure and content depend one on the other. For example, documents may share the same type of structure but content (word) distribution is different for the different structural elements across classes. Documents may also come from different sources, all the sources addressing the whole set of thematic but with different thematic distributions across sources.

For simplification we will suppose for both tasks that each document may have only one label (we do not consider the multilabeled case here).

1.2 Corpora

We have developed different corpora for the challenge. For all of them, we have been using or gathering first an existing corpus of XML documents – we make use of three such "original" corpus in the challenge (MovieDB, INEX, WIPO). We have then defined by hand structural DTD transformations and applied them on these original collections thus creating document sets with new characteristics. These transformations were defined so as to highlight and test some specific capacity of ML algorithms and to create a whole set of classification or clustering tasks of different complexities.
The different corpora prepared for the challenge are described below. For each task a training and a test set will be provided. Participant may participate to an arbitrary number of tasks.
Two runs may be submitted for each task.

1.2.1 MovieDB

The **MovieDB** corpus is a corpus of XML documents describing movies. It was built using the IMDB database[1]. It contains 9643 XML documents. Each document is labelled by one thematic category which represents the genre of the movie in the original collection and one

[1] http://www.imdb.com

structure category. There are 11 thematic categories and 11 possible structure categories which correspond to transformations of the original data structure.

MovieDB will be used for both the **structure only** and **structure and content tasks.** For all tasks, there are 11 predefined classes (or clusters)**.** For structure only**,** the first two tasks (denoted **M-DB-S-1-c and M-DB-S-1-cg** in the table below) correspond respectively to classification (-c) and clustering (-cg) of the XML labelled trees of the original MovieDB documents. Other structure only tasks for this corpus (tasks **M-DB-S-2-c and –cg to M-DB-S-4-c and -cg**) do correspond to classification and clustering on a series of transformed MovieDB collections. The transformations have been defined so that each series should be – in principle- more difficult to classify / cluster than the preceding. For s**tructure and content tasks, (M-DB-C&S-1-c or –cg and M-DB-C&S-2-c or –cg) there are** 11 classes which correspond to a specific correlation between structure and content. For the Second data set (-2-c and -2-cg) classes have a higher overlap than for the first one (-1-c and -1-cg) and classification / clustering should be more difficult.
MovieDB has been preprocessed using a Porter stemmer.

1.2.2 INEX

The INEX corpus is the document corpus used for ad-hoc retrieval in INEX since its beginning in 2002. It is composed of 12017 XML documents. Each document comes from an IEEE journal (18 journals in all). The collection has been re-labelled into 6 thematic classes (Computer, Graphics, Hardware, Artificial Intelligence, Internet, Parallel) and 2 structural classes (Transaction journals vs Others). INEX will be used here for the Structure only and the structure and content tasks. For the former (**INEX-S-c and INEX-S-cg**) the goal is to perform classification / clustering into the two structural classes (Transaction journals vs Others). There are two types of structure and content tasks: (**INEX-C&S-1-c and INEX-C&S-1-cg)** correspond respectively to classification and clustering into the 6 hand defined thematic categories. Heterogeneous sources may contribute to the same thematic class, for these two tasks). (**INEX-C&S-2-c and INEX-C&S-2-cg)** do correspond to classification and clustering into the 18 original journal categories for which both structure and content should be used to identify the journal.
INEX has been preprocessed using a Porter stemmer.

1.2.3 WIPO

The WIPO corpus is a corpus of XML documents describing patents. This corpus is content oriented since its structure only brings little information. The corpus is composed of 75250 documents from 114 thematic categories. It will only be used here for the **structure and content** task where the goal will be to find the patent categories (**WIPO- C&S –c and WIPO- C&S –cg tasks).**
WIPO has not been preprocessed.

The tables below summarize the characteristics of the tasks.

1.3 Tasks summary

In the first tables, column -c and –cg hold respectively for classification and clustering. This means for example that M-DB-S-1-c and M-DB-S-1-cg are respectively a classification and a clustering task on the same corpus M-DB-S-1.

We will provide a series of training corpuses with the document labels by August 31 (a preliminary training corpus is made available by the end of July) and test corpuses by October 15. Labels should not be used for clustering. The number of classes is supposed to be known for clustering.

1.3.1 Structure only tasks

Name of the task	**Description of the corpus**	**Number of labels**
M-DB-S-1-c **M-DB-S-1-cg**	Corpus of XML labelled trees from the IMDB database.	11
M-DB-S-2-c **M-DB-S-2-cg**	Same corpus with additional noise on the structure	11
M-DB-S-3-c **M-DB-S-3-cg**	Same corpus with additional noise on the structure	11
M-DB-S-4-c **M-DB-S-4-cg**	Same corpus with additional noise on the structure	11

Tasks **M-DB-S-2-x** to **M-DB-S-4-x** correspond to classification and clustering on transformations of the original labelled trees dataset **M-DB-S**. Transformations 2 to 4 are of increased complexity. Both clustering and classification complexity should increase accordingly.

Name of the task	**Description of the corpus**	**Number of labels**
INEX-S-c **INEX-S-cg**	Labelled trees from the INEX database. The goal is to find if a document comes from a “Transaction on…” series or from an other series	2

1.3.2 Content and Structure tasks

Name of the task	**Description of the corpus**	**Number of labels**
INEX-C&S-1-c **INEX-C&S-1-cg**	Original INEX document database content + structure. 6 classes corresponding to hand defined thematic.	6

Name of the task	**Description of the corpus**	**Number of labels**
WIPO- C&S -c **WIPO- C&S -cg**	This is the WIPO corpus of patents. The goal is to find the patent category.	114

Name of the task	**Description of the corpus**	**Number of labels**
M-DB-C&S-1-c **M-DB-C&S-1-cg**	MovieDB corpus with a hand-defined dependency between content and structure.	11
M-DB-C&S-2-c **M-DB-C&S-2-cg**	Same, but more complex correlations and therefore more complex classification or clustering task.	11

Name of the task	**Description of the corpus**	**Number of labels**
INEX-C&S-2-c	INEX original collection. The task consists	18

INEX-C&S-2-cg	in finding the journal of each document.	

1.4 Evaluation Measures

1.4.1 Clustering: Entropy and Purity

In order to evaluate clustering quality, we will use two measures:

- The entropy between the clusters and the original labels of the documents
- The purity of the clusters

1.4.2 Categorization: Precision and Recall

In order to evaluate classification quality, we will use the classical precision-recall curves and the micro-average and macro-average F1 measures.

1.5 Schedule

The challenge will run in two rounds: the first round corresponds to the present call and will run between July and November 2005 according to the schedule below, with results presented at the INEX workshop at the end of November 2005. A second round will be organized between January and March 2006 with a presentation of the participants' results at a workshop to be organized later on (e.g. the Pascal challenge workshop in April 2006 in Venice, or a joint workshop INEX - Pascal).
For each round, training sets and test sets will be provided. Participants are encouraged to test their methods on the training sets before submitting their results.
For each round, participant may send results for an arbitrary number of tasks.
Two runs may be submitted for each task.

First round schedule

JULY – 30	Training corpora with document labels downloadable from the Web Site. These corpora can be used to make preliminary experiments for clustering and categorization.
AUGUST – 31	Final corpora downloadable on the Web Site (same as above – errors corrected if any).
OCTOBER – 15	Test corpora available.
OCTOBER – 31	Deadline for submitting results.
NOVEMBER – 10	Deadline for submitting articles.
NOVEMBER – 28-30	INEX Workshop in Schloss Dagstuhl (http://www.dagstuhl.de/).

Second round schedule

To come

1.6 How to…

1.6.1 How to get the corpora

The INEX corpus may be used by all participants provided an agreement form is signed. Access to all corpuses will then be conditioned by the signature of this agreement.
The corpora can be downloaded from the challenge web site (http://xmlmining.lip6.fr).
In order to get them, you must:

- Download the agreement form from http://xmlmining.lip6.fr/agreement.pdf and fax us the signed agreement (+33144277000) with the following indications – to: P. Gallinari – subject: xml-mining challenge.
- Send an e-mail with subject "register" at xmlmining@lip6.fr

Once we have received both, you will be sent a login/password to access the collections.

Registered INEX participants are only required to send the e-mail with subject "register" and the name of the INEX registered organization.

1.6.2 Corpora format

Each corpus is included in a zip file containing:

- A set of XML documents. Each file corresponds to one XML document
- A "*relfile*" directory containing one file for each category, each file contains the name of the documents in the corresponding category.

1.6.3 Preprocessing

The corpora (except WIPO) have all been pre-processed. Participants are not supposed to use additional preprocessing in order to facilitate the comparison of the performances of the proposed models.

1.6.4 Additional Information and contact

For any question, please send an email to:
xmlmining@lip6.fr

1.7 Submitting the results

An evaluation server will be made accessible later.

1.8 Organizers

Challenge organizers

Ludovic DENOYER (University Paris 6, Fr)
Patrick GALLINARI (University Paris 6, Fr)
Anne-Marie VERCOUSTRE (Inria Rocquencourt, Fr)

Guillaume WISNIEWSKI (University Paris 6, Fr)

Challenge committee

Pascal side

Samy BENGIO (IDIAP, Martigny, Ch)
Ludovic DENOYER (University Paris 6, Fr, also for INEX)
Patrick GALLINARI (University Paris 6, Fr, also for INEX)
Marko GROBELNIK (Jozef Stefan Institute, Ljubljana, Sl)
Massi PONTIL (University College, London, UK)
Juho ROUSU (Royal Holloway, University of London, UK)

Inex Side

Remi GILLERON (University Lille & INRIA, Fr)
Mounia LALMAS (Queen Mary, London, UK)
Marie-Christine ROUSSET (University Orsay & INRIA, Fr)
Marc TOMMASI (University Lille & INRIA, Fr)
Anastasios TOMBROS (Queen Mary, London, UK)
Anne-Marie VERCOUSTRE (Inria Rocquencourt, Fr)

Relevance Feedback Task

The process of information retrieval is an uncertain one. Searchers may have less than well developed ideas of what they are searching for and the types of information available for retrieval; they may be unable to express a conceptual need for information in terms of a suitable query. Early in the development of IR, researchers recognized that although users often had difficulty expressing their informational needs precisely, they could recognize useful information when they saw it. That is, although searchers may be unable to readily convert informational needs into requests, once the system presents them with an initial set of documents, they can easily differentiate between those documents that do contain useful information and those that do not.

This recognition led to the notion of relevance feedback (RF): users evaluating (marking or selecting) a small set of documents as relevant or irrelevant with respect to an informational need. RF techniques use data from the selected documents (i.e., those returned by the system in response to the user's original query and then evaluated by the user for relevance) to automatically reformulate that query. They modify the initial query and produce a revised query - the feedback query - to be processed by the retrieval system. RF algorithms can be also used for Automatic Query Refinement (AQR) by applying an automatic process that marks the top results returned by the search engine as relevant and the tail results as non relevant for use by subsequent iterations

The aim of this track is to investigate relevance feedback in the context of XML retrieval. In standard full text search engines, RF has been translated into detecting a "bag of words" that are good (or bad) at retrieving relevant information. These terms are then added to (or removed from) the query and weighted according to their power in retrieving relevant information. With XML documents, a more sophisticated approach - one that can exploit the characteristics of XML - is necessary. The approach should ideally consider not only content but also the structural features of XML documents. The query reformulation process must therefore infer which content and structural elements are important for effectively retrieving relevant data.

INEX has two types of tasks, Content Only (CO) and Context and structure (CAS). CO queries are free text queries (like those used in TREC) for which the retrieval system should retrieve relevant XML elements of varying granularity, whereas CAS queries contain explicit structural constraints such as containment conditions. This year is the second year of RF track at INEX. While in the first year the RF track concentrated on CO queries only, this year we will expand the RF track to CAS queries as well.

Please note that participants in the track must register for the INEX initiative. To have access to the test collection and in particular the relevance assessments, participants must perform the relevance assessment task. Participants in the RF track are also required to submit retrieval runs to the ad-hoc task, since the ad-hoc runs will serve as baselines for the RF task.

Description of the task

By 10 August 2005, participants in the Relevance Feedback (RF) track should submit their retrieval runs (search results) as per the Ad Hoc task guidelines. The participant's runs will serve as the baselines upon which RF will be performed. Participants should refer to the Ad Hoc retrieval task guidelines for detailed information on the formatting requirements of search results.

On 21 October 2005, relevance assessment data will be distributed to the participants. RF feedback runs can then be performed, using feedback from the top-ranked 20 elements retrieved in the original CO/CAS runs. To limit the number of RF submissions we chose a subset of some common ad-hoc tracks for participants to test their RF algorithms. Participants may submit up to 3 RF runs for each of their original submitted Ad Hoc runs for the **CO.Thorough, CO+S.Thorough and VVCAS** tracks. Totally there could be at most 9 CO submissions (3 RF * 3 original), 9 CO+S submissions (3 RF * 3 original) and 6 VVCAS submissions (3 RF * 2 Original). Please note that some topics may not be used in the RF track if they are judged inadequate for that purpose (e.g., if they do not retrieve enough relevant elements). There are no restrictions on the number of iterations of relevance feedback for a given query. Participants must submit their RF runs by 09 Nov 2005.

An RF run is built as follows: The assessment of the top 20 elements from the original base run is checked against the relevance assessment data. Those top 20 elements are then frozen with their original rank and the rest of the elements (the unseen elements) are re ranked based on the assessment of the top 20. So an RF submission must have the top 20 elements of the base run as the top 20 elements in the RF run while elements from rank 20 and lower can be re ranked. A participant may apply several RF iterations in each run where in iteration i (i starts from 1) the elements at position (i-1)*20 till i*20-1 (first element is at position 0) are frozen and the elements below them (starting from position i*20) are re ranked. The submission format for the RF runs is similar to the original Ad-hoc runs where we add two new fields to the submission header:

base_run_id – the id of the original ad-hoc run
iterations – number of iterations used for the RF submission

The reported evaluation scores for each RF submission will measure the improvement of the RF run over the original base run.

Participants should be aware that they will have 18 days between the distribution of relevance assessments and submission of RF runs. Evaluation results will be distributed on 15 November 2005.

Schedule

Apr 08:	Deadline for the submission of "Application for Participation".
Apr 09 - Apr 15:	The collection of XML documents will be distributed to all participants on the receipt of their signed data handling agreement.
Apr 18:	Participants will be provided with detailed instructions and

	formatting criteria for candidate topics/queries.
May 06:	Submission deadline for candidate topics.
May 27:	Distribution of final set of topics/queries to participants along with detailed information on the formatting requirements of the search results.
Aug 10:	Submission deadline of search results.
Aug 26:	Distribution of merged results to participants for relevance assessments.
Sep 20:	Submission deadline for relevance assessments.
Oct 21:	Distribution of XML test collection and evaluation scores to participants.
Nov 9:	Submission of the relevance feedback runs
Nov 13:	Submission of papers for the workshop pre-proceedings.
Nov 15:	**Distribution of evaluation scores to participants for the RF track**
Nov 22:	Workshop pre-proceedings and workshop programme online.
Nov 28-30:	Workshop in Schloss Dagstuhl. (http://www.dagstuhl.de/).

Organisers:

Yosi Mass
yosimass@il.ibm.com

Carolyn Crouch
ccrouch@d.umn.edu

NLPX Task:Natural Language Processing for XML Information Retrieval

XML is rapidly becoming an accepted standard for storage, communication, and exchange of information. Most information in typical XML documents is expressed in natural language texts. Yet, most software tools built for XML document retrieval tasks do not use natural language interfaces. Natural Language Processing (NLP) techniques are yet to be applied to Information Retrieval tasks over XML collections.

The field of NLP has been extensively studied in the context of Information Retrieval over Text collections. NLP has the potential to offer better tools for easing the syntax overload of interfaces to XML collections such as the proposed Xpath and XQuery languages. NLP interfaces can exploit the syntax of XML, and pragmatics of natural language, better than current Text Retrieval interfaces that are based primarily on language semantics, keyword/phrase matching and conventional database indexing methods. XML, through its rich self documenting contextual information, is an excellent domain to develop practical and more powerful applications of NLP.

The ultimate goal is to design and build software that will analyse, understand, and generate results in response to queries that humans express naturally. The primary objective of retrieval would be to interpret both structural and content constraints of an information need expressed in a natural language query (as opposed to the rigid syntax of XPath). The IR system would not only select and rank suitable documents, but select the more suitable XML elements within documents that best satisfy the information need (both accurately and concisely).

The purpose of the NLP track at INEX 2005 is to bring together researchers and developers who are interested in exchanging new ideas and presenting results in the field of XML Information Retrieval with emphasis on related NLP tools. The track is intended to act as a forum for promoting interaction among researchers in the field of Natural Language Processing and XML Information Retrieval.

Tasks:

There are two distinct tasks in the NLP track in 2005 - NLQ2NEXI and NLP.

NLQ2NEXI - a simplified task that does not require participants to index the collection or to implement a search engine. Instead, NLQ2NEXI requires the translation of a natural language query, provided in the element of a topic, into a formal INEX <title> element (NEXI is a derivative of XPath with a simplified syntax and having an IR-flavoured interpretation.) The submissions of all participants will be evaluated by a running the titles on search engine/s that can operate on NEXI expressions. The objective is to compare the results obtained with natural language queries (translated into NEXI) with the results that are obtained by the same search engine/s when using the original NEXI expressions. This task is designed to allow new participants with NLP expertise to join the INEX workshop without the need to develop a search engine.

NLQ - this task has no restrictions on the use of any NLP technique to interpret the queries as they appear in the <description> element of a topic. Here participants are required to submit retrieval runs, but enjoy the freedom to implement any NLP techniques in their search engine. The objective is not only to compare between different NLP based systems, but to also compare the results obtained with natural language queries with the results obtained with NEXI queries by any other system in the Ad-hoc track. We wish to test whether natural language queries are effective alternatives to formal queries and to quantify the trade off in performance.

Organisers:

Shlomo Geva
s.geva@qut.edu.au

Tony Sahama
t.sahama@qut.edu.au