

INEX 2004 Workshop Pre-Proceedings

**Norbert Fuhr
Mounia Lalmas
Saadia Malik
Zoltán Szlávik**

December 6-8, 2004
Schloss Dagstuhl
International Conference and Research
Center for Computer Science

<http://inex.is.informatik.uni-duisburg.de:2004/>



<http://www.delos.info/>

Contents

Organisers	6
Preface	7
Acknowledgement	8
Schloss Dagstuhl	9
Methodology	10
NEXI, Now and Next <i>Andrew Trotman, Börkur Sigurbjörnsson</i>	10
If INEX is the Answer, what is the Question? <i>Richard A. O’Keefe</i>	16
Building and Experimenting with a Heterogeneous Collection <i>Zoltán Szlávik, Thomas Rölleke</i>	19
The Interactive Track at INEX 2004 <i>Anastasios Tombros, Birger Larsen, Saadia Malik</i>	24
Reliability Tests for the XCG and inex-2002 Metrics <i>Gabriella Kazai, Mounia Lalmas, Arjen de Vries</i>	33
Ad hoc retrieval	41
MultiText Experiments for INEX 2004 <i>Charles L. A. Clarke, Philip L. Tilker</i>	41
Logic-Based XML Information Retrieval for Determining the Best Element to Retrieve <i>Maryam Karimzadegan, Jafar Habibi, Farhad Oroumchian</i>	43
Analyzing the Properties of XML Fragments decomposed from the INEX Document Collection <i>Kenji Hatano, Hiroko Kinutani, Toshiyuki Amagasa, Yasuhiro Mori, Masatoshi Yoshikawa, Shunsuke Uemura</i>	50

An algebra for Structured Queries in Bayesian Networks <i>Jean-Noël Vittaut, Benjamin Piwowarski, Patrick Gallinari</i>	58
IR of XML documents A collective Ranking Strategy <i>Maha Salem, Alan Woodley, Shlomo Geva</i>	65
TRIX 2004 struggling with the overlap <i>Jaana Kekäläinen, Marko Junkkari, Paavo Arvola, Timo Aalto</i>	72
Merging XML Indices <i>Giambattista Amati, Claudio Carpineto, Giovanni Romano</i>	77
The Utrecht Blend: Basic Ingredients for an XML Retrieval System <i>Roelof van Zwol, Frans Wiering, Virginia Dignum</i>	82
Hybrid XML Retrieval Revisited <i>Jovan Pehcevski, James A. Thom, Anne-Marie Vercoustre</i>	90
A Voting Method for XML retrieval <i>Gilles Hubert</i>	98
The University of Amsterdam at INEX 2004 <i>Börkur Sigurbjörnsson, Jaap Kamps, Maarten de Rijke</i>	104
GPX - Gardens Point XML Information Retrieval at INEX 2004 <i>Shlomo Geva</i>	110
Hierarchical Language Models for XML Component Retrieval <i>Paul Ogilvie, Jamie Callan</i>	118
Ranked Retrieval of Structured Documents with the STerm Vector Space Model <i>Felix Weigel, Klaus U. Schulz, Holger Meuss</i>	126
Component ranking and Automatic Query Refinement for XML Retrieval <i>Yosi Mass, Matan Mandelbrod</i>	134
Ad hoc and Relevance Feedback Track	141

TIJAH at INEX 2004 Modeling Phrases and Relevance Feedback	
<i>Vojkan Mihajlović, Georgina Ramírez, Arjen P. de Vries, Djoerd Hiemstra, Henk Ernst Blok</i>	141
Flexible XML Retrieval Based on the Extended Vector Model	
<i>Carolyn J. Crouch, Aniruddha Mahajan, Archana Bellamkonda</i>	149
Relevance Feedback Track	154
Relevance Feedback for XML Retrieval	
<i>Yosi Mass, Matan Mandelbrod</i>	154
Ad hoc and Heterogeneous Track	158
A Universal Model for XML Information Retrieval	
<i>Maria Izabel M. Azevedo, Lucas Pantuza Amorim, Nivio Ziviani</i>	158
Cheshire II at INEX 04: Fusion and Feedback for the Adhoc and Heterogeneous Tracks	
<i>Ray R. Larson</i>	163
Using a relevance propagation method for Adhoc and Heterogeneous tracks in INEX 2004	
<i>Karen Sauvagnat, Mohand Boughanem</i>	171
Heterogeneous Track	177
A Test Platform for the INEX Heterogeneous Track	
<i>Serge Abiteboul, Ioana Manolescu, Benjamin Nguyen, Nicoleta Preda</i>	177
EXTIRP 2004: Towards heterogeneity	
<i>Miro Lehtonen</i>	183
Natural Language Processing Track	188
NLPX at INEX 2004	
<i>Alan Woodley, Shlomo Geva</i>	188

Analysing Natural Language Queries at INEX 2004 <i>Xavier Tannier, Jean-Jacques Girardot, Mihaela Mathieu</i>	196
Interactive Track	204
Kyungpook National University at INEX 2004: Interactive Track <i>Heesop Kim, Heejung Son</i>	204
APPENDIX	211
Workshop schedule	211
Ad hoc track	212
INEX04 Guidelines for Topic Development <i>Börkur Sigurbjörnsson, Birger Larsen, Mounia Lalmas, Saadia Malik</i>	212
Narrowed Extended XPath I (NEXI) <i>Andrew Trotman, Börkur Sigurbjörnsson</i>	219
INEX 2004 Retrieval Task and Result Submission Specification <i>Mounia Lalmas, Saadia Malik</i>	237
INEX 2004 Relevance Assessment Guide <i>Gabriella Kazai, Mounia Lalmas, Benjamin Piwowarski</i>	241
Evaluation Metrics 2004 <i>Arjen P. de Vries, Gabriella Kazai, Mounia Lalmas</i>	249
Heterogeneous track	251
Guidelines for Topic Development in Heterogeneous Collections <i>Virginia Dignum, Roelof van Zwol</i>	251
Track Result Submission Specification <i>Zoltán Szlávik, Thomas Rölleke</i>	257
Interactive track	260

Track Guidelines	
<i>Tassos Tombros, Birger Larsen, Saadia Malik</i>	260
HyREX for INEX iTrack	
<i>Saadia Malik, Tassos Tombros, Birger Larsen</i>	264
Instructions for Searchers	270

Organisers

Project Leaders:

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

Contact person:

Saadia Malik, University of Duisburg-Essen

Topic format specification:

Börkur Sigurbjörnsson, University of Amsterdam
Andrew Trotman, University of Otago

Online relevance assessment tool:

Benjamin Piwowarski, University of Chile

Metrics:

Gabriella Kazai, Queen Mary University of London
Arjen P. de Vries, Centre for Mathematics and Computer Science

Interactive Track:

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

Relevance feedback track:

Carolyn Crouch, University of Minnesota-Duluth
Mounia Lalmas, Queen Mary, University of London

Heterogeneous Collection Track:

Thomas Rölleke, Queen Mary University of London
Zoltán Szilávik, Queen Mary University of London

Natural Language Processing:

Shlomo Geva, Queensland University of Technology
Tony Sahama, Queensland University of Technology

Preface

The ultimate goal of many information access systems (e.g. digital libraries, web, intranet) is to provide the right content to their end-users. This content is increasingly a mixture of text, multimedia, metadata, and is formatted according to the adopted W3C standard for information repositories, the so-called eXtensible Markup Language (XML). Whereas 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.

The aim of the INEX campaign (Initiative for the Evaluation of XML Retrieval) is to provide the infrastructure and a framework to investigate the performance of information retrieval systems that aim at providing effective access to XML content. More precisely, the aim of the INEX initiative is to provide means, in the form of a large XML test collection and appropriate scoring methods, for the evaluation of content-oriented XML retrieval systems.

The aim of the INEX 2004 workshop is to bring together researchers in the field of XML retrieval who participated in the INEX 2004 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 evaluation of XML retrieval systems.

INEX 2004 was composed of five tracks:

- *Ad hoc retrieval track*, which can be regarded as a simulation of how a digital library might be used, where a static set of XML documents and their components is searched using a new set of queries (topics) containing both content and structural conditions.
- *Interactive track*, which aims to investigate the behaviour of users when interacting with components of XML documents.
- *Heterogeneous collection track*, where retrieval is on a collection comprising various XML sub-collections from different digital libraries, as well as material from other computer science-related resources.
- *Relevance feedback track* dealing with relevance feedback methods for XML.
- *Natural language track* where natural language formulations of structural conditions of queries have to be answered.

The workshop is organised into presentation and workshop sessions. During the presentation sessions participants will have the opportunity to present their approaches to XML indexing and retrieval taken within INEX 2004, in any of the above tracks. Papers related to evaluation methodology in any of or across the above five tracks will also be presented.

The workshop sessions will serve as discussion forums to review issues related to the creation of the INEX topics, the definition of relevance, the use of the on-line assessment system, the development of evaluation metrics, and the various tracks.

Acknowledgement

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. Special thanks go to Shlomo Geva for setting up the WIKI server and Gabriella Kazai for helping with the various documentations.

We gratefully acknowledge the involvement of Börkur Sigurbjörnsson and Andrew Trotman (Topic Format Specification), Benjamin Piwowarski (Online Assessment tool), and Gabriella Kazai and Arjen de Vries (Metrics).

The organizers of the various tracks have done a great job and their work is greatly appreciated: Anastasios Tombros, Birger Larsen, Thomas Rölleke, Carolyn Crouch, Shlomo Geva and Tony Sahama.

Finally, we would like to thank the participating organisations and people for their participation to INEX 2004 as well as their contributions to evaluation methodologies for XML retrieval.

We hope you have enjoyed the INEX 2004 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
Zoltán Szilávik, Queen Mary University of London

December 2004

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/>

NEXI, Now and Next

Andrew Trotman
Department of Computer Science
University of Otago
Dunedin, New Zealand
andrew@cs.otago.ac.nz

Börkur Sigurbjörnsson
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands
borkur@science.uva.nl

ABSTRACT

NEXI was introduced in INEX 2004 as a query language for specifying structured and unstructured queries on XML documents. A language expressive enough for INEX yet simple enough for users to get right. These goals have been achieved. In particular, the error rate in CAS queries has dropped from 63% in 2003 to 12% in 2004. This drop is shown to be a consequence of not only the language, but the tools introduced with it: the source code for a parser was downloaded by 13 IP addresses, while a web implementation was accessed 635 times from 71 addresses.

Although NEXI is suitable for the *ad hoc* track, it is not sufficiently expressive enough for the heterogeneous track, or for question answering. The syntax necessary to extend to these purposes is proposed. This includes weighted terms and weighted paths. The new syntax is strictly an extension so does not invalidate any existing queries.

1. INTRODUCTION

Each of the first three INEX [4] workshops used a different query language. At the first workshop queries were specified in XML [6], at the second in XPath [7], and at the third in NEXI [14]. This succession of languages occurred because, as a consequence of each workshop, new and different query types, and how to specify them, have become clear.

The first INEX workshop was modeled on TREC, and consequently a TREC-like topic format was chosen. Topics were broken into four parts, title, description, narrative and keywords. Of these, the title contained the IR query, and is consequently of focus. For Content Only (CO) queries, the title was a two or three word description of the topic. For Content And Structure (CAS) queries, the title was further marked up in XML. The optional `<te>` tag was used to specify target elements for the search, while `<cw>` was used to identify content words that were optionally associated with a container element, `<ce>`.

```
<Title>  
  <te>tig</te>  
  <cw>QBIC</cw><ce>bibl</ce>  
  <cw>image retrieval</cw>  
</Title>
```

Figure 1: An INEX 2002 query fragment (INEX topic 05).

An example query, the title element from INEX topic 05 is given in Figure 1. In this example, the user is searching for documents that contain the phrase “image retrieval”, contain the word QBIC in a `<bibl>` element, and asking for `<tig>` elements to be retrieved.

It was quickly established that this query language was insufficient for the need [11].

First, the XML format allowed the user to specify queries that were simple mechanical processes. In the above example, once relevant documents have been identified, the process of extracting the `<tig>` (or title group) is mechanical. There is one, and only one, `<tig>` element in each document. Identifying and extracting it can be done with a simple text search.

Second, the language was not expressive enough. The target element was specified irrespective of the context of the query. It was not possible to specify a query of the nature “find sections about sunny New Zealand”; the nearest such query was “find sections from documents about sunny New Zealand” – two quite different queries.

For the second workshop XPath [1] was adopted in the hope it would alleviate these problems, and it did. With the addition of a function for ranked information retrieval (*about*), and the elimination of non-IR functions (e.g. *contains*) XPath proved sufficiently expressive.

XPath introduced new problems! O’Keefe and Trotman [10] provide an analysis of the failure of XPath as a query language for INEX. Perhaps the most damning evidence is the error rate in the official topics. Of the 30 CAS topics, 19 contained errors; that is a 63% error rate in queries written by IR experts.

Subsequently, the INEX 2003 Queries Working Group identified the requirements for a query language suitable for INEX [13]. In brief, it had to look like XPath, be easier to use, and oriented to IR.

Considerable effort was spent defining the query language NEXI [14], used at the 2004 workshop. Designed with the sole purpose of satisfying the requirements of INEX (and the Queries Working Group), this language is a simplified XPath containing only the descendant axis; while at the same time an extended XPath containing the *about* function. NEXI is in use at the current (2004) workshop.

The use of NEXI within and without INEX is examined. From this, the conclusion is drawn that it has successfully proven to be a suitable language for XML retrieval. Future requirements are examined, and extensions are proposed. Adoption of these extensions is recommended.

2. CURRENT STATE OF PLAY

The *ad hoc* track at INEX consists of two tasks, the Content Only (CO) and Content and Structure (CAS) tasks.

In the CO task, it is the task of the search engine to identify relevant document elements that satisfy a user query. By

definition, the query does not specify where to look, or what elements to retrieve. A CO query is a sequence of terms, and example of which is INEX Topic 37: “temporal database queries and query processing”. For this query, the search engine is expected to identify and return a relevance ranked list of document elements about temporal database queries and query processing.

There are two variants of the CAS task, the Strict CAS (SCAS)¹ and the Vague CAS (VCAS). The queries for both are the same; it is only the interpretation that differs – the reader is referred to Fuhr, Malik, and Lalmas [3] for details. In a CAS query, structural elements are included in the query. If a user wishes to find document abstracts that discuss INEX, it is necessary to specify <abstract> as the target element. If a user is searching for smith, but knows they want Dr. Smith and not an ironmonger, they may specify that Smith is an author.

The Queries Working Group at INEX 2003 [13] identified the requirements of a query language necessary to satisfy CAS queries within the context of INEX. In brief, that language must:

- Be a subset of XPath, so as to be familiar to the XML community. Tag instancing was removed, axes were limited to only the descendant axis, filters remained but the not-equals operator was not permitted with string types.
- Support multiple data types. String and numeric types were specified. XPath filters remained, but a restricted set of operators was included.
- Be vaguely interpretable. It must be an IR language. To this end, the AND operator and OR operator were specified as ANDish and ORish.
- Specify one and only one target element (shown below to have been violated).

Additionally, this language allowed the specification of CO queries. It was also specified as extensible.

Trotman and Sigurbjörnsson [14] proposed NEXI, an IR query language for XML that satisfied the requirements of the Working Group and was subsequently adopted for the 2004 INEX. They also provided the source code to a parser, and for INEX 2004 an on-line parser.

2.1 Query Errors at INEX 2004

Examining the first release of the topics for 2004 (version 2004-01), 4 of the 34 CAS queries contain errors (12%). In the CO queries 6 of 39 contain errors (15%). The error rate in CAS is now lower than that in CO.

2.1.1 Examining CAS errors:

Topics 137 and 158 were missing a close bracket at the end of the query. There are corrected by appending ‘}’.

Topic 138 contained the incorrect expression “about(./sec,thread implementation)” which is incorrect in the first comma. This is corrected by removing the erroneous comma.

Topic 161 contained the incorrect expression “about(./atl, database access methods)” which is incorrect in so far as it uses the child axis. This is corrected by replacing “/” with “//”.

¹ At INEX 2004, SCAS was deprecated

2.1.2 Examining CO errors:

Topics 176, 177, and 196 contained illegal punctuation. This is corrected by removing the punctuation.

Topic 190 contained the quoted expression ““e-commerce”” which, as the hyphen makes e-commerce a single word, is a single word phrase. Phrases consist of strictly more than one word so this is erroneous. This is corrected by removing the quotes.

Topics 178 and 179 contain phrases delimited with question mark characters “?”. This is corrected by replacing those characters with quotes.

2.2 Online Parser

In 2004 an online query syntax checker was introduced. Use was logged, with accesses from the University of Otago stripped (to avoid skewing by the developers). Logs were analyzed for the period April 12th through to October 26th, between the date when the parser went online, and when analysis began. Table 1 shows the number of times the parser was accessed each month.

There was a total of 635 requests on 37 distinct dates from 71 internet addresses. Most of the requests occurred during April and May. The topic submission date was May 7th. In Figure 1, the cumulative number of requests on each day of activity is shown. There is a clear burst of activity around the submission date, and finishing on 11th May. Activity immediately after submission date may be caused by late submissions.

Table 1: Parse requests to the online NEXI parser

Month	Requests
April	167
May	447
June	4
July	3
August	5
September	9

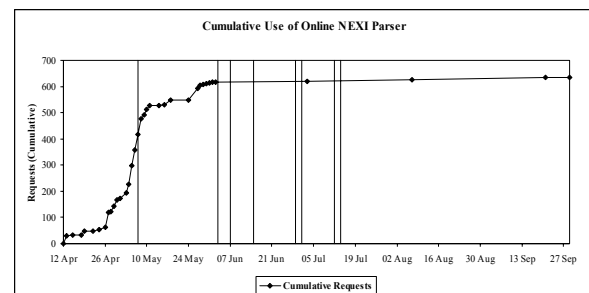


Figure 1: Cumulative use of the online NEXI parser shows considerable use between April 27th and May 11th. The topic submission date was May 7th. Vertical lines are shown for the topic submission date, and each revision date.

After the submission date, but before the first release of the topic set, there was a clear burst of activity (18th through 28th May), this is likely to be the period in which topics were corrected. There was very little activity during the period in which the topic set was under revision, with only 3 requests between the first release (version 2004-001) and the final release (version 2004-07).

It is hard to account for activity in August and September. The requests were valid and the authors are using the parser for the purpose in which it was designed (users are not hacking the parser).

The parser was in the New Zealand time zone, whereas a time-zone for the due and release dates was not given. Requests from the University of Otago were removed from the logs before analysis.

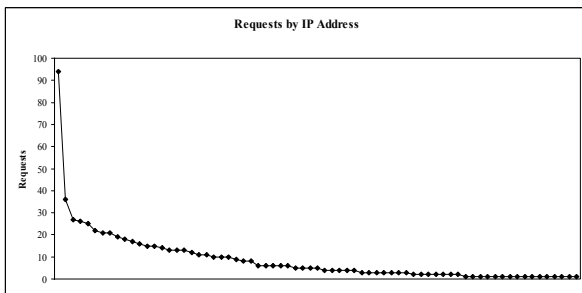


Figure 2: Number of requests from each IP address in decreasing order.

Figure 2 shows the number of requests for each accessing IP address. The number of requests ranged from 94 to 1. The 94 accesses appears to be an outlying point; with the next highest accesses being 36 and then 27 requests. The mean number of requests per address was 8.9, the median being 5.

No effort has been spent trying to resolve IP addresses to institutions; doing so is likely to decrease the number of addresses and increase the mean and median.

2.3 Was NEXI Successful?

The initial error rate in queries has dropped from 63% in 2003 to 12% in 2004. The error rate for CAS topics is now about the same as that in CO topics. The number of topic revisions has halved. From this it would be reasonable to conclude changes made between 2003 and 2004 had a marked effect on syntactic correctness of queries. Those changes were not, however, limited to query language changes.

First, the queries submitted to INEX were checked for syntax errors as part of the selection process. This bias, although present, is not a major contributing factor. Of the originally submitted 84 CAS queries, 18 (21%) contained errors, whereas of the 107 CO topics, 19 (18%) contained errors. These two error rates are about equal. The error rate in the original submissions in 2003 is not known.

Second, having written XPath parsers for 2003, the participants themselves should have been familiar with the language, and therefore more able to write syntactically correct queries than before.

Third, web access to an online parser was made available during the topic development period. This has, no doubt, had an effect on the correctness of the submitted queries.

Fourth, the source for a command line version of the parser was attached to the language specification; and downloadable from the web site. It was downloaded by 13 IP addresses; discussion with some INEX participants suggests it was also used.

The decrease of errors in CAS topics is considered a sign of NEXI success; however, there are still areas that need addressing. During 2003, the topics underwent 12 revisions over a period of 38 days. In 2004, it took only 7 revisions, but 41 days. One can but hope that in future years topics are submitted correctly and on time.

3. THE FUTURE

NEXI was, by design, the simplest query language that could possibly work. The subset of XPath was chosen in order to ensure nothing unnecessary was included. To this end, NEXI has proven a success for *ad hoc* searching, but only for *ad hoc* searching – it has proven unsuitable for other types of search. This shortfall is now addressed with additions for question answering, heterogeneous searching, and a new wildcard.

3.1 Wildcards

The NEXI path wildcard operator, *, is defined as meaning “first or subsequent descendant” [14]. A new “here or below” wildcard, +, is introduced, but it is of limited use.

As `//article//+` means “article or below”, `//+` must mean “nothing or below”. This nothingness is meaningless, as there must be at least one element present. Specifying the existence of one or more elements is done with `//*`. Use of `//+` is therefore prohibited.

Use of two or more adjacent `//+` operators is meaningless; `//article//+` and `//article//+//+` are semantically equivalent. The two forms `//article//+//bm` and `//article//bm` are also equivalent. Use of the + inside a path is meaningless as it simply specifies there might be a node, which is implicit in the descendant operator.

There exists only one place this new operator can be used; the end of a path specification. The form `//*//+` is redundant, and equivalent to `//*`, further restricting the use of +.

The new addition to the path syntax is:

```
zero_any_node: NODE_QUALIFIER '+'
```

which requires the following changes:

```
path: any_node
    | node_sequence
    | node_sequence any_node
    | node_sequence attribute_node
    | node_sequence any_node attribute_node
    | node_sequence zero_any_node

node_sequence: node
    | node_sequence node
    | node_sequence any_node node
```

```
| node_sequence any_node any_node
```

```
node: named_node | tag_list_node
```

It is unfortunate that the late addition of the + wildcard operator results in * meaning one or more and + meaning zero or more because these two operators have each other's definition in regular expressions.

Strict interpretation: “//A/+” means at or below the “//A” element.

Loose interpretation: “As paths are only hints, feel free to ignore this”

3.2 Multiple Target Elements

The tag list syntax, “//(A|B)” means “either the A or the B element”. As this syntax is not forbidden as the target element, it might be exploited by a topic author to identify multiple target elements. This use, although valid, is discouraged.

3.3 NEXI for Question Answering

There is currently no question answering track at INEX, however the authors anticipate there being so. Ogilvie [9] has already discussed the inadequacies of NEXI to fulfill this role. We concede, it was not designed for this purpose and does not fulfill the role. Ogilvie does, however, propose syntax for the purpose.

In place of an *about* function, Ogilvie suggests a *weight* function; which he gives by example:

```
//sentence[./event//VBD[weight(0.4 kill 0.3
assassinate 0.2 murder 0.1 shoot)] AND
./patient//person[weight(0.4 'Abraham
Lincoln' 0.4 'President Lincoln' 0.1 'honest
Abe' 0.1 Lincoln)]]/agent//person
```

weight differs from *about* in three ways. First, phrases are specified using single quotes in place of double quotes. Second, the path occurs outside the clause rather than inside it. Third, weights for each term are given. Altering the *weight* to resemble *about* results in:

Example:

```
//sentence[weight(./event//VBD, 0.4 kill 0.3
assassinate 0.2 murder 0.1 shoot) AND
weight(./patient//person, 0.4 "Abraham
Lincoln" 0.4 "President Lincoln" 0.1 "honest
Abe" 0.1 Lincoln)]]/agent//person
```

the formal syntax of which is:

```
decimal: NUMBER | NUMBER '.' NUMBER

WEIGHT: "weight"

weighted_co: decimal term
| weighted_co decimal term

weight_clause: WEIGHT '(' relative_path ','
```

```
weighted_co ')'
```

additionally, the definition of filter is altered to:

```
filter: about_clause
| weight_clause
| arithmetic_clause
```

Strict interpretation: “In the example, only a //sentence//agent//person element is correct, that said, it will most likely tell me who killed honest Abe”.

Loose interpretation: “What I want is most likely a //sentence//agent//person element that will tell me who assassinated honest Abe. I know several ways of saying assassinate, and honest Abe, here are some and how likely I think you are to see them – but I might be wrong about this”.

3.3.1 QA Paths

Ogilvie notes that path semantics may require relaxation for Question Answering. The paths may, instead, refer to a structural annotation of the document content. In no way should NEXI be interpreted as prohibiting any such interpretation of paths – this is the loose interpretation embraced.

3.4 NEXI for Heterogeneous Searching

The heterogeneous track chose a subset of topics from the *ad hoc* track, and added to them some special purpose topics. Of the chosen topics, 161 and 196 contained errors (discussed above). In version 2 of the heterogeneous topics there are 4 added topics, one of which contains spurious punctuation (topic 4). Topics should be checked for syntax errors before inclusion in any topic list.

The heterogeneous track has four types of queries, Content Only (CO), Basic CAS (BCAS), Complex CAS (CCAS) and Extended CCAS (ECCAS).

This year CO topics from the *ad hoc* track were used for the heterogeneous track. As the IEEE collection is part of the heterogeneous collection, this decision avoids any additional relevance assessing on that collection. Consequently, all CO topics in the heterogeneous track are already in NEXI.

Basic CAS topics contain one structural constraint and one textual constraint. They can all be specified in the form

```
//constraint[about(., content)]
```

where constraint and content are single terms. This is a subset of NEXI which was, consequently, chosen for specifying BCAS topics.

Complex CAS topics are the heterogeneous equivalent of *ad hoc* CAS topics. They are in the form //A[B] or //A[B]//C[D]. CCAS topics are specified in NEXI.

Extended Complex Content and Structure (ECCAS) topics allow the query author to specify a belief in the correctness of a structural constraint. The example given in the track guidelines [2] is:

```
//author(0.8)[about(title(0.5), 'Information Retrieval')],
```

in which the user has an 80% certainty the answer is an author element, thinks the article will be about information retrieval, but has only a 50% certain that this will be discussed in the title. There were no ECCAS topics submitted and NEXI did not support syntax for them.

ECCAS topics are expected in future years. To this end, syntax supporting user certainty in tag specification is needed. Extending NEXI would require only small changes from the syntax proposed in the heterogeneous track guidelines.

First, in NEXI phrases are specified using double quotes, phrases in ECCAS should be specified in the same way. Second, paths in a NEXI *about* function are relative to the context path (the path being filtered) but in the example given in the heterogeneous track guidelines [2], the path is an absolute path. The change to absolute paths prevents the specification of queries that can be resolved through a mechanical process, however it also restricts the expressiveness of the query – these kinds of queries can't be written. This tradeoff is considered acceptable.

The syntax requires only small changes:

```
weight: '(' decimal ')'
```

```
tag: XMLTAG | XMLTAG weight
```

Strict interpretation “//A(0.5)” is a 0.5 certainty in the correctness of “//A” for the purpose in which it is being used. “//A(0.5)//B(0.3)” is a 0.3 certainty of “//A//B” for its purpose and a 0.5 certainty in “//A” for its purpose. In the expression “//(A(0.2) | B(0.5))”, the certainty of being “//A” is given along with the certainty of “//B”. The certainty values are only hits, and are open to interpretation.

Loose interpretation “I’m not sure where to look, these places might be good”

3.5 Uncertain NEXI

The heterogeneous additions combined with the question answering additions provide the syntax necessary for certainty of path and certainty of search term combinations. A query of this nature can be considered super-loose or utterly uncertain; the user is uncertain of everything (a THISish search?).

Example:

```
//bb(0.3)[weight(., 0.2 "Information Retrieval")]
```

Strict interpretation: There is no strict interpretation.

Loose interpretation: “The answer is probably a <bb> element, and it probably says something about Information Retrieval, but I’m not certain about this”

3.6 Relevance Feedback NEXI

In relevance feedback it is not uncommon to add additional search terms or to weight search terms. The natural analogue for structured searching is adding paths and weighting paths. Syntax for both weighting terms and paths is suggested above. Here the applicability to relevance feedback is identified.

4. OTHER NEXI RELATED WORK

Kamps *et al.* [5] suggest adding the ancestor axis to NEXI. They call this superset Positive Temporal XPath. Although this syntax is not more expressive (all queries specifiable in Positive Temporal XPath can be expressed in NEXI), they suggest specifying a path from child to parent is more natural to some users than *vice versa*. They conjecture that paths specified using both ancestor and descendant may be more succinct than using just one or the other.

It is unfortunate that some users prefer parent to child, while others prefer child to parent; using one or the other is simpler than using either or both. In an effort to remain simple, the introduction of an ancestor axis to NEXI is left as future work.

Mihajlović *et al.* [8] choose to store the INEX collection in a relational database. Between the relational database and NEXI they introduce an algebra. With this approach it is possible to change (and experiment with) the underlying relational structure independent of the algebraic optimization of query expressions. It also allows the introduction and optimization of XML IR operators such as *about*. They choose the range approach for searching structured documents and consequently their introduced algebra is an algebra of regions. Piwowarski and Gallinari [12] prefer a probabilistic implementation and introduce a probabilistic algebra for a subset of XPath which is a superset of NEXI.

5. CONCLUSIONS

NEXI has proven to be successful for INEX. This success is due to a combination of the simple XPath like syntax, the online parser, and the command-line parser. The online parser was used a total of 635 times from 71 IP addresses, the command line parser was downloaded from 13 IP addresses. As a consequence of this use the error rate in CAS queries dropped from 63% in 2003 to 12% in 2004.

Although NEXI has proven suitable for *ad hoc* retrieval, it has also proven inadequate for question answering and heterogeneous searching. New syntax is added for these purposes. In essence, this new syntax adds weighted paths and weighted search terms. These extensions might also be used for relevance feedback.

Wildcards in paths are extended to include a zero or more descendants wildcard, +. The new wildcard is meaningless except at the end of a path.

The adoption of the extensions proposed herein will allow tracks in addition to *ad hoc* to use NEXI. This use, and continued use in the *ad hoc* track, is recommended.

6. REFERENCES

- [1] Clark, J., & DeRose, S. (1999). XML path language (XPath) 1.0, W3C recommendation. The World Wide Web Consortium. Available: <http://www.w3.org/TR/xpath> [2004.
- [2] Dignum, V., & Zwol, R. v. (2004). Guidelines for topic development in heterogeneous collections. Available:

<http://inex.is.informatik.uni->

duisburg.de:2004/internal/hettrack/downloads/hettopics.pdf

- [3] Fuhr, N., Malik, S., & Lalmas, M. (2003). Overview of the initiative for the evaluation of XML retrieval (INEX) 2003. In *Proceedings of the INEX 2003 Workshop*, (pp. 1-11).
- [4] Gövert, N., & Kazai, G. (2002). Overview of the initiative for the evaluation of XML retrieval (INEX) 2002. In *Proceedings of the 1st Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, (pp. 1-17).
- [5] Kamps, J., Marx, M., Rijke, M. d., & Sigurbjörnsson, B. (2004). Best-match querying for document-centric XML. In *Proceedings of the 7th International Workshop on the Web and Databases (WebDB 2004)*, (pp. 55-60).
- [6] Kazai, G., Lalmas, M., & Malik, S. (2002). INEX guidelines for topic development. In *Proceedings of the 1st workshop of the initiative for the evaluation of XML retrieval (INEX)*, (pp. 178-181).
- [7] Kazai, G., Lalmas, M., & Malik, S. (2003). INEX '03 guidelines for topic development. In *Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX)*.
- [8] Mihajlovic, V., Hiemstra, D., Blok, H. E., & Apers, P. M. G. (2004). An XML-IR-db-sandwich: Is it better with an algebra in between? In *Proceedings of the SIGIR workshop on Information Retrieval and Databases (WIRD'04)*.
- [9] Ogilvie, P. (2004). Retrieval using structure for question answering. In *Proceedings of the 1st Twente Data Management Workshop - XML Databases and Information Retrieval*, (pp. 15-23).
- [10] O'Keefe, R. A., & Trotman, A. (2003). The simplest query language that could possibly work. In *Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX)*.
- [11] Pehcevski, J., Thom, J., & Vercoustre, A.-M. (2003). XML-search query language: Needs and requirements. In *Proceedings of the AusWeb03: Changing the Way We Work*.
- [12] Piwowarski, B., & Gallinari, P. (2004). An algebra for probabilistic XML retrieval. In *Proceedings of the 1st Twente Data Management Workshop - XML Databases and Information Retrieval*.
- [13] Sigurbjörnsson, B., & Trotman, A. (2003). Queries: INEX 2003 working group report. In *Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX)*.
- [14] Trotman, A., & Sigurbjörnsson, B. (2004). Narrowed Extended XPath I (NEXI). In *Proceedings of the 3rd workshop of the initiative for the evaluation of XML retrieval (INEX)*.

If INEX is the Answer, what is the Question?

Richard A. O’Keefe
Department of Computer Science
University of Otago
Dunedin, New Zealand
ok@cs.otago.ac.nz

ABSTRACT

The INEX query languages allow the extraction of fragments from selected documents. This power is not much used in INEX queries. The paper suggests reasons why, and considers which kind of document collection this feature might be useful for.

1. WHAT IS THE INEX ANSWER?

We can distinguish four kinds of IR-like query for semi-structured data:

CO Content Only—a classical information retrieval query to select a document from a collection of documents based on the occurrence of terms and phrases anywhere within it.

CC Content-in-Context—a combination of contexts (paths) and CO queries to apply in those contexts, used to select documents from a collection of documents. Queries like this have been around almost as there have been SGML collections to search in.

EC Element-in-Context—a CC-like query is used to select elements from documents in a collection, with each element being treated as if it were a document and reported separately. These are NEXI “Basic CAS” queries. You can see CC queries as BCAS queries that just happen to select `article` elements, but the distinction between CC and EC is useful.

2S Two-Stage—An EC query is used to select elements, and then a further EC query is used to select portions of those elements. This is not used for highlighting within documents; the elements selected in the second stage are reported separately.

The INEX Answer is “EC and 2S queries”.

2. WHAT IS PROBLEMATIC ABOUT THE INEX ANSWER?

It turns out that INEX participants have found it very hard to formulate non-trivial EC and 2S queries, and even harder to evaluate them. The INEX’03 topics included thirty Content and Structure queries:

N	type	tag	gloss
14	CC	<code>article</code>	whole articles
3	EC	<code>sec</code>	sections
1	EC	<code>abs</code>	abstracts
1	EC	<code>p</code>	paragraphs
1	EC	<code>vt</code>	<i>curricula vitae</i>
6	2S	<code>sec</code>	sections
2	2S	<code>abs</code>	abstracts
1	2S	<code>bb</code>	bibliography items
2	2S	*	IR engine’s choice

That is, nearly half of the queries did not exploit the INEX Answer.

One reason for this is simply that there is not a lot of structure that one can usefully exploit in the INEX collection. Basically, there’s front matter, including authors, title, and abstracts, body with a whole bunch of variously tagged sections and subsections, and back matter with bibliography and author biographies.

Things changed in 2004, but not much. There were 35 CAS topics.

N	type	tag	gloss
8	CC	<code>article</code>	whole articles
2	EC	<code>sec</code>	sections
1	EC	<code>abs</code>	abstracts
1	EC	<code>p</code>	paragraphs
1	EC	<code>vt</code>	<i>curricula vitae</i>
1	EC	<code>bib</code>	entire bibliographies
1	EC	<code>(p fgc)</code>	paragraphs or figure captions
8	2S	<code>sec</code>	sections
1	2S	<code>abs</code>	abstracts
1	2S	<code>bb</code>	bibliography items
1	2S	<code>p</code>	paragraphs
1	2S	<code>fig</code>	figures
1	2S	<code>bdy</code>	whole bodies
2	2S	*	IR engine’s choice

A little over three quarters of the INEX’04 CAS queries did

exploit the INEX Answer, but how usefully?

Some of these queries are thought-provoking.

- In query 161, the containing `article` must be about access methods for spatial data and text, while the selected `bb` elements need not be about either. They could be about access methods for time series, for example.
- In query 158, the containing `article` must be about the Turing test, while the selected `bdy` element must be about the “turning” test. Nor is it clear why it’s useful to see an article without its title, authors, or abstract.
- Query 158 also makes one wonder how a query of the form `about(./fm, x)` or `about(./abs, x)` differs from a simple `about(./fm, x)`, since `abs` only occurs inside `fm`.
- Query 127 with its `(p|fgc)` reminds us that while the average `p` in the INEX collection has about 300 characters of text, the average `fgc` has about 150 characters. So perhaps more (all?) queries that accept `p` elements should also accept `fgc` elements.
- Query 136, selecting entire bibliographies on the basis of “text” and “categorisation” appearing somewhere and “Support Vector Machines” and “SVM” appearing somewhere else reminds us that titles are not a reliable guide to relevance. Who would dream from the title alone that *Bananas in Space* was about “functional programming” using the “Bird-Meertens” formalism?
- Query 142, of the form `//abs[about(...)]`, makes one wonder why it is useful to find an interesting abstract if you can’t tell which article it’s an abstract *of*.

Queries must not only be formulated, they must be evaluated. And to evaluate the relevance of an element, you may need a greater or lesser amount of context. As IR researchers well know, words are ambiguous. If you see “Algol is very old”, is that talking about the star or the programming language (and if so, which)? If you see “The tables were too crowded”, is this a complaint about a paper or a dining hall?

This points out a serious methodological problem in the INEX evaluation procedure. Judges rate elements within the scope of complete articles (which they can and do look at), while users would presumably just see the elements. That is, for CO and CC queries, the judge and the user have the same information available to them, while for EC and 2S queries, the judge has far more information at his or her disposal in making relevance judgements than someone just receiving the paragraphs or sections in question would. For abstracts and sections, this may not be too much of a problem, but paragraph, title, and bibliography item it is almost certainly a distortion. Even for sections, I know that I found myself either able to dismiss an entire article quickly (having looked at a portion that was not part of the selected response) or else having to read the entire article with care to decide what the flagged elements actually *meant* before

I could decide how relevant they were. Does it even make sense to talk about a small element *having* any relevance without its context?

3. WHAT MIGHT THE QUESTION BE?

3.1 Strong semantics for markup

Some markup in the INEX collection has strong semantics. An `ead` element should be an e-mail address, nothing else. The `mo`, `day`, and `yr` elements are parts of dates. A `bb` element is always a bibliographic reference. The `abs`, `bb`, and `vt` elements are clearly useful in queries.

Some markup in the INEX collection has presentation semantics. The `it` and `rm` elements select italic and roman faces, but say nothing about why. It is not accidental that none of the queries mention these elements, and it is only regrettable that the evaluation system requires people to judge these elements.

Some markup is structural, without having much semantics. There is nothing to mark the rhetorical structure of a document or the rhetorical force of any element. There is, for example, no distinction between “quoted in support” and “quoted for rebuttal”. Structural elements are surprisingly popular in queries, principally `sec` with some `p`. One feels that this may be an artefact of the INEX setup: people are under pressure to select *something* to show that the INEX Answer is useful, and `sec` is the smallest nearly-self-contained element. It is difficult to imagine any queries where `ss1` or `ss2` would be meaningful choices.

An INEX Question really needs a wider range of elements with strong semantics: `exercise`, `example`, `poetry` (in the INEX DTD, but apparently not used anywhere), `warning`, `listing`, `scene`, `design.pattern`, that kind of thing.

3.2 Low coupling

What really matters is not how big the fragments are but how tightly they are coupled to their context. The Wall Street Journal documents from TREC are smaller than most of the IEEE `sec` elements, but they were written to be free-standing. The `bb` and `vt` elements make good sense as fragments in the existing INEX collection because they depend hardly at all on their context. Abstracts are crafted to be fairly self-contained. In contrast, `p` elements are so tightly linked to their context as to be difficult to judge, even though they are bigger than most `bb` elements. The very smallest body extracts that work are `sec`, and even they depend too much on context for comfort.

We need a collection of documents which have pieces whose relevance can be judged on their own.

3.3 Some coupling

If the fragments we want are not coupled to their containing document at all, why aren’t they stored as free-standing documents in the first place? There has to be enough coupling so that the first EC filter usefully limits the scope of the second EC filter.

3.4 Sizeable fragments

If you find a relevant `sec`, don't you want to know what article it came from in case there's more good stuff there, or to find the author's address to write for more information? One reason you might not want to do this is if the "documents" are too big to examine or too unlikely to contain other relevant material.

3.5 Examples

- From the Otago Daily Times, issues in 2003, find stories about Don Brash.

Newspapers contain many stories with low or no coupling. This is almost a WSJ query. The trick is to find queries with more constraints on the container (issue).

- From the Otago Daily Times, issues since 2000 having editorials about the foreshore or race relations, find stories about Don Brash and the foreshore or race relations.

This is almost the same as the previous query, but basically uses the newspaper editor as a relevance filter. It feels contrived; basically these two examples fail the "some coupling" requirement.

- From movies in the detective story genre set in San Francisco, select scenes where Nicole Kidman speaks.

This satisfies the "sizeable fragment" requirement.

- From CDs that contain Irish music, select planxties.

This satisfies "low coupling", "sizeable fragment", and "some coupling".

- From books about anatomy, select sections about the articulation of the jaw.

This is a real query I had while I was writing the paper. The answers I found satisfied "low coupling" and "sizeable fragment".

- From books about Bioinformatics published after 1994, select portions about Dynamic Time Warps.

Publication date is a property of the books as wholes, not of sections. Dynamic Time Warps have many applications other than Bioinformatics. So this satisfies "some coupling" as well as "sizeable fragments".

- From books by Terry Pratchett, select chapters that mention a "Soul Cake" day.

- From R packages that are about trees, select function descriptions that are about pruning trees.

There are over 1200 pages of function documentation for core R; the contributed packages add about as much more. The function descriptions are similar to UNIX manual pages, only bigger. This satisfies "some coupling" and "sizeable fragments".

Building and Experimenting with a Heterogeneous Collection

Zoltán Szlávik
Queen Mary University of London
London, United Kingdom
zolley@dcs.qmul.ac.uk

Thomas Rölleke
Queen Mary University of London
London, United Kingdom
thor@dcs.qmul.ac.uk

ABSTRACT

Today's integrated retrieval applications retrieve documents from disparate data sources. Therefore, as part of INEX 2004, we ran a heterogeneous track to explore the experimentation with a heterogeneous collection of documents. We built a collection comprising various sub-collections, reused topics (queries) from the sub-collections and created new topics, and participants submitted the results of retrieval runs. The assessment proved difficult, since pooling the results and browsing the collection posed new challenges and requested more resources than available. This report summarises the motivation, activities, results and findings of the track.

1. INTRODUCTION

A heterogeneous track has been part of INEX 2004. The task of the track was to explore how to build and maintain a testbed, how to create topics, and how to perform retrieval runs, assessment and evaluation.

1.1 Motivation

Before 2004, the INEX collection has been a collection of XML documents with a single DTD. However, in practical environments, XML retrieval requires to deal with XML documents with different DTDs, because a collection comprises documents of different purpose, authors and sources. Further, information in practical environments is spread over XML documents, relational databases, and other data source formats. Therefore, we included in INEX 2004 a heterogeneous track (het-track) that addressed the heterogeneity of a collection.

A heterogeneous collection poses a number of challenges:

- For content-only (CO) queries, approaches for homogeneous and well-typed collections can make direct use of the DTD. The DTD can be used, for example, for identifying what element type is reasonable to present in

the retrieval result. In a heterogeneous collection, we might have several or no DTD's, and retrieval methods independent of DTD are essential, and DTD mappings might be useful.

- For content-and-structure (CAS) queries, there is the problem of mapping structural conditions to different sub-collections. If we consider structural conditions as useful, then a DTD-based mapping of structural conditions is essential for CAS queries. Methods known for federated databases could be applied here. We can distinguish between manual, semi-automatic or fully automatic methods for creating the schema mappings.
- When performing retrieval runs, the retrieval algorithms need to merge the results retrieved from different sub-collections. For an experimental point of view, we can compare global strategies that know the whole collection with local strategies which make only use of the knowledge that can be derived per sub-collection. The latter strategies are probably closer to what we meet in reality.
- The content of a relational database can be represented in an XML document (collection, respectively). The question is whether the retrieval of relational databases via XML is beneficial.

The goal of the INEX het-track was to set up a test collection, and investigate the new challenges.

This track aims to answer, among others, the following research questions:

- For CO queries, what methods are feasible for determining elements that would be reasonable answers? Are pure statistical methods appropriate and sufficient, or are ontology-based approaches also helpful?
- What methods can be used to map structural criteria such that they can be applied (make sense) for a collection for which the DTD might be different or even not known!?
- Should mappings focus on element names (types) only, or also deal with element content?
- Should the data be organized (and indexed) as a single collection of heterogeneous documents, or is it better

to treat het-coll as a set of homogeneous subcollections?

- Are evaluation criteria developed for homogeneous collections also suitable for heterogeneous collections, or should other criteria and metrics be applied?

Since this was the first year of the heterogeneity track, the focus of the activities was on making a test collection available to participants, create some topics and perform retrieval runs and assessment, and apply evaluation measures.

The emphasis was on investigating the How to do it, with a detailed look at individual topics and runs, and the technicalities involved. A statistical measure has been not the aim of the first year of het-track.

1.2 Activities

The participants of this track carried out the following activities:

- Construction of a heterogeneous test collection: We used the current INEX corpus, and added various subcollections including DBLP, HCIBIB, Berkeley lib, Duisburg bibdb, and QMUL bibdb (the latter an XML representation of a relational database). The collection is maintained at <http://inex.is.informatik.uni-duisburg.de:2004/internal/hettrack/>.
- Selection of 20 CO and CAS queries from the existing INEX body and creation of four new topics. The topics were selected and created with the aim to retrieve documents from several sub-collections.
- INRIA has developed and experimented with a tool, XSum, for graphically representing XML documents; one of the main purposes of the tool was to enable the user to grasp the structure and aspect of various XML datasets, with or without a DTD. Currently, XSum represents the XML elements and attributes structure within an XML document, statistics such as numbers of elements on a given path. The tool is developed in Java, and freely available.
- Retrieval runs on the heterogeneous collection for this set of queries (see appendix).
- The assessment has been not carried out yet, due to technical problems and restricted resources. The aim is to join the het-coll with the relevance assessment tool used for the INEX IEEE collection.
- For the evaluation, we aim at a qualitative (query-and-run-oriented) analysis rather than a quantitative average-oriented analysis of results.

Based on the results and experience gained in 2004, a larger and quantitative het-track can be carried out in following years.

2. COLLECTION CREATION

The following table shows the subcollections that may have been used this year.

Collection	MB(unpacked)	No of elements
IEEE Computer Society	494	8.2M
Berkeley	33.1	1194863
CompuScience	313	7055003
bibdb Duisburg	2.08	40118
DBLP	207	5114033
hcibib	30.5	308554
qmul-dcs-pubdb	1.05	23436

From creating the subcollections, we have learned the following:

1. For a larger scale het-track, methods and tools are needed for managing a set of sub-collections. With restricted resources, the management of 5-10 sub-collections is achievable, more sub-collections will require tools and resources.
2. Sub-collections come with syntax errors (non-tidy XML). It is best to correct those errors centrally and “by hand”, but keep a carefully maintained log of the changes made.

3. TOPIC CREATION

Given the objectives of the het track, four types of topics have been proposed in the topic creation guideline:

1. CO (Content Only Topics): Since CO queries do not take structural information into account, this type had not been found challenging, however, any CO query used in the ad-hoc track could be used in the het track and gave similar results (because the test collection used for the ad-hoc track is part of the het track).
2. BCAS (Basic Content and Structure Topics): This type of topics focuses on the combination of singular structural constraints with a content-based constraint. The aim is synonym matches for structural constraints.
3. CCAS (Complex Content and Structure Topics): are the het track equivalent of the CAS topics of the ad-hoc track, specified used the NEXI language. The aim is to enable transformations and partial mappings of the topic path upon the different collections in het track, without losing the IR component of the topic.
4. ECCAS (Extended Content and Structure Topics): extended CCAS to enable the specification of the correctness path transformation and mapping probabilities.

3.1 Re-used Topics

Twenty topics were selected from the ad-hoc topics to reuse in het-track. After examining the ad-hoc topics, 10 CO topics were selected that probably contain results not only in the IEEE (also referred to and used as inex-1.3 and inex-1.4) subcollection. 10 CAS topics were also selected. The main criterion was that topics should possibly have relevant results in more subcollections. Selected CAS topics were identified as CCAS het track topics.

3.2 New Topics

Four new topics (see B) were created by participants of which three topics are CCAS and one is BCAS.

4. RETRIEVAL RUNS

The main difference between a mono- and a heterogeneous track is that sub-collections are specified in the run submissions. In order to be able to examine results with respect to the considered subcollections, a slightly modified version of the ad-hoc track's submission format has been proposed (see C).

Actually, the consideration of sub-collections poses some major research question, since we cannot assume that each run considers all subcollections:

1. How do we pool results from runs if some runs considered a sub-collection X and other runs considered a sub-collection Y?
2. How does an evaluation measure deal with the incompleteness of runs?

Another issue is the assignment of topics to participants. Is it useful to assign topics under strict rules and supervision, trying to make sure that sub-collections are covered equally, and the same number of runs is performed per topic, etc? Or is it the nature of heterogeneous track that this effort is not justified and is rather to be replaced by a random assignment?

5. ASSESSMENT AND EVALUATION

During the preparation for assessment and evaluation, we identified the following two main challenges:

1. Browsing the results and the collection. The browsing tool X-Rai was initially developed for the IEEE collection only, and currently cannot handle larger sub-collection files, even the QMUL subcollection with its 1.05MB, efficiently. Therefore, the two smallest sub-collections (bibdbpub and qmuldcdbpub) were converted into many small files, and made available for browsing.
2. Pooling. The aforementioned problem also affected the pooling procedure, the format of submission runs could not be exactly used for pooling. The other challenge in pooling was that, unlike the ad hoc track runs, het-track runs could consider various sets of subcollections, and there has not been a straightforward method to create pools from this kind of source, e.g. "use the first 150 results in each run" method may create larger pools for subcollections having more elements in the top-ranked results and small for those having less.

6. SUMMARY AND CONCLUSIONS

The first year of het-track established a heterogeneous collection, reused and created topics, and performed retrieval runs. The assessment and evaluation is currently outstanding; we intend to complete some of these at the December workshop.

The discussion among the participants and the work carried out raised the following questions:

1. What makes the heterogeneity of a collection? The current het-coll is viewed as little heterogeneous since it consists "only" of XML documents, and all documents are about computer science literature. Can we measure heterogeneity?
2. How can we manage many and large sub-collections? In particular creating the browsing facilities for the sub-collections and the assessment proved difficult. Can we easily split (and evtl. merge files)?
3. Topics and retrieval runs relate only to some sub-collections. Topics might have been created and runs might have been performed without considering the whole collection. How is this incompleteness captured in an evaluation?

Het-track has established a collection and experience about how to do it and where the difficulties are. INEX is now ready for the next phase of het-track, and it can re-use and extend the existing collection and pay particular attention to the efficient inclusion of new sub-collections into the whole process.

APPENDIX

A. TOPIC FORMAT

```
<!ELEMENT inex_topic (title,
  content_description,
  structure_description,
  narrative,keywords)>
<!ATTLIST inex_topic
  topic_id CDATA #REQUIRED
  query_type CDATA #REQUIRED
>

<!ELEMENT title (#PCDATA)>
<!ELEMENT content_description (#PCDATA)>
<!ELEMENT structure_description (#PCDATA)>
<!ELEMENT narrative (#PCDATA)>
<!ELEMENT keywords (#PCDATA)>
```

B. HET TRACK TOPICS

Topic created by IRIT:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<inex_topic topic_id="1" query_type="CCAS">
  <title>
    //bb[about(., "PhD thesis amsterdam")]
  </title>
  <content_description>
    I'm looking for bibliography entries concerning
    PhD thesis obtained at the university of Amsterdam
  </content_description>
  <structure_description>
    I'm looking for full references of PhD thesis: it
    means that results elements should contain the author,
    the title, the year and the school/city where the PhD
    thesis was obtained.
  </structure_description>
  <narrative>
    I'm maybe interested in working in Amsterdam next year
    and I would like to know what are the research subjects
    in the city. I think that a way to obtain this information
    (in the collections we have) is to see what are the subjects
    of the PhD thesis obtained in Amsterdam.
  </narrative>
  <keywords>
    phd thesis, university, amsterdam
  </keywords>
</inex_topic>
```

Topic created by UMONTES:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<inex_topic topic_id="2" query_type="CCAS">
  <title>
    //article[about(../author, nivio ziviani)]
  </title>
  <content_description>
    We are seeking for works with Nivio Ziviani as one of its authors
  </content_description>
  <structure_description>
    Title is a tag identifying works title and author is a
    tag identifying who wrote those works. They are usually part of
    front matter of a document, or part of bottom matter in a
    bibliography reference or can be an item in a volume index.
  </structure_description>
  <narrative>
    We are seeking for works with Nivio Ziviani as one of its
    authors. We want to catalogue all Nvio Ziviani works, so any
    reference, index entry , abstract or complete article will be
    relevant, but biography works will not.
  </narrative>
  <keywords>
    Nivio Ziviani
  </keywords>
</inex_topic>
```

Topic created by RMIT:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<inex_topic topic_id="3" query_type="CCAS">
  <title>
```

```
    //article[about(../abs, Web usage mining) or
    about(../sec, "Web mining" traversal navigation patterns)]
  </title>
  <content_description>
    We are looking for documents that describe capturing and mining
    Web usage, in particular the traversal and navigation patterns;
    motivations include Web site redesign and maintenance.
  </content_description>
  <structure_description>
    Article is a tag identifying a document, which can also be
    represented as a book tag, an inproceedings (or incollection)
    tag, an entry tag, etc. Abs is a tag identifying abstract of
    a document, which can be represented as an abstract tag, an abs
    tag, etc. Sec is a tag identifying an informative document
    component, such as section or paragraph. It can also be represented
    as sec, ssl, ss2, p, ip1 or other similar tags.
  </structure_description>
  <narrative>
    To be relevant, a document must describe methods for capturing
    and analysing web usage, in particular traversal and navigation
    patterns. The motivation is using Web usage mining for site
    reconfiguration and maintenance, as well as providing recommendations
    to the user. Methods that are not explicitly applied to the Web
    but could apply are still relevant.
    Capturing browsing actions for pre-fetching is not relevant.
  </narrative>
  <keywords>
    Web usage mining, Web log analysis, browsing pattern,
    navigation pattern, traversal pattern, Web statistics, Web design,
    Web maintenance, user recommendations.
  </keywords>
</inex_topic>
```

Topic created by LIP6:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<inex_topic topic_id="4" query_type="CCAS">
  <title>
    //article[about(., "text categorization") and
    (about(../fm//au, "David D. Lewis")
    or about(../bib//au, "David D. Lewis"))]
  </title>
  <content_description>
    I am looking for documents about text categorization which
    have been written by David D. Lewis, or related work from other authors
  </content_description>
  <structure_description>
    The tags which are used in this topic come from the DTD of the
    ad hoc task collection. Article is a tag identifying a document,
    which can also be represented as a book tag, an inproceedings
    (or incollection) tag, an entry tag, etc. Fm is a tag identifying
    the header of a document which usually contains title, authors...
    Bib is a tag identifying the bibliography of a document.
    Au is a tag identifying an author name.
  </structure_description>
  <narrative>
    To be relevant, a document must describe text categorization methods.
    It must have been written by David D. Lewis or must contain
    a bibliography entry with David D. Lewis.
  </narrative>
  <keywords>
    Text categorization, Text classifier
  </keywords>
</inex_topic>
```

C. RUN FORMAT

```
<!ELEMENT inex_het_track_submission (description, topic+)>
<!ATTLIST inex_het_track_submission
  participant-id CDATA #REQUIRED
  run-id CDATA #REQUIRED
  query (automatic | manual) #REQUIRED
  topic-part (T|D|K|TD|TK|DK|TDK) #IMPLIED
  task CDATA #IMPLIED
>

<!ELEMENT description (#PCDATA)>

<!ELEMENT topic (subcollections, result*)>
<!ATTLIST topic
  topic-id CDATA #REQUIRED
```

```

>
<!ELEMENT subcollections (subcollection+)>
<!ELEMENT result (subcollection, file, path, rank?, rsv?)>
<!ELEMENT subcollection EMPTY>
<!ATTLIST subcollection name CDATA #REQUIRED>

<!ELEMENT file (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT rank (#PCDATA)>
<!ELEMENT rsv (#PCDATA)>

```

D. SUBMITTED RUNS

- IRIT submitted 3 runs. One run is for CCAS, one for CO and one for BCAS topics. Files contain results for all the 24 het track topics. Various groups of subcollections were considered for topics.
- RMIT submitted results of three different approaches, all approaches were applied to all topic types and topics (9 files - file groups of 3 - one file is for a specific approach, specific topic type (CCAS,CO,BCAS)). Various groups of subcollections were considered for topics, often all subcollections were used.
- UBERKELEY submitted 2 runs, used all CO topics, 12 (i.e. all but one) CCAS topics. All subcollections were considered.
- UMONTES submitted 6 runs, 3 runs for all CO topics, 3 for all 'VCAS' (CCAS and BCAS together) topics, considered 5 subcollections.
- UNIDU submitted 3 runs, considered only topic no. 1 (as CO) and used 3 subcollections.

The Interactive Track at INEX 2004

Anastasios Tombros
Dept. of Computer Science
Queen Mary, University of London,
United Kingdom
tassos@dcs.qmul.ac.uk

Birger Larsen
Dept. of Information Studies
Royal School of LIS,
Copenhagen, Denmark
blar@db.dk

Saadia Malik
Fak. 5/IIS, Information Systems
University of Duisburg-Essen
Duisburg, Germany
malik@is.informatik.uni-duisburg.de

ABSTRACT

An interactive track was included in INEX for the first time this year, at INEX 2004. The main aim of the track was to study the behaviour of searchers when interacting with components of XML documents. In this paper, we describe the motivation and aims of the track in detail, we outline the methodology and we present some initial findings from the analysis of the results.

1. INTERACTIVE TRACK MOTIVATION

In recent years there has been a growing realisation in the IR community that the interaction of searchers with information is an indispensable component of the IR process. As a result, issues relating to interactive IR have been extensively investigated in the last decade. A major advance in research has been made by co-ordinated efforts in the interactive track at TREC. These efforts have been in the context of unstructured documents (e.g. news articles) or in the context of the loosely-defined structure encountered in web pages. XML documents, on the other hand, define a different context, by offering the possibility of navigating within the structure of a single document, or of following links to another document.

Relatively little research has been carried out to study user interaction with IR systems that take advantage of the additional features offered by XML documents, and so little is known about how users behave in the context of such IR systems. One exception is the work done by Finesliver and Reid [4], who studied end user interaction with a small test collection of Shakespeare's plays formatted in XML.

The investigation of the different context that is defined in the case of user interaction with XML documents has provided the main motivation for the establishment of an interactive track at INEX. The main aims for the interactive track are 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.

In the first year, we focused on investigating the behaviour of searchers when presented with components of XML documents that have a high

probability of being relevant (as estimated by an XML-based IR system). Presently, metrics that are used for the evaluation of system effectiveness in the INEX ad-hoc track are based on certain assumptions of user behaviour [7]. These metrics attempt to quantify the effectiveness of IR systems at pointing searchers to relevant elements of documents. Some of the assumptions behind the metrics include that users would browse through retrieved elements in a linear order, that they would "jump" with a given probability p from one element to another within the same document's structure, that they would not make use of links to another document, etc. These assumptions have not been formally investigated in the context of XML retrieval; their investigation formed the primary aim for the first year of the interactive track.

Since the investigation of user behaviour forms our primary focus, the format of the track for the first year differs to that typically followed by, for example, the interactive track at TREC. The main difference was that a comparison between different interactive approaches was not our main focus. Instead, a more collaborative effort was planned, with the outcome of the studies expected to feed back to the INEX initiative. Participating sites still had the option to develop and evaluate their own interactive approaches, but this was not a requirement for participation. It should be noted that none of the participating sites opted to develop their own system.

We first describe the experimental setup and methodology in section 2, then we present an initial analysis of the data in section 3, and we conclude in section 4.

2. EXPERIMENTAL SETUP

In this section we outline the experimental set up for the first interactive track at INEX.

2.1 Topics

We used content only (CO) topics from the INEX 2004 collection. We added an additional dimension to the investigation of this year's interactive track by selecting topics that corresponded to different types of tasks. The effect that the context determined by task type has on the behaviour of online searchers has been demonstrated in a number of studies [e.g. 8].

One way to categorise tasks is according to the "type" of information need they correspond to. In [8] the

categorisation included background (find as much general information on a topic as possible), decision (make a decision based on the information found) and many-items task (compile a list of items related to the information need) types. It was shown that different task types promote the use of different criteria when assessing the relevance of web pages. It is likely that a similar effect, in terms of user behaviour within structured documents, may exist in the context of XML documents. Searchers may exhibit different browsing patterns and different navigational strategies for different task types.

Four of the 2004 CO topics were used in the study, and they were divided into two task categories:

- *Background category (B)*: Most of the INEX topics fall in this category. The topics express an information need in the form of “I’d like to find out about X”. The two tasks in this category were based on topics 180 and 192.
- *Comparison category (C)*: There are a number of topics whose subject is along the lines of: “Find differences between X and Y”. The tasks given in this category were based on topics 188 and 198.

In order to make the tasks comprehensible by other than the topic author, it was required that all INEX 2004 topics not only detail *what* is being sought for, but also *why* this is wanted, and in what *context* the information need has arisen. Thereby the INEX topics are in effect simulated work task situations as developed by Borlund [5, 6]. Compared to the regular topics, more context on the motives and background of the topic is provided in the simulated work tasks. In this way, the test persons can better place themselves in a situation where they would be motivated to search for information related to the work tasks. The aim is to enable the test persons to formulate and reformulate their own queries as realistically as possible in the interaction with the IR system. The task descriptions used in the study were derived from part of the Narrative field. We include the task descriptions as given to searchers in the Appendix.

2.2 System

A system for the interactive track study was provided by the track organisers. The system was based on the HyRex¹ retrieval engine, and included a web-based interface with a basic functionality.

Searchers were able to input queries to the system. In response to the query, HyRex returns a ranked list of components as shown in Figure 1. The information presented for each retrieved component included the title of the component and the authors, its retrieval value and the XPath of the component. Searchers can explore the ranked list of components, and can visit components by clicking on the component title in the ranked list.

¹ <http://www.is.informatik.uni-duisburg.de/projects/hyrex/>

In Figure 2 we show the detailed component view. This view is divided into two parts: the right hand of the view includes the actual textual contents of the selected component; the left side contains the table of contents for the document containing the component. Searchers can access other components within the same document either by using the table of contents on the left, or by using the next and previous buttons at the top of the right part of the view. A relevance assessment for each viewed component could be given, as shown in Figure 2. The assessment was based on two dimensions of relevance: how useful and how specific the component was in relation to the search task. The definition of usefulness was formulated very much like the one for Exhaustivity in the Ad hoc track, but was labelled usefulness, which might be easier for users to comprehend. Each dimension had three grades of relevance as this is shown in Figure 2. Ten possible combinations of these dimensions could be made.

A	Very useful & Very specific
B	Very useful & Fairly specific
C	Very useful & Marginally specific
D	Fairly useful & Very specific
E	Fairly useful & Fairly specific
F	Fairly useful & Marginally specific
G	Marginally useful & Very specific
H	Marginally useful & Fairly specific
I	Marginally useful & Marginally specific
J	Contains no relevant information
U	Unspecified

Table 1. The applied relevance scale

To return to the ranked list, searchers would need to close the currently open document.

A different version of the system with graphical features was also developed. This system (Graphical system) differed to the Baseline system both in the way of presenting the ranked list (Figure 3) and in the way of presenting the detailed view of components (Figure 4). The graphical system retrieves documents rather than components, and presents the title and authors of each retrieved document. In addition, it also presents a shaded rectangle (the darker the colour the more relevant the document to the query) and a red bar (the longer the bar the more query hits are contained in the document).

The detailed view for each selected document component is similar to that for the Baseline system, with the addition of a graphical representation at the top of the view (Figure 4). A document is represented in a rectangular area and is split horizontally and vertically to represent the different document levels. Tooltips (on mouse-over) provide additional information about the retrieved components, such as the first 150 characters of the contents and the component's name, the selected section, subsection, etc. On the top part of the this view, all the retrieved

documents are shown as small rectangles in gray shades along with the *Next* and *Previous* links to allow navigation between the retrieved results.

2.3 Participating sites

The minimum requirement for sites to participate in this year's interactive track was to provide runs using 8 searchers on the Baseline version of the XML retrieval system that the track organisers provided. In addition to the minimum requirement, sites could choose to employ more users, to expand the experimental design by comparing both versions of the system (baseline and graphical), or to test their own experimental system against the baseline system provided.

Ten sites participated in the Interactive track. In the following table we give the sites' name, number of searchers used and types of comparisons performed.

Site	Baseline system	Additional studies
Oslo University College, Norway	8 users	-
RMIT, Australia	16 users	-
U. Twente/CWI, The Netherlands	8 users	8 users (baseline vs. graphical)
Norwegian University of Science and Technology, Norway	8 users	-
U. Tampere, Finland	8 users	-
Kyungpook National University, Korea	8 users	-
Robert Gordon University, Scotland	8 users	-
University Duisburg-Essen, Germany	8 users	-
Royal School of Library and Information Science, Denmark	8 users	-
Queen Mary, University of London, England	8 users	-

Table 2. Participating sites in the Interactive Track

2.4 Experimental protocol

A minimum of 8 searchers from each participating site were used. Each searcher searched on one task from each task category. The task was chosen by the searcher. The order in which task categories are performed by searchers was permuted. This means that one complete round of the experiment requires only 2 searchers. The minimum experimental matrix consisted of the following 2x2 block:

Searcher	1 st Task category	2 nd Task category
1	Background (B)	Comparison (C)
2	Comparison (C)	Background (B)

Table 3. Basic experimental matrix

This block was repeated 4 times for the minimum requirements for participation. This matrix could be augmented by adding blocks of 4 users (a total of 12, 16, 20, etc. users).

For the comparison of the baseline and the graphical systems, searchers would be involved in the study in addition to the ones used only for the baseline system.

The experimental matrix in this case consisted of the following blocks of system-task conditions:

Searcher	1 st Condition	2 nd Condition
1	Graphical-B	Baseline-C
2	Graphical-C	Baseline-B
3	Baseline-B	Graphical-C
4	Baseline-C	Graphical-B

Table 4. Augmented experimental matrix

The order of an experimental session was as follows:

1. Introduction: Briefing about the experiment and procedures
2. Before-experiment questionnaire
3. Hand out Instructions for Searchers
4. System tutorial
5. Task selection from the appropriate category
6. Before-task questionnaire
7. Search session
8. After-task questionnaire
9. Repeat steps 5-8 for the other task category
10. After-experiment questionnaire
11. Informal discussion/interview: any additional views on the experiment, system, etc. the searcher wishes to share.

Each searcher was given a maximum of 30 minutes to complete each task. The goal for each searcher was to locate sufficient information towards completing a task.

2.5 Data collection

The collected data comprised questionnaires completed by the test persons, the logs of searcher interaction with the system, the notes experimenters kept during the sessions and the informal feedback provided by searchers at the end of the sessions.

The logged data consisted of the queries issued, the components returned by the system, the components actually viewed and the order in which they were viewed, relevance assessments of these, any browsing behaviour, as well as time stamps for each interaction between searchers and the system.

3. INITIAL RESULTS ANALYSIS

In this section we present an initial analysis of the collected data. In section 3.1 we analyse data collected from the questionnaires, then in section 3.2 we present some general statistics collected from the system logs, and in section 3.3 we outline the detailed analysis of browsing behaviour which is currently in progress.

3.1 Questionnaire data

A total of 88 searchers were employed by participating sites. The average age of the searchers was 29 years. Their average experience in

bibliographic searching in online digital libraries, computerised library catalogs, WWW search engines etc. was 4, on a scale from 1 to 5 with 5 signifying highest experience level. The education level of the participants spanned undergraduate (39%), MSc (49%), and PhD (12%) levels.

In terms of task selection, from the Background task category 66% of participants selected task B1 (cybersickness, topic 192) and 34 % selected B2 (ebooks, topic 180). From the Comparison task category, 76% selected task C2 (Java-Python, topic 198) and 24% selected task C1 (Fortran90-Fortran, topic 188).

In Table 5 we present data for task familiarity, task difficulty and perceived task satisfaction. With respect to task familiarity, we asked searchers before the start of each search session to rate how familiar they were with the task they selected on a scale from 1 to 5, with 5 signifying the greatest familiarity. With respect to task difficulty, we asked searchers to rate the difficulty of the task once before the start of the search session, and once the session was completed (pre- and post- task difficulty, columns 3 and 4 respectively). Searchers also indicated their satisfaction with the results of the task. All data in Table 5 correspond to the same 5-point scale.

	Task familiarity	Pre-task difficulty	Post-task difficulty	Task satisfaction
B1 (no.192)	2.1	2.03	1.47	3.39
B2 (no.180)	2.73	2.1	1.97	1.97
C1 (no.188)	2.67	1.95	1.74	2.62
C2 (no.198)	2.91	2.1	1.52	2.9

Table 5. Searchers' perceptions of tasks

The data in Table 5 suggest that there are some significant differences in the searchers' perceptions of the tasks. The most notable of these differences are in task familiarity and task satisfaction. It should be noted that at this time a thorough statistical analysis of the results has not been performed. An initial analysis of the correlation between task familiarity and satisfaction did not show a strong relationship between these two variables across the tasks.

The overall opinion of the participants about the Baseline system was recorded in the final questionnaire they filled in after the completion of both tasks. Participants generally felt at ease with the system, finding it easy to learn how to use (average rating 4.17), easy to use (3.95) and easy to understand (3.94). There were also many informal comments by the participants about specific aspects of the system. These comments were recorded by the experimenters and will be analysed at a later stage.

3.2 General statistics

This analysis concerns approximately 50 % of the log data for the baseline system. The remainder could not be analysed reliably at present because of problems with the logging software.

Ranks

A maximum of 100 hits were presented to searchers on the ranked list, and they were free to choose between these in any order they liked (See Figure 1). For the Background (B) tasks 86 % of the viewed components were from top10 of the ranked list (80 % for the Comparison (C) tasks). The ranks viewed furthest down the list were 71 for B and 96 for C.

Queries

The possible query operators were '+' for emphasis, '-' for negative emphasis, and " " for phrases. The phrase operator was 24 used times in B, and 16 in C. No one used plus or minus. 217 unique queries were given for B, and 225 for C across all searchers. On average, the queries for B consisted of 3.0 search keys (counting a phrase as one search key), and 3.4 for C including stop words. 81 % of the queries for B consisted of 2, 3 or 4 search keys for B, 80 % for C.

Viewed components

In total, searchers viewed 804 different components for B, and 820 for C. On average this was 10.9 unique components viewed for B, and 10.8 for C.

Three possibilities existed for accessing a component: to click a hit from the ranked list, to click a part of the document structure (via the table of contents), and to use the next/previous buttons. From Table 6 below it can be seen that very few chose to use the next/previous buttons: only 2 % of component viewing arose from this (both B and C). For B 63 % of viewings came from the ranked list, for C this was 62 %. For B 35 % came from the table of contents, and 37 % for C.

Access	B	C	Total	B	C
nextprev	17	17	34	2%	2%
rankedlist	588	550	1138	63%	62%
structure	327	327	654	35%	37%
Total	932	894	1826	100%	100%

Table 6. Access modes to viewed components

Assessed components

503 components were assessed for B, 489 for C, or 6.8 per searcher per task for B, and 6.4 for C. This corresponds to 63 % of the viewed components for B and 60 % for C.

The distribution of relevance assessments on tasks can be seen in Table 7 below. It may be observed that 12-13 % of the assessed documents were 'Very useful & Very specific' [A] for both B and C, and that 15-16 % of the assessed documents were 'Marginally useful & Marginally specific' [I] for both B and C. The most noteworthy difference is that B had 38 % non-relevant assessments [J], and C only 17 %.

Relevance	B	C	Total	B	C
A	65	61	126	13%	12%
B	28	36	64	6%	7%
C	8	13	21	2%	3%
D	19	45	64	4%	9%
E	36	61	97	7%	12%
F	28	38	66	6%	8%
G	12	20	32	2%	4%
H	33	47	80	7%	10%
I	79	80	159	16%	16%
J	191	84	275	38%	17%
U	4	4	8	1%	1%
Total	503	489	992	100%	100%

Table 7. Relevance assessments distributed on task type (see Table 1 above for relevance scale)

The next two tables show the distribution of relevance assessments on the access possibilities, one for B and one for C (i.e. how did the searchers reach the components which they assessed). The total number of component viewings with relevance assessments is lower (992) than the total number of components viewed (1826, Table 6) because not all viewed components were assessed.

Relevance	nextprev	rankedlist	structure	Total
A	1	38	26	65
B	-	16	12	28
C	-	4	4	8
D	-	11	8	19
E	-	21	15	36
F	-	21	7	28
G	-	10	2	12
H	2	23	8	33
I	1	45	33	79
J	1	142	48	191
U	-	4		4
Total	5	335	163	503

Table 8. Relevance assessments distributed on access modes for the B tasks

Relevance	nextprev	rankedlist	structure	Total
A	-	43	18	61
B	-	25	11	36
C	-	11	2	13
D	-	30	15	45
E	-	34	27	61
F	-	26	12	38
G	-	13	7	20
H	-	35	12	47
I	-	60	20	80
J	-	64	20	84
U	-	4		4
Total	0	345	144	489

Table 9. Relevance assessments distributed on access modes for the C tasks

For both B and C very few viewings with next/previous section buttons resulted in assessments: 0 for C, and 5 for B. The latter 5 were given low assessments. In both cases the majority of assessments resulted as a direct consequence of clicking a hit from the ranked list: 67% for B and 71% for C. Apart from 1 % next/previous navigation in B the remainder the rest is taken up by navigation from the table of contents. Large variations are, however, obvious in the data, and can be uncovered by an in-depth analysis of the browsing behaviour.

Overall browsing behaviour

Table 10 shows this variation on an overall level by counting the number of requests for components within the same document. The raw figures included double counting, because whenever an assessment was made the component was reloaded from the server. In this table, the number of assessments has therefore been subtracted from the number of requests for components. It can be seen that for the most part (70% of cases) searchers viewed 1 component and assessed it (or viewed two and didn't assess any), and then moved on to a new document rather than continuing the navigation within the same document.

	B	C	Total	B	C
1	406	394	800	69.0%	71.6%
2	93	84	177	15.8%	15.3%
3	47	39	86	8.0%	7.1%
4	23	9	32	3.9%	1.6%
5	13	8	21	2.2%	1.5%
6	2	4	6	0.3%	0.7%
7	2	5	7	0.3%	0.9%
8	1	1	2	0.2%	0.2%
9	1		1	0.2%	0.0%
10		1	1	0.0%	0.2%
11		2	2	0.0%	0.4%
12		1	1	0.0%	0.2%
13		1	1	0.0%	0.2%
14		1	1	0.0%	0.2%
Total	588	550	1138	100%	100%

Table 10. Overall browsing behaviour within the same document: number of components viewed

A more in-depth analysis of the data will be performed with the aim to further break down user browsing behaviour within an accessed document. From informal comments made by searchers, and from an initial observation of the log data, one possible reason for the low degree of interaction with documents and their components was overlap. Searchers generally recognised overlapping components, and found them an undesirable "feature" of the system. Through more detailed analysis of the logs we can determine how searchers behaved when the system returned overlapping components.

3.3 Detailed browsing behaviour

A detailed analysis on the browsing behaviour of searchers is currently underway. The main aim of this analysis is to determine how users browsed within each document they visited, and how their browsing actions correlated with their relevance assessments. More specifically, we aim to look into the relationship of the relevance assessments' dimensions to whether searchers browse to more specific or more general components in the document tree, whether they browse to components of the same depth or whether they return to the ranked list of components. For example, we could see where users would browse to after they have assessed a component as "Very useful and fairly specific", and also how they would assess further documents along the browsing path.

This detailed analysis, together with the analysis on the overlapping components, can yield results that can be useful for the development of metrics that may take into account actual indications of user behaviour.

4. CONCLUSIONS

In this paper we described the motivation and aims, and the methodology of the INEX 2004 interactive track. We also presented some initial results gathered from user questionnaires and system logs.

We are currently performing a more detailed analysis of the gathered data, with the aim to establish patterns of browsing behaviours and to correlate them to the assessments of the visited document components. This analysis can also provide insight as to whether there are different browsing behaviours for the two different task categories included in the study. We expect that the results of this analysis will lead to the development of effectiveness metrics based on observed user behaviour.

5. REFERENCES

- [1] Chieramella, Y. (2001): Information retrieval and structured documents. In: *Lectures on information retrieval : third European summer-school, ESSIR 2000, Varenna, Italy: Revised Lectures*. Berlin: Springer, pp 286-309.
- [2] Fuhr, N., Gövert, N., Kazai, G. and Lalmas, M. (2002): INEX : initiative for the evaluation of XML retrieval. In: *ACM SIGIR'2002 workshop on XML and information retrieval*, pp 62-70.
- [3] Gövert, N. and Kazai, G. (2003): Overview of the initiative for the evaluation of XML retrieval (INEX) 2002. In: *Proceedings of the 1st workshop of the initiative for the evaluation of XML retrieval (INEX)*, pp 1-17.
- [4] Finesilver, K. and Reid, J. (2003): User Behaviour in the Context of Structured Documents. In: *Advances in Information Retrieval: 25th European Conference on IR Research, ECIR 2003*, pp 104-119.
- [5] Borlund, P. (2000): *Evaluation of interactive information retrieval systems*. Åbo: Åbo Akademi University Press. vi, 276 p. (PhD dissertation).
- [6] 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>]
- [7] Kazai, G. (2003): Report of the INEX 2003 metrics working group. In: *Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX)*, pp 184-190.
- [8] Tombros, A., Ruthven, I. and Jose, J (2004): Searchers criteria for assessing web pages. In: *Journal of the American Society for Information Science and Technology*. In press.

APPENDIX

A. HyRex Retrieval System Screenshots

dbdk_training in Baseline System

Search

query was: text classification naive bayes
Results 1 - 10 of 100.
Result pages: 1 2 3 4 5 6 7 8 9 10 next

Search Result

- (0.247) **Scalable Feature Mining for Sequential Data**
Neal Lesh Mitsubishi Electric Research Lab Mohammed J. Zaki Rensselaer Polytechnic Institute Mitsunori Ogihara University of Rochester
Result path: /article[1]/bdy[4]/sec[5]
- (0.204) **Probability and Agents**
Marco G. Valtorta University of South Carolina mgv@cse.sc.edu Michael N. Huhns University of South Carolina huhns@sc.edu
Result path: /article[1]/bdy[4]/sec[3]
- (0.176) **Combining Image Compression and Classification Using Vector Quantization**
Karen L. Oehler Member IEEE Robert M. Gray Fellow IEEE
Result path: /article[1]/bdy[4]/sec[4]/ss1[2]/ss2[4]
- (0.175) **Text-Learning and Related Intelligent Agents: A Survey**
Dunja Mladenic J. Stefan Institute
Result path: /article[1]/hm[5]/app[4]/sec[5]
- (0.175) **Detecting Faces in Images: A Survey**
Ming-Hsuan Yang Member IEEE David J. Kriegman Senior Member IEEE Narendra Ahuja Fellow IEEE
Result path: /article[1]/bdy[4]/sec[2]/ss1[9]/ss2[10]

Figure 1. The ranked list of documents in the Baseline system

Table of Contents

- 1 Introduction
- 2 Detecting faces in a single image
 - 2.1 Knowledge-Based Top-Down Methods
 - 2.2 Bottom-Up
 - 2.2.1 Facial Features
 - 2.2.2 Texture
 - 2.2.3 Skin Color
 - 2.2.4 Multiple Features
 - 2.3 Template Matching
 - 2.3.1 Predefined Templates
 - 2.3.2 Deformable Templates
 - 2.4 Appearance-Based Methods
 - 2.4.1 Eigenfaces
 - 2.4.2 Distribution-Based Methods
 - 2.4.3 Neural Networks
 - 2.4.4 Support Vector Machines
 - 2.4.5 Sparse Network of Winnows
 - 2.4.6 Naive Bayes Classifier
 - 2.4.7 Hidden Markov Model
 - 2.4.8 Information-Theoretical Approach
 - 2.4.9 Inductive Learning
 - 2.5 Discussion
- 3 Face image databases and performance evaluation

To which extent this piece of information covers your problem or topic of interest:

Unspecified submit

2.4.6 NaiveBayes Classifier

In contrast to the methods in [[107]], [[120]], [[154]] which model the global appearance of a face, Schneiderman and Kanade described a NaiveBayes classifier to estimate the joint probability of local appearance and position of face patterns (subregions of the face) at multiple resolutions [[140]]. They emphasize local appearance because some local patterns of an object are more unique than others; the intensity patterns around the eyes are much more distinctive than the pattern found around the cheeks. There are two reasons for using a NaiveBayes classifier (i.e., no statistical dependency between the subregions). First, it provides better estimation of the conditional density functions of these subregions. Second, a NaiveBayes classifier provides a functional form of the posterior probability to capture the joint statistics of local appearance and position on the object. At each scale, a face image is decomposed into four rectangular subregions. These subregions are then projected to a lower dimensional space using PCA and quantized into a finite set of patterns, and the statistics of each projected subregion are estimated from the projected samples to encode local appearance. Under this formulation, their method decides that a face is present when the likelihood ratio is larger than the ratio of prior probabilities. With an error rate of 93.0 percent on data set 1 in [[128]], the proposed Bayesian approach shows comparable performance to [[128]] and is able to detect some rotated and profile faces. Schneiderman and Kanade later extend this method with wavelet representations to detect profile faces and cars [[141]].

A related method using joint statistical models of local features was developed by Rickert et al. [[124]]. Local features are extracted by applying multiscale and multiresolution filters to the input image. The distribution of the features vectors (i.e., filter responses) is estimated by clustering the data and then forming a mixture of Gaussians. After the model is learned and further refined, test images are classified by computing the likelihood of their feature vectors with respect to the model. Their experimental results on face and car detection show interesting and good results.

To which extent this piece of information covers your problem or topic of interest:

Unspecified submit

- Very useful & Very specific
- Very useful & Fairly specific
- Very useful & Marginally specific
- Fairly useful & Very specific
- Fairly useful & Marginally specific**
- Marginally useful & Very specific
- Marginally useful & Fairly specific
- Marginally useful & Marginally specific
- Contains no relevant information

Figure 2. Detailed view of document components in the Baseline system

dbdk_training in Graphical System

Search

query was: text classification naive bayes
Results 1 - 10 of 61.
Result pages: 1 2 3 4 5 6 7 next

HyREX

Search Results

- Scalable Feature Mining for Sequential Data**
Neal Lash Mitsubishi Electric Research Lab Mohammed J. Zaki Rensselaer Polytechnic Institute Mitsunori Ogihara University of Rochester
- Probability and Agents**
Marco G. Valtorta University of South Carolina mgv@cse.sc.edu Michael N. Huhns University of South Carolina huhns@sc.edu
- Combining Image Compression and Classification Using Vector Quantization**
Karen L. Oehler Member IEEE Robert M. Gray Fellow IEEE
- Text Learning and Related Intelligent Agents: A Survey**
Dunja Mladenic J. Stefan Institute
- Detecting Faces in Images: A Survey**
Ming-Hsuan Yang Member IEEE David J. Kriegman Senior Member IEEE Narendra Ahuja Fellow IEEE

Figure 3. The ranked list of documents in the Graphical system

HyREX Close Document

Previous [Progress Bar] Next

Detecting Faces in Images: A Survey

Section
Detecting Faces in A Single Image In this section, we review existing techniques to detect faces from a single intensity or color image. We classify...

1 Introduction

2 Detecting faces in a single image

- 2.1 Knowledge-Based Top-Down Methods
- 2.2 Bottom-Up Feature-Based Methods
 - 2.2.1 Facial Features
 - 2.2.2 Texture
 - 2.2.3 Skin Color
 - 2.2.4 Multiple Features
 - 2.3 Template Matching
 - 2.3.1 Predefined Templates

To which extent this piece of information covers your problem or topic of interest:

Unspecified submit

Very useful & Very specific

Very useful & Fairly specific

Very useful & Marginally specific

Fairly useful & Very specific

Fairly useful & Fairly specific

Fairly useful & Marginally specific

Marginally useful & Very specific

Marginally useful & Fairly specific

Marginally useful & Marginally specific

Contains no relevant information

[[128]], [[154]] which model the global appearance of a face, NaiveBayes classifier to estimate the joint probability of local appearance ns of the face) at multiple resolutions [[140]]. They emphasize local ns of an object are more unique than others; the intensity patterns found in the pattern found around the cheeks. There are two reasons for using a ll dependency between the subregions). First, it provides better estimation these subregions. Second, a NaiveBayes classifier provides a functional form the joint statistics of local appearance and position on the object. At each scale, a face image is decomposed into four rectangular subregions. These subregions are then projected to a lower dimensional space using PCA and quantized into a finite set of patterns, and the statistics of each projected subregion are estimated from the projected samples to encode local appearance. Under this formulation, their method decides that a face is present when the likelihood ratio is larger than the ratio of prior probabilities. With an error rate of 93.0 percent on data set 1 in [[128]], the proposed Bayesian approach shows comparable performance to [[128]] and is able to detect some rotated and profile faces. Schneiderman and Kanade later extend this method with wavelet representations to detect profile faces and cars [[141]].

A related method using joint statistical models of local features was developed by Rickert et al. [[124]]. Local features are extracted by applying multiscale and multiresolution filters to the input image. The distribution of the features vectors (i.e., filter responses) is estimated by clustering the data and then forming a mixture of

Figure 4. Detailed view of document components in the Graphical system

B. Task Descriptions

Task category: Background (B)

Task ID: B1

You are writing a large article discussing virtual reality (VR) applications and you need to discuss their negative side effects. What you want to know is the symptoms associated with cybersickness, the amount of users who get them, and the VR situations where they occur. You are not interested in the use of VR in therapeutic treatments unless they discuss VR side effects.

Task ID: B2

You have tried to buy & download electronic books (ebooks) just to discover that problems arise when you use the ebooks on different PC's, or when you want to copy the ebooks to Personal Digital Assistants. The worst disturbance factor is that the content is not accessible after a few tries, because an invisible counter reaches a maximum number of attempts. As ebooks exist in various formats and with different copy protection schemes, you would like to find articles, or parts of articles, which discuss various proprietary and covert methods of protection. You would also be interested in articles, or parts of articles, with a special focus on various disturbance factors surrounding ebook copyrights.

Task category: Comparison (C)

Task ID: C1

You have been asked to make your Fortran compiler compatible with Fortran 90, and so you are interested in the features Fortran 90 added to the Fortran standard before it. You would like to know about compilers, especially compilers whose source code might be available. Discussion of people's experience with these features when they were new to them is also of interest.

Task ID: C2

You are working on a project to develop a next generation version of a software system. You are trying to decide on the benefits and problems of implementation in a number of programming languages, but particularly Java and Python. You would like a good comparison of these for application development. You would like to see comparisons of Python and Java for developing large applications. You want to see articles, or parts of articles, that discuss the positive and negative aspects of the languages. Things that discuss either language with respect to application development may be also partially useful to you. Ideally, you would be looking for items that are discussing both efficiency of development and efficiency of execution time for applications. You would like a good comparison of these for application development. You would like to see comparisons of Python and Java for developing large applications. You want to see articles, or parts of articles, that discuss the positive and negative aspects of the languages. Things that discuss either language with respect to application development may be also partially useful to you. Ideally, you would be looking for items that are discussing both efficiency of development and efficiency of execution time for applications.

Reliability Tests for the XCG and inex-2002 Metrics

Gabriella Kazai
Queen Mary University of London
gabs@dcs.qmul.ac.uk

Mounia Lalmas
Queen Mary University of London
mounia@dcs.qmul.ac.uk

Arjen de Vries
CWI
arjen@acm.org

Abstract

In this paper we compare the effectiveness scores and system rankings obtained with the inex-2002 and the XCG metrics. For the comparisons, we use simulated runs as we can then easily derive the desired system rankings based on a predefined set of user preferences. The results indicate that the XCG metric is better suited for comparing systems for the INEX content-only (CO) task, where systems aim to return the highest scoring elements according to the user preferences reflected in a quantisation function, while also aiming to avoid returning overlapping components.

1 INTRODUCTION

The official metric of INEX 2004 is the `inex_eval` or, as referred here, the `inex-2002` metric. This metric has been chosen by INEX as the official measure partly because at the time it was still not clear how much its known weaknesses would effect the overall system rankings and partly because alternative measures were not yet ready to take this role. Some of the known weaknesses were reported in [4, 5]. One such issue is that the metric does not take into account the overlap between result elements and hence produces better effectiveness scores for systems that return multiple nested components, e.g. a paragraph and its container section and article. At the INEX 2003 workshop, it was agreed that such a system behaviour should not be rewarded, but in fact should be penalised [4]. Another issue with the `inex-2002` metric is that it calculates recall based on the full recall-base, which also contains large amounts of overlapping components. This means that 100% recall can only be reached by systems that return all elements, including all overlapping components, in the full recall-base. For systems that aim to avoid returning overlapping, and hence redundant, elements to the user, the affect of the latter issue is that the precision scores get plotted against lower recall values than merited [5].

An argument for the `inex-2002` metric could be that it can produce reliable rankings of systems provided none of the systems retrieve overlapping result elements. Although the effectiveness scores would still reflect a pessimistic estimate of performance (due to the overlap amongst the reference elements in the full recall-base), the relative ranking of systems could provide a true reflection with respect to the evaluation criterion.

However, most of the current systems at INEX output result lists, where high overlap ratios in the region of 70-80% are not uncommon. This then raises the question whether we can trust the scores obtained by the `inex-2002` metric.

In this paper, we investigate this question by means of a basic reliability test. We refer to the reliability test of this study as “basic”, as we do not provide here a comprehensive survey of acceptable error rates and significant differences in effectiveness scores, etc., but concentrate only on evaluating “a metric’s ability to rank a better system ahead of a worse system” [8]. We test two metrics, the `inex-2002` metric and the XCG metric proposed in [5] and further developed in this paper. For the comparisons, we use simulated runs instead of the actual INEX runs submitted by participants. The reason for this is that by controlling which elements and in what order should form a run, we can get clearer conclusions regarding the two metrics’ behaviours.

In the following, we first give a quick overview of the two metrics (Section 2) and then describe the setup and results of our metric reliability test (Section 3). We close with conclusions in Section 4.

2 THE METRICS

This section gives a brief summary of the `inex-2002` (aka. `inex_eval`) [2] and XCG [5] metrics.

2.1 The *inex-2002* metric

The *inex-2002* metric applies the measure of *precall* [7] to document components and computes the probability $P(\text{rel}|\text{retr})$ that a component viewed by the user is relevant:

$$P(\text{rel}|\text{retr})(x) := \frac{x \cdot n}{x \cdot n + \text{esl}_{x \cdot n}} \quad (1)$$

where $\text{esl}_{x \cdot n}$ denotes the *expected search length* [1], i.e. the expected number of non-relevant elements retrieved until an arbitrary recall point x is reached, and n is the total number of relevant components with respect to a given topic.

To apply the above metric, the two relevance dimensions are first mapped to a single relevance scale by employing a quantisation function, $\mathbf{f}_{\text{quant}}(e, s): ES \rightarrow [0, 1]$, where ES denotes the set of possible assessment pairs (e, s) :

$$ES = \{(0, 0), (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)\}$$

There are a number of quantisation functions currently in use in INEX, e.g. strict or generalised (see Equations 2 and 3 in [4]), each representing a different set of user preferences. In this paper we concentrate on the “specificity-oriented generalised” (*sog*) quantisation function proposed in [5]:

$$\mathbf{f}_{\text{sog}}(e, s) := \begin{cases} 1 & \text{if } (e, s) = (3, 3) \\ 0.9 & \text{if } (e, s) = (2, 3) \\ 0.75 & \text{if } (e, s) \in \{(1, 3), (3, 2)\} \\ 0.5 & \text{if } (e, s) = (2, 2) \\ 0.25 & \text{if } (e, s) \in \{(1, 2), (3, 1)\} \\ 0.1 & \text{if } (e, s) \in \{(2, 1), (1, 1)\} \\ 0 & \text{if } (e, s) = (0, 0) \end{cases} \quad (2)$$

The argument in [5] is that the relative ranking of assessment value pairs in the above formula better reflects the evaluation criterion for XML retrieval as defined within the CO task. According to this, specificity plays a more dominant role than exhaustivity. This is not the case for the generalised quantisation function, which shows slight preference towards exhaustivity, assigning high scores to exhaustive, but not necessarily specific components. Due to the propagation effect and the cumulative property of exhaustivity, such components are generally large, e.g. *body* or *article*, elements. This means that relatively high effectiveness scores could be achieved with simple article runs, which contradicts the goal of the retrieval task. The *sog* mapping overcomes this bias.

Like all quantisation functions, the *sog* quantisation captures a relative ranking of exhaustivity-specificity value pairs reflecting user preferences, such that, e.g., $(e, s) = (3, 3)$ nodes are preferred to $(e, s) = (2, 3)$ nodes, which in turn are better than $(e, s) \in \{(1, 3), (3, 2)\}$ nodes and so on.

2.2 The XCG metrics

The XCG metrics are extensions of the cumulated gain (CG) based metrics proposed by Järvelin and Kekäläinen in [3]. The motivation for the CG metrics was to develop a measure for multi-grade relevance values, i.e. to credit IR systems according to the retrieved documents’ degree of relevance. The motivation for XCG was to extend CG in such a way that the problem of overlapping result and reference elements can be addressed within the evaluation framework.

2.2.1 A brief recap on the CG metrics

The Cumulated Gain (CG) measure, accumulates the relevance scores of retrieved documents along the ranked list G , where the document IDs are replaced with their relevance scores. The cumulated gain at rank i , $CG[i]$, is computed as the sum of the relevance scores up to that rank:

$$CG[i] := \sum_{j=1}^i G[j] \quad (3)$$

For example, based on a four-point relevance scale with relevance degrees of $\{0, 1, 2, 3\}$, the ranking $G = \langle 3, 2, 3, 0, 1, 2 \rangle$ produces the cumulated gain vector of $CG = \langle 3, 5, 8, 8, 9, 11 \rangle$.

For each query, an ideal gain vector, I , can be derived by filling the rank positions with the relevance scores of all documents in the recall-base in decreasing order of their degree of relevance. A retrieval run’s CG vector can then be compared to this ideal ranking by plotting the gain value of both the actual and ideal CG functions against the rank position. We obtain two monotonically increasing curves (levelling after no more relevant documents can be found).

By dividing the CG vectors of the retrieval runs by their corresponding ideal CG vectors, we obtain the normalised CG (nCG) measure. Here, for any rank the normalised value of 1 represents ideal performance. The area between the normalised actual and ideal curves represents the quality of a retrieval approach.

2.2.2 The XCG metrics

XCG makes use of both the CG and nCG metrics. The extension of these metrics to XML documents, and in particular to INEX, lies partly in the way the relevance score for a given document - or in this case document component - is calculated via the definition of so-called relevance value (RV) functions, and partly in the definition of the ideal recall-bases.

An ideal recall-base is a set of ideal result nodes selected from the full recall-base based on a given quantisation function and the following methodology. Given any two components on a relevant path¹, the component with the higher quantised score (as per chosen quantisation function) is selected. In case two components' scores are equal, the one deeper in the tree is chosen². The procedure is applied recursively to all overlapping pairs of components along the 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 resulting ideal recall-base contains the best elements to return to a user based on the assumptions that overlap between result nodes should be avoided and that the user's preferences are reflected within the employed quantisation function. The derived ideal recall-bases then form the basis for the ideal gain vectors for each topic.

While I is derived from the ideal recall-base, the gain vectors, G , for the runs under evaluation are based on the full recall-base in order to enable the scoring of near-miss components. All relevant components of the full recall-base that are not included in the ideal recall-base are considered as near-misses.

In order to obtain a given component's relevance score (both for I or G) at a given rank position, XCG defines the following result-list dependent relevance value (RV) function:

$$rv(c_i) = f(quant(assess(c_i))) \quad (4)$$

where $assess(c_i)$ is a function that returns the assessment value pair for the component c_i , if given within the recall-base and $(0, 0)$ otherwise. The $rv(c_i)$ function then returns, for a not-yet-seen component c_i , the quantised assessment value pair $quant(assess(c_i))$, where $quant$ is a chosen quantisation functions, e.g. sog . In this case $f(x) = x$. For a component, which

¹A relevant path is defined as 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. E.g. in Figure 1 there are 6 relevant paths.

²We are also experimenting with the alternative option, i.e. selecting the the node higher in the tree.

has been previously fully seen by the user, we have $rv(c_i) = (1 - \alpha) \cdot quant(assess(c_i))$, i.e. $f(x) = (1 - \alpha) \cdot x$. With α set to 1, the RV function returns 0 for a fully seen, hence redundant, component, reflecting that it represents no value to the user any more. Finally, if c_i has been seen only in part before (i.e. some descendant nodes have already been retrieved earlier in the ranking), then $rv(c_i)$ is calculated as:

$$rv(c_i) = \alpha \cdot \frac{\sum_{j=1}^m (rv(c_j) \cdot |c_j|)}{|c_i|} + (1 - \alpha) \cdot quant(assess(c_i)) \quad (5)$$

where m is the number of c_i 's relevant child nodes.

In addition to the above, the final RV score is obtained by applying a normalisation function, which ensures that the total score for any group of descendant nodes of an ideal result element cannot exceed the score achievable if retrieving the ideal node itself. For example, in Figure 1 the two ideal result nodes for the quantisation function sog are `sec4` and `sec6`. Since these results represent the best nodes for the user, a system returning these should be ranked above others. However, if another system retrieved all the leaf nodes, it may achieve a better overall score if the total RV score for these nodes exceeds that of the ideal nodes. The following normalisation function safeguards against this by ensuring that for any $c_j \in S$:

$$\sum_{c \in S} rv(c) \leq rv(c_{ideal}) \quad (6)$$

where S is the set of retrieved descendant nodes of the ideal node and where c_{ideal} is the ideal node that is on the same relevant path as c_j .

3 EVALUATION SETUP

3.1 What to evaluate?

The evaluation of a metric requires a number of tests. Voorhees in [8] identifies two aspects to qualify an evaluation: fidelity and reliability. Fidelity reflects the extent to which an evaluation metric measures what it is intended to measure, while reliability is the extent to which the evaluation results can be trusted. In this paper we concentrate on the latter test. We take the viewpoint of [8] that in a comparative evaluation setting, reliability reflects "a metric's ability to rank a better system ahead of a worse system". This is, of course, highly dependent on a definition on what makes a system better or worse than another. The basis for such a

decision lies within the user satisfaction criterion defined within the given retrieval task.

This criterion in the INEX CO track is (largely) defined by the task definition. According to this, within the CO task, the aim of an XML retrieval system is to point users to the specific relevant portions of documents, where the user’s query contains no structural hints regarding what the most appropriate granularity of relevant XML elements should be. The evaluation of a system’s effectiveness should hence provide a measure with respect to the system’s ability in retrieving such components. But what exactly are these “most appropriate” components? At the moment, we don’t actually have an exact answer to this in INEX. Intuition dictates that users would prefer elements that contain as much relevant information and as little irrelevant information as possible. Therefore, given a set of possible retrievable components in an arbitrary document (such as an article in INEX), the best elements to return to the user should be those that are “most” exhaustive and “most” specific to the user’s request³. However, given two relevant components, one highly exhaustive but only fairly specific ($(e, s) = (3, 2)$) and another which is only fairly exhaustive but highly specific ($(e, s) = (2, 3)$), which one should be regarded as better? The answer to these kinds of questions in INEX is provided by the quantisation functions. As mentioned in the previous section, each quantisation function reflects a set of possible user preferences. According to these preferences, it is then possible to identify the “best” components as those elements that score highest.

Overall, systems should then rank these “best” components in decreasing order of their quantised scores, i.e. highest scoring elements should be ranked first. In addition, we reason that users do not want to be returned overlapping redundant elements, so systems should be either penalised or at least not rewarded for such redundancy.

Given a user satisfaction criterion, a simple method to evaluate a metric’s reliability is to construct appropriate test data for which we can derive expectations as to what the metric’s outcome should be and check if the expected output is indeed obtained. The expected output is in the form of rankings of systems that meets user expectations. A system ranking is simply an ordered list of runs sorted by decreasing value of effectiveness. For example, according to the user preference that non-overlapping results are preferred to over-

³Note that most exhaustive and specific here **does not!** equate to $(e, s) = (3, 3)$ nodes, but refers to the nodes with the highest available exhaustivity and specificity score. For example, it may be that amongst all the possible retrievable components in an article, the most exhaustive node is $e = 1$, or the most specific node is $s = 2$.

lapping ones, we would expect that from two systems producing respective result rankings, the former would be regarded as the better system by a reliable evaluation metric.

For our test data, we constructed a number of simulated runs, which are described next.

3.2 Simulated runs

Each simulated run is populated with components derived from the full recall-base⁴, where the selection and ordering of the components is according to an assumed set of user preferences defined by the *sog* quantisation function (Equation 2). We constructed the following simulated runs:

iBsoG: is a ranked result list that contains only ideal results selected according to the quantisation function *sog*, where the ordering of the components within the ranking is also according to *sog*. The selection of the ideal results here is done according to the procedure described in Section 2.2.2. As an example, consider the relevant nodes in Figure 1 as an imaginary full recall-base. From this, we would obtain the following result ranking for our iBsoG run: {sec6, sec4}.

frbBsoG: is a run that contains all relevant components of the full recall-base, where the components are ordered by the quantisation function *sog*. E.g. for Figure 1, all shown nodes will be included as follows: {sec6, sec6/p1, sec6/p2, sec6/ip12, sec4/p1, sec4/ip12, sec4, sec4/p2, bdy1, article1}.

iaBsoG: contains all ideal results and all their relevant ascendant nodes ordered by *sog*. E.g. from Figure 1, we obtain: {sec6, sec4, bdy1, article1}.

idBsoG: contains all ideal results and all their relevant descendant nodes ordered by *sog*. E.g. from Figure 1, we get: {sec6, sec6/p1, sec6/p2, sec6/ip12, sec4/p1, sec4/ip12, sec4, sec4/p2}.

loBsoG: contains all relevant leaf nodes ordered by *sog*. E.g. from Figure 1, we get: {sec6/p1, sec6/p2, sec6/ip12, sec4/p1, sec4/ip12, sec4/p2}. Note that a leaf node here refers to leaf nodes on the relevant paths within an article, which may be non-leaves within the article file itself (e.g. sec[6]/p[2]

⁴We used assessments04-v1.0.tar.gz.

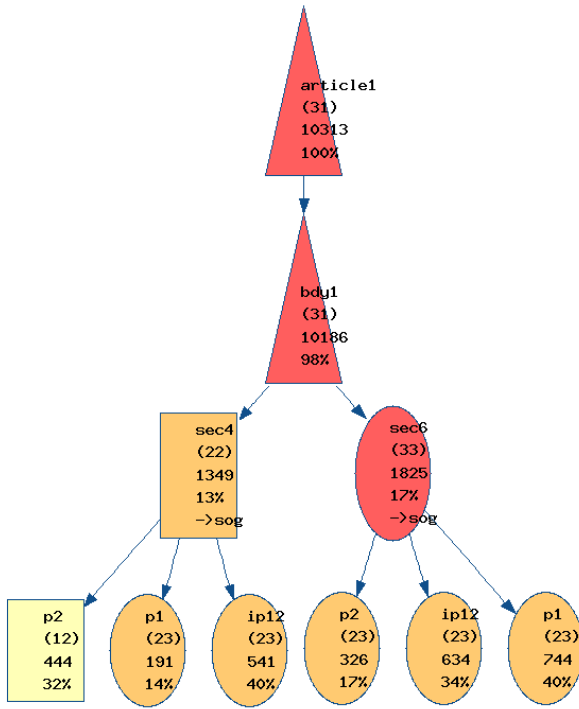


Figure 1: Sample assessments showing only relevant nodes (i.e. $e > 0$ and $s > 0$) for topic 163 in the article file `co/2001/r7022.xml`. For each node, the node name, the assessment value pair (es), the size in characters and the size ratio to its parent node is shown.

may have a number of irrelevant descendant nodes).

aoBsog: contains only relevant article nodes ordered by *sog*. E.g. from Figure 1, we get: `{article1}`.

3.3 Expected system rankings

Based on the user preferences captured by the *sog* quantisation function, systems that return the best components (i.e. highest quantised-scoring elements) should be ranked above others. Based on the intuition that users do not want to be inundated with multiple redundant, nested components, systems that return minimum amount of overlapping results would be preferred.

From these two assumptions, we can derive a relative ranking of simulated runs that should then be matched by a metric if it is to be proved reliable. From the latter assumption, we can reason that the runs

should be ranked as follows:

$$iBsog(0\%) \succeq loBsog(0\%) \succeq aoBsog(0\%) \succ idBsog(57.9\%) \succ iaBsog(70.7\%) \succ frbBsog(77.5\%)$$

where $a \succ b$ signals that ‘run a performs better than b ’.

Based on the quantisation function, a metric should rank those systems first that are able to return the best components. Since, the best components are defined by the quantised score of the quantisation function, without looking at the document collection and the actual relevance assessments we cannot safely predict much more than:

$$iBsog \succeq b$$

where $b \in \{loBsog, aoBsog, idBsog, iaBsog, frbBsog\}$. This is because the best scoring elements could be, e.g., the leaf or article nodes (depending on the judgements of the assessor). With respect to the runs *idBsog*, *iaBsog*, *frbBsog*, since *iBsog* is a subset of all these runs, the expectation is that they could produce possibly as good results as the ideal run, but not better.

The combination of these two criteria, if producing conflicting rankings, is currently an open question. It may be solved by defining the relative importance of the two aspects, which may be a parameter of a given user’s model for XML retrieval. Within the XCG metrics, given that the derived ideal recall-bases are completely overlap-free and that the overlap of result elements is considered directly within the way the relevance scores are calculated (RV function) for a run, it is not actually possible for the two criteria to produce conflicting rankings.

3.4 Metric reliability tests

We evaluated each of the simulated runs using the two metrics⁵. The resulting graphs are shown in Figures 2 and 3.

	inex-2002	nXCG
iBsog	0.1430 (5)	1.0000 (1)
frbBsog	0.7437 (1)	0.5369 (5)
iaBsog	0.2567 (4)	0.7936 (3)
idBsog	0.6195 (2)	0.7790 (4)
loBsog	0.3944 (3)	0.8269 (2)
aoBsog	0.0296 (6)	0.4096 (6)

Table 1: System Rankings

In order to obtain an overall ranking, we use the MAP measure for the *inex-2002* metric and the mean

⁵Throughout the paper, we used $\alpha = 1$ within the XCG metrics.

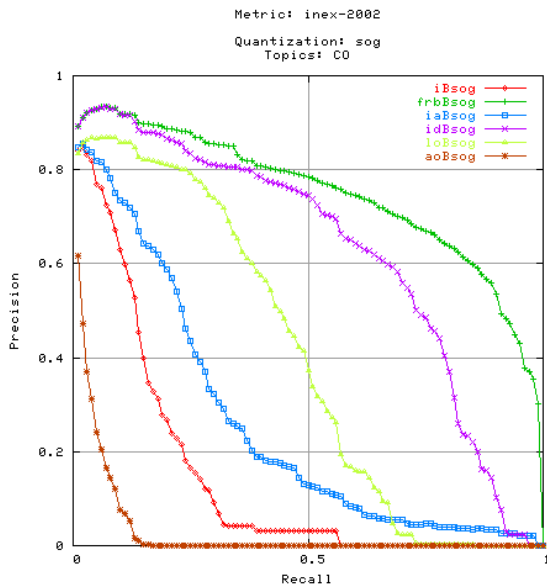


Figure 2: Results of the inex-2002 metric

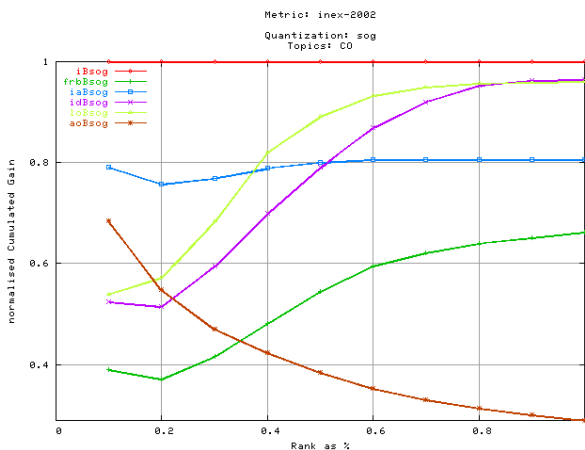


Figure 3: Results of the nXCG metric

average of the nCG values for nXCG (averaged over the 10 rank % points). Table 1 summarises the results. The numbers in brackets show the achieved system (run) ranks.

For the inex-2002 metric, both the graph and the MAP results clearly illustrate that better effectiveness is achieved by systems that return not only the most desired components, but also their ascendant (iaBsog) or descendant (idBsog) elements, hence inundating users with redundant components. In fact, according to this measure only the article-only run (aoBsog) has worse performance than the ideal run (iBsog). Best performance is achieved by the run that returns the full recall-base (frbBsog).

Looking at the results for the nXCG metric, we can see that best performance is achieved by the ideal run (iBsog), which is registered at a constant 1 normalised cumulated gain value. The worse performer is the article-only run (aoBsog), followed by the full recall-base run (frbBsog). The performance of the remaining runs (iaBsog, idBsog and loBsog) is evaluated as worse than the ideal, but better than the full recall-base run.

What is clear from the above is that the inex-2002 metric cannot reflect true performance differences when systems return overlapping elements as these can artificially raise the performance indicator. The nXCG metric's ranking of system's performances, on the other hand, corresponds to the user satisfaction criterion: the retrieval of ideal nodes representing the best nodes for the user (in accordance with a given set of user preferences expressed within a chosen quantisation function) is rewarded, while the retrieval of near-misses is also considered. Systems that retrieve such near-misses can achieve good performances, but cannot surpass an ideal system's score.

In both graphs, the comparison of the article-only (aoBsog), the leaf-only (loBsog) and the ideal (iBsog) runs gives an indication of the metrics' capabilities for ranking systems whose output contains no overlapping results. With the inex-2002 metric, the ideal run is scored lower than the leaf-only run, while with the nXCG metric, although the leaf-only run's performance is the second best, it never beats the ideal run's effectiveness. The article-only run achieves worst performance in both cases. This suggests that the inex-2002 metric is able to rank systems when no overlapping results are returned. However, the fact that the leaf-only run seems to perform better than the ideal run points to the need that a similar score-normalisation function to that described in Section 2.2.2 would be required.

3.5 Top ten INEX CO runs

In this section, we list the top ten of the INEX 2004 runs for both metrics as an indication of how the rankings are effected, see Tables 2 and 3. As it can be seen, amongst the top ten only the run by Carnegie Mellon University appears in both tables, although the University of Amsterdam and the University of Waterloo also appear in both, but with different runs (and one run by Queensland University of Technology is actually ranked 11th by nXCG).

It can be observed that runs with less overlap score better in nXCG, but overlap-free runs are not a sufficient condition for obtaining high values, but relevant nodes still need to be found. In fact, in total there are 18 submitted runs with 0% overlap (while several other runs also have minimal overlap, e.g. 1% or less), but their average rank is only 41 (out of 69). There are a couple of runs, which nicely reflect the effect of overlap on the nXCG scores: e.g. the University of Amsterdam submitted these runs: UAms-CO-T-FBack (81.85% overlap) and UAms-CO-T-FBack-NoOverl (0% overlap), which are scored as 0.2636 and 0.3521, respectively. Another example may be the runs submitted by the University of Tampere (see Table 3). On the other hand, the runs submitted by RMIT: Hybrid_CRE (82.12% overlap), Hybrid_CRE_specific (0% overlap) and Hybrid_CRE_general (0% overlap) achieve scores of 0.2791, 0.2576 and 0.2540, respectively. The drop in effectiveness score suggests that when reducing overlap, the higher scoring nodes are removed from the ranking, leaving lower scoring nodes in the ranking and (presumably) filling the rest of the ranks with irrelevant nodes (provided the three runs are produced from the same baseline, of course).

Note that the detailed analysis of the difference in these rankings will follow in a separate paper.

4 CONCLUSIONS

In this paper we investigated how closely the output of the two metrics, inx-2002 and nXCG, reflect the user satisfaction criteria defined within the INEX CO task. The results confirm the weaknesses of the inx-2002 metric reported in [4, 5], but show that with an appropriate quantisation function (e.g. when leaf nodes represent the best nodes) or with an arbitrary quantisation function when combined with a normalisation method, the inx-2002 metric is able to produce system rankings that match the evaluation criteria, provided no overlapping results are returned by the systems.

We also described and further developed the XCG

metrics, which produced promising results in our metric reliability test. A weakness of the XCG metrics, however, is that they produce effectiveness scores for a given rank (or rank %) and not for recall. To address this issue, Gabriella Kazai is currently working on a version of the metric that is able to give recall related performance indicators. She is also working on an extension of the generalised Precision and Recall measures introduced in [6]. These will be published in the near future.

In the future, we also hope to be able to derive better user models and hence arrive at more accurate user satisfaction criteria based on the outcome of the INEX 2004 interactive track.

References

- [1] W. Cooper. Expected search length: A single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. *American Documentation*, 19(1):30–41, 1968.
- [2] N. Gövert and G. Kazai. Overview of the INitiative for the Evaluation of XML Retrieval (INEX) 2002. In N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, editors, *Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*. Dagstuhl, Germany, December 8–11, 2002, ERCIM Workshop Proceedings, pages 1–17, Sophia Antipolis, France, March 2003. ERCIM. <http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf>.
- [3] K. Järvelin and J. Kekäläinen. Cumulated Gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (ACM TOIS)*, 20(4):422–446, 2002.
- [4] G. Kazai. Report of the inx 2003 metrics working group. In N. Fuhr, M. Lalmas, and S. Malik, editors, *Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, Dagstuhl, germany, December 2003, pages 184–190, April 2004.
- [5] G. Kazai, M. Lalmas, and A. 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, Sheffield, UK, 2004.*, pages 72–79. ACM, July 2004.
- [6] J. Kekäläinen and K. Järvelin. Using graded relevance assessments in IR evaluation. *Journal of*

rank	Run	MAP	overlap %
1	IBM Haifa Research Lab (CO-0.5-LAREFIENMENT)	0.1327	80.89
2	IBM Haifa Research Lab (CO-0.5)	0.1274	81.46
3	University of Amsterdam (UAms-CO-T-FBack)	0.1060	81.85
4	LTI, Carnegie Mellon University (Lemur_CO_KStem_Mix02_Shrink01)	0.0941	73.02
5	IBM Haifa Research Lab (CO-0.5-Clustering)	0.0923	81.10
6	LTI, Carnegie Mellon University (Lemur_CO_NoStem_Mix02_Shrink01)	0.0879	74.82
7	Queensland University of Technology (CO_PS_Stop50K_049_025)	0.0839	71.06
8	Queensland University of Technology (CO_PS_099_049)	0.0803	76.81
9	Queensland University of Technology (CO_PS_Stop50K_099_049)	0.0784	75.89
10	University of Waterloo (Waterloo-Baseline)	0.0781	76.32

Table 2: Top ten INEX 2004 runs according to inex-2002, quant: sog

rank	Run	MAnCG	overlap %
1	University of Tampere (UTampere_CO_average)	0.3725	0
2	University of Tampere (UTampere_CO_fuzzy)	0.3699	0
3	University of Amsterdam (UAms-CO-T-FBack-NoOverl)	0.3521	0
4	Oslo University College (4-par-co)	0.3418	0.07
5	University of Tampere (UTampere_CO_overlap)	0.3328	39.58
6	LIP6 (bn-m1-eqt-porder-eul-o.df.t-parameters-00700)	0.3247	74.31
7	University of California, Berkeley (Berkeley_CO_FUS_T_CMBZ_FDBK)	0.3182	50.22
8	University of Waterloo (Waterloo-Filtered)	0.3181	13.35
9	LIP6 (bn-m2-eqt-porder-o.df.t-parameters-00195)	0.3098	64.2
10	LTI, Carnegie Mellon University (Lemur_CO_KStem_Mix02_Shrink01)	0.2953	73.02

Table 3: Top ten INEX 2004 runs according to nXCG, quant: sog

the American Society for Information Science and Technology, 53(13):1120–1129, 2002.

- [7] V. Raghavan, P. Bollmann, and G. Jung. A critical investigation of recall and precision. *ACM Transactions on Information Systems*, 7(3):205–229, 1989.
- [8] E. M. Voorhees. Overview of the TREC 2003 question answering track. In *Text REtrieval Conference, Gaithersburg*, 2003.

MultiText Experiments for INEX 2004

Charles L. A. Clarke Philip L. Tilker
School of Computer Science, University of Waterloo, Canada
{claclark,pltilker}@plg.uwaterloo.ca

1. INTRODUCTION

This is the first year that the MultiText Group participated in INEX, submitting three runs for the content-only adhoc retrieval task. To generate these runs, we combined our existing experience and tools with the advice and ideas found in recent INEX papers [1, 4], engineering a solid system capable of performing the basic task in a reasonable fashion.

2. RETRIEVAL METHODS

All runs used a version of the Okapi BM25 measure, augmented and tuned to meet the requirements of an XML retrieval task. One run (**Waterloo-Baseline**) used only the basic method, a second run (**Waterloo-Expanded**) added pseudo-relevance feedback, and a third (**Waterloo-Filtered**) added filtering to reduce overlap.

2.1 Basic Method

The MultiText system supports a number of facilities for querying XML and other structured document types, including generalized support for Okapi BM25 queries of the form

`rank X by Y`

where X is a sub-query specifying a set of document elements to be ranked and Y is a vector of sub-queries specifying individual retrieval terms.

For our INEX 2004 runs, the sub-query X specified a list of the following “acceptable” leaf nodes:

`sec p article ss1 bdy bb ip1 ss2 vt abs
fig app bm li fm`

This list of acceptable leaf nodes was created manually from the collection and the 2003 relevance judgments. An acceptable leaf node is one that occurs frequently in the collection, has a reasonable average length, and has many positive relevance judgments associated with it. Since the list was created manually, no specific thresholds were set for these criteria.

In general, terms in Y may be complex, containing proximity and structural constraints. However, for INEX 2004, Y was derived from the topic title simply by eliminating stopwords and negative terms (those starting with “-”), splitting apart phrases, and stemming the remaining terms with the Porter stemmer. For example, the title from topic 166

Copyright is held by the author/owner.
INEX 2004, December 2004

`+"tree edit distance" + XML - image`

became the four-term query

`"$tree" "$edit" "$distance" "$xml"`

where the “\$” operator within a quoted string stems the term that follows it.

Our implementation of Okapi BM25 is based on the description of Robertson et al. [5] with parameter settings of $b = 0.80$, $k_1 = 10$, $k_2 = 0$ and $k_3 = \infty$. The values chosen for k_1 and b were the result of tuning over the INEX 2003 topics and judgments; the other parameter values are standard for our system. Specifically, given a term set Q , a document d is assigned the score

$$\sum_{t \in Q} w^{(1)} q_t \frac{(k_1 + 1)d_t}{K + d_t} \quad (1)$$

where

$$w^{(1)} = \log \left(\frac{D - D_t + 0.5}{D_t + 0.5} \right)$$

D = number of documents in the corpus

D_t = number of documents containing t

q_t = frequency that t occurs in the topic

d_t = frequency that t occurs in d

$K = k_1((1 - b) + b \cdot l_d / l_{avg})$

l_d = length of d

l_{avg} = average document length

For the purposes of computing D and D_t , a document was defined to be an **article**, and these term statistics were used for ranking all element types. After retrieval the results were filtered to eliminate very short elements (under 25 words) and elements with unusual path expressions, those with forms that did not appear in the INEX 2003 relevant set.

2.2 Pseudo-Relevance Feedback

Two runs (**Waterloo-Expanded** and **Waterloo-Filtered**) augmented this basic method with pseudo-relevance feedback. For both runs, we used the QAP passage-retrieval algorithm [2, 3] to generate the top 25 passages from the INEX collection and the top 40 passages from a large Web collection. The top terms were extracted from these passages, re-weighted and added to the original query. In most respects we followed the procedure for our TREC Robust

Track experiments described in the MultiText TREC 2003 paper [3], which may be consulted for further details.

2.3 Filtering for Element Overlap

In addition to pseudo-relevance feedback, one of our runs (**Waterloo-Filtered**) extended the basic method with filtering to reduce overlap. An element was eliminated from the final ranked list if it was entirely contained within a higher ranked element, or if it contained a higher ranked element covering at least 80% of its contents. During a code review after our INEX 2004 runs were submitted, we detected a possible error in the implementation of this filtering procedure. However, we have not had an opportunity to verify and correct this error, or to re-run the experiment.

3. RESULTS AND CONCLUSION

For INEX 2004, our primary goal was to successfully complete the basic adhoc task, and we believe that we have satisfied this goal. However, both pseudo-relevance feedback and overlap filtering had a negative impact on performance, and we surprised by this result. In future, we plan to investigate these issues, and next year we hope to extend our participation to include other INEX tasks.

4. REFERENCES

- [1] David Carmel, Yoelle S. Maarek, Matan Mandelbrod, Yosi Mass, and Aya Soffer. Searching XML documents via XML fragments. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 151–158. ACM Press, 2003.
- [2] Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. Exploiting redundancy in question answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 358–365, New Orleans, September 2001.
- [3] David L. Yeung, Charles L. A. Clarke, Gordon V. Cormack, Thomas R. Lynam, and Egidio L. Terra. Task-Specific Query Expansion (MultiText Experiments for TREC 2003). In *Proceedings of the Twelfth Text Retrieval Conference*, Gaithersburg, MD, 2003. National Institute of Standards and Technology.
- [4] Jaap Kamps, Maarten de Rijke, and Börkur Sigurbjörnsson. Length normalization in XML retrieval. In *Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 80–87. ACM Press, 2004.
- [5] S. E. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC-7: Automatic ad hoc, filtering, VLC and interactive track. In *Proceedings of the Seventh Text REtrieval Conference*, Gaithersburg, MD, 1998. National Institute of Standards and Technology.

Logic-Based XML Information Retrieval for Determining the Best Element to Retrieve

Maryam Karimzadegan

Department of Computer
Engineering,
Sharif University of Technology,
Azadi Street, Tehran, Iran.
karimzadegan@ce.sharif.edu

Jafar Habibi

Department of Computer
Engineering,
Sharif University of
Technology,
Azadi Street, Tehran, Iran.
habibi@sharif.edu

Farhad Oroumchian

University of Wollongong in Dubai
FarhadOroumchian@uowdubai.ac.ae

Abstract – This paper presents UOWD-Sharif team's approach for XML information retrieval. This approach is an extension of PLIR which is an experimental knowledge-based information retrieval system. This system like PLIR utilizes plausible inferences to first infer the relevance of sentences in XML documents and then propagates the relevance to the other textual units in the document tree. Two approaches have been used for propagation of confidence. The first approach labeled "propagate-DS" first propagates the confidence from sentences to upper elements and then combines these evidences by applying Dempster-Shafer theory of evidence to estimate the confidence in that element. The second approach "DS-propagate" first applies the Dempster-Shafer theory of evidence to combine the evidences and then propagates the combined confidence to the parent element. The second approach performs relatively better than the first approach.

Index Terms-- Dempster-Shafer theory of evidence, Knowledge-based Information Retrieval, Plausible Reasoning, XML information retrieval.

1. Introduction

The widespread use of Extensible Markup Language (XML) has brought up a number of challenges for information retrieval systems. These systems exploit the logical structure of documents instead of a whole document. In traditional information retrieval (IR), a document is considered as an atomic unit and is returned to a user as a query result. XML assumes a tree like structure for the documents for example sentences, paragraphs, sections etc. Therefore XML retrieval not only is concerned with finding relevant documents but with finding the most appropriate unit in the document that satisfies users' information need. A meaningful retrievable unit shouldn't be too small because in

this case it might not cover all the aspects of users need (coverage). It shouldn't be too large either because in this case there could be a lot of non-relevant information that are of no particular interest to users current information need (specificity). Therefore, XML retrieval is an approach for providing more focused information than traditionally offered by search engines when we know the structure of the documents.

We have used the INEX collection for evaluation of our XML retrieval system. The INEX document collection is made up of the full-texts, marked up in XML that consists of 12,107 articles of the IEEE Computer Society's publications from 12 magazines and 6 transactions, covering the period of 1995-2002. Its size is about 494 megabytes. The collection contains scientific articles of varying length. On average an article contains 1,532 XML nodes, where the average depth of a node is 6.9. Overall, the collection contains over eight millions XML elements of varying granularity (from table entries to paragraphs, sub-sections, sections and articles, each representing a potential answer to a user's query [12].

The INEX collection consists of two sets of queries: CO (content only) and CAS (Content and Structure). There are 40 Co and 40 CAS queries in INEX 2004. In CO topics, the retrieval system is expected to return a ranked list of the most relevant elements. In other words, the granularity of the response varies depending on the relevance of the element while in CAS queries; a retrieval system should return a ranked list of elements as specified in the topic. For more information about CO and CAS queries, one can refer to [13]. The focus of this paper is on CO topics.

This paper explores the possibility of using Human Plausible Reasoning [1] and theory of Dempster-Shefer [2] for combining evidences as a means of retrieving relevant units (elements) of

documents. Collins and Michalski [3] developed the theory of Human Plausible Reasoning for question-answering situations. An experimental information retrieval system called PLIR which utilizes HPR is described in [4]. In [5], [6] and [7] authors suggest some applications of the theory for adaptive filtering, intelligent tutoring and document clustering, respectively. All these implementations confirm the usefulness and flexibility of HPR for applications that need to reason about users' information need. In this study, the theory HPR has been extended to accommodate XML information retrieval. This method utilizes Rich Document Representation [6] using single words, phrases, logical terms and logical statements that are captured from document contents.

2. Basics of Human Plausible Reasoning

For approximately 15 years, Collins and his colleagues have been collecting and organizing a wide variety of human plausible inferences made from incomplete and inconsistent information [1]. These observations led to the development of a descriptive theory of human plausible inferences that categorizes plausible inferences in terms of a set of frequently recurring inference patterns and a set of transformations on those patterns. According to the theory, a specific inference combines an inference pattern with a transformation that relates the available knowledge to the questions based on some relationship (i.e. generalization, specialization, similarity or dissimilarity) between them. The primitives of the theory consist of basic expressions, operators and certainty parameters. In the formal notation of the theory, the statement "coffee grows in the Lianos" might be written:

GROWS-IN (Lianos) = Coffee, $\gamma = 0.1$

This statement has the *descriptor* GROWS-IN applied to the argument Lianos and the *referent* coffee. The certainty of the statement (γ) 0.1, since it declares a fact about the Lianos. The pair descriptor and argument is called a *term*. Expressions are terms associated with one or more referents. All descriptors, arguments and referents are nodes in (several) semantic hierarchies. Any node in the semantic network can be used as a descriptor, argument or referent when appropriate. Figure 1 demonstrates the basic elements of the core theory.

There are many parameters for handling uncertainty in the theory. There is no complete agreement on their computational definitions and different computer models have implemented them

in different ways. The definition of the most important ones according to [1] is:

1. γ The degree of certainty or belief that an expression is true. This is applied to any expressions.
2. ϕ Frequency of the referent in the domain of the descriptor (e.g. a large percentage of birds fly). Applies to any non-relational statements.
3. τ Degree of typicality of a subset within a set. This is applied to generalization and specification statements.
4. δ Dominance of a subset in a set (e.g. chickens are not a large percentage of birds but are a large percentage of barnyard fowl). That is applied to generalization and specification statements.
5. σ Degree of similarity of one set to another set. Sigma applies to similarity and dissimilarity statements.

This theory provides a variety of inferences and transforms that allow transformation of known knowledge (statements) into not known information (new statements). For more information on how to implement the theory, one can refer to [8].

<p>Arguments $a_1, a_2, f(a_1)$ e.g. Fido, collie, Fido's master</p> <p>Descriptors d_1, d_2 e.g. bread, color</p> <p>Terms $d_1(a_1), d_2(a_2), d_1(d_2(a_1))$ e.g. bread(Fido), color(collie), color(breed(Fido))</p> <p>Referents $r_1, r_2, r_3, \{r_1, \dots\}$ e.g. collie, brown and white, brown plus other colors</p> <p>Statements $d_1(a_1)=r_1: \gamma, \phi$ e.g. means-of-location(bird)={fly...} :certain, high frequency(I am certain almost all birds fly)</p> <p>Dependencies between terms $d_1(a_1) \leftrightarrow d_2(f(a_1)): \alpha, \beta, \gamma$ e.g. latitude(place) \leftrightarrow average-temperature(place): moderate, moderate, (I am certain that latitude contains average temperature with moderate reliability, and that average temperature constrains latitude with moderate reliability)</p> <p>Implication between statements $d_1(a_1)=r_1 \leftrightarrow d_2(f(a_1))=r_2: \alpha, \beta, \gamma$ e.g. grain(place)={rice...} \leftrightarrow rainfall(place)=heavy: high, low certain</p> <p>(I am certain that if a place produces rice, it implies the place has heavy rainfall with high reliability, but that if a place has heavy rainfall it only implies the produces rice with low reliability)</p>

Figure 1. Basic Elements of the Core Theory

3. Information Retrieval by Plausible Inferences

There are four elements in a logic based IR system. Those are the description of documents, the

representation of queries, a knowledge base containing domain knowledge and a set of inference rules. This study also acknowledges that retrieval is inference but relevance is not material implication [9]. A document is retrieved only if its partial description can be inferred from a query description. Thus the retrieval process is expanding a query description by applying a set of inference rules continuously on the description of the query and inferring other related concepts, logical terms and statements until locating a document or documents which are described partially by these concepts or logical terms or statements. In XML retrieval the smallest unit that is inferred is a sentence.

3-1 Document Representation

In this model, documents are represented in possible worlds by a partial set of single words, phrases, logical terms and logical statements, i.e., the representation of a document is not limited to the set of its representative phrases or logical terms and statements. Any concept that can be inferred from representation, by plausible reasoning using the given knowledge base, is also a representative of the document content. In its simplest form, a typical document such as Van Rijsbergen's 1986 article entitled "A non-classical logic for information retrieval" can be represented as follows:

1. REF (Information Retrieval) = {doc#1 }
2. REF (Non-classical Logic)= {doc#1 }
3. REF (Non-classical Logic (Information Retrieval))= {doc#1 }

The first statement indicates the concept Information Retrieval is a reference for doc#1. The second statement states that the concept Non classical Logic is a reference for doc#1. The third statement expresses that the term Non-classical Logic (Information retrieval) is a reference for doc#1.

3-2 Representing a Query as an Incomplete Statement

A query can be represented as an incomplete logical statement in which the descriptor is the keyword REF (reference) and its argument is the subject in which the user is interested. The referents of this statement i.e. the desired documents, are unknown. So, we should find the most suitable referent for this logical statement. A typical query in logical notation will have the form like this below:

$$\text{REF (A-Subject)}=\{?\}$$

Therefore the retrieval process can be viewed as the process of finding referents and completing this incomplete sentence.

A query with a single phrase, such as "Content Retrieval Technique ", can be formulated as:

$$\text{REF(Content-based Retrieval Technique)} = (?)$$

A query consisting of a sentence fragment can be treated as a regular text. Therefore it can be scanned for extracting its logical terms. For example, consider the topic number 197 from the INEX2004 [10] collection.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="197" query_type="CO"
ct_no="178">
<title>"data compression" +"information
retrieval"</title>
<description>We are interested in articles about usage
data
compression in information retrieval systems, because
IR systems are very memory consuming, and these
systems offer wide range of various data to be
compressed i.e. texts, index data, images, video
etc.</description>
<narrative>Our research team ARG (AmphorA
Research Group) develops experimental information
retrieval system called AmphorA. The AmphorA
includes many retrieval techniques such as
implementation of vector and boolean queries,
multidimensional indexing etc. Other research activity
is background of such system, which means data
compression and storage for indexing and querying
algorithms. We are especially interested in word-based
compression. Article is considered relevant if it
provides information about compression in IR system.
Compression means compression of text, index, query
evaluation on compressed index and text, image
retrieval e.g. retrieval in JPEG compressed images.
Watermarking, straightforward storage of compressed
images in database etc. is considered as non-relevant
article.</narrative>
<keywords>data, compression ,information, retrieval,
indexing, data structure</keywords>
</inex_topic>
```

Figure 2. CO Topic number 197 of INEX2004 collection

The query in Fig 2 contains the sentence fragment "data compression in information retrieval systems". This query can be converted into a logical term, which is revealed by the proposition *in*. The query can be represented as:

$$\text{REF(data compression (information retrieval system))}=\{?\}$$

Queries with more than one concept or term can be represented as a set of simple queries and the system can retrieve a set of references for each one separately and then reexamine the sets by

combining the confidence on references, which are members of more than one set. Then the sets can be joined and the resulting set can be sorted according to the confidence value.

3-3 Document Retrieval

The process of information retrieval in this system as mentioned above is about finding referents and completing an incomplete statement. The incomplete statement which is formed from the query has one of the following two formats:

- REF(c) = {?}
- REF(a(b)) = {?}

The above statements mean, we are interested in referents (references, documents) for the concept c or logical term a(b). The following steps describe the process of completing the above query statements.

STEP 1- SIMPLE RETRIEVAL

Find references that are indexed by the concepts or terms in the query.

- Scan the query and extract single words, phrases and logical terms.
- Find all the references in the collection for the followings:
 - o All the single words such as “Software” in the query.
 - o All the phrases such as “information retrieval”
 - o All the terms such as $a(b)$ that are in query such as (coding algorithm(text compression)).

In the experiments, syntactic phrases of length 2 or 3 have been used.

STEP2- SIMPLE BUT INDIRECT RETRIEVAL

Find references that are rewording of the logical term in query.

- find referents c for all the logical terms $a(b)$ where $a(b) = \{c\}$.
- find all the references to the referents.

For example *Fortran* is a referent for the logical term *Language (programming)* in the logical sentence: *Language (programming)=Fortran*.

The above statements means *Fortran* is a *programming language*. Therefore if query is about *programming languages*, system will return all the references for *Fortran*.

STEP3- USE RELATIONSHIPS AND INFERENCE

This step uses all the transforms and inference of the theory to convert the original concepts and/or logical statements into new statements and retrieve their references as the references of the query.

- find other referents such as f with SPEC, GEN and/or SIM relationship with referent c where $f \{SPEC \text{ or } GEN \text{ or } SIM\} c$ in order to conclude $a(b) = \{f\}$. Then find all references indexed by f in the collection.

- find all the logical terms such as $d(e)$ with mutual dependency relationship with term $a(b)$ where $a(b) <---> d(e)$. Find all references for $d(e)$.

- find all the logical statements such as $d(e)=\{b\}$ with mutual implication with statement $a(b)=\{c\}$ where $a(b)=\{c\} \leftrightarrow d(e)=\{b\}$. Find all references for new logical statements.

Step 3 is repeated as many times as necessary in order to find the best elements. Basically, the process is similar to rewriting query and looking for references for the new query.

Since a term, referent or sentence in a document could be reached through several different relationships or inferences, therefore a method for combining the confidence values attributed from these different evidences should be taken. For combining these confidence or uncertainty values, the Dempster-Shefer theory of evidence has been employed.

STEP 4- PROPAGATION

Through the application of steps 1 through 3 the best possible sentence candidates will be recovered. However, since the documents have structures therefore system needs to propagate the confidence in the sentences to the confidence in the other elements of this structure.

The inference depicted in figure 3 propagates the certainty value of a sentence to the paragraph that this sentence resides in. The first line represents our assumption that if a sentence is relevant to a concept then the paragraph that this sentence resides in is also relevant. The second line expresses the confidence on a specific sentence such as $s1$ to be relevant to some concept such as $c1$. The third sentence describes the importance of sentence $s1$ for the paragraph $p1$. The parameter $\delta1$ represents the dominance of the sentence among other sentences in the paragraph. We have assumed that the middle sentences in a paragraph are of less important than beginning and ending sentences. For that, we use a linear function that the slope of it is negative 1 from the first sentence till the middle sentence, then the slope of the function increases to positive 1 from the middle sentence to the last sentence in the paragraph. The parameter $\mu1$ describes how much

of concepts in the paragraph are covered by the sentence s1. This parameter is estimated by “The number of concepts in sentence s1 divided by the total number of concepts in p1”. The parameter A1 represents acceptability of the sentence s1 being relevant to a concept c1 by the user population. The optimum value of this parameter could be learned during experiments. The rest are true for all cases. The inference estimates the confidence γ on paragraph p1 to be relevant to concept c1. This confidence is influenced by the confidence on the relevance of the sentence s1 to the query, the dominance of sentence s1 in the paragraph p1 and the amount of paragraph p1 which is covered by the sentence s1. The propagation does not stop at paragraph level and with the help of inferences similar to the one described in figure 3, it will continue until the document itself receives confidence values from its children.

1- REF(C) = {sentence} AND Located (sentence) = { paragraph } \Leftarrow REF (c) = {paragraph} α_1, γ_1 2- REF(c1)={s1} γ_2, A_1 3- Located (p1)={s1} δ_1, μ_1 4- P1 SPEC Paragraph $\gamma_3 \rightarrow 1$ 5- S1 SPEC Sentence $\gamma_4 \rightarrow 1$ 6- C1 SPEC Concept $\gamma_5 \rightarrow 1$ ----- REF (C1) = {P1} $\gamma = \gamma_2 * SQRT(\delta_1, A_1, \mu_1) * SQRT(\alpha_1, \gamma_1$

Fig.3. Inference for Propagating the Certainty Value from a sentence to a paragraph.

Each element in the document structure may receive multiple confidence values. Sentences retrieved through different inferences will have a confidence value from derived from each inference. Other elements receive different confidence values through propagation of the confidence values of their children. For combining these different values, we used the Dempster-Shafer theory of evidence. In sentence level, every inference returns a certainty value for each sentence of the document inferred by each term of the query. These certainty values are modeled by a density function $m : 2^\Omega \rightarrow [0,1]$ called a basic probability assignment (bpa).

$$m(\phi) = 0, \sum_{A \subset \Omega} m(A) = 1 \quad (1)$$

$m(A)$ represents the belief exactly committed to A, that is the exact evidence that the sentence of the document is relevant to a query term. If there is positive evidence for relevance of a sentence of a document to a query term, then $m(A) > 0$, and A is called a focal element. The focal element and bpa define a body of evidence. In this problem, we assume that focal elements are singleton sets. Each body of evidence is composed of the confidence on relevance of a document to each query term as estimated by inferences of plausible reasoning. Then,

$$m(\phi) = 0, m(\{doc_j\}) + m(T) = 1 \quad (2)$$

$m(T)$ is referred to evidence that can not be assigned yet. The $m(T)$ represents the uncertainty associated to the entire set of sentences of documents being relevant to a query term. Given a bpa m, belief function is defined as the total belief provided by the body of evidence for relevance of a sentence of a document to a query term. Because the focal elements are singleton, then the belief function equates to the mass function. Dempster-Shafer rules for combination, aggregates two independent bodies of evidence defined within the same frame of discernment into one body of evidence. Since the focal elements are singleton, the combination function becomes simpler than Dempster's rules of combination. DS provides three functions for scoring of documents: mass, belief, and plausibility functions. For the first level, we compute the mass function to combine the evidences for one query term. In the second level, the evidences of each query part for different sentences of the documents should be combined to compute the final result. In this level, no preference are given to any of the query terms, therefore we have used the average function. These processes are repeated for each level (paragraph, section, ...) of the documents.

In the first phase of experiments, first sentences with the highest confidence in their relevance to user's information need have been inferred using plausible inferences. Then by using the inference depicted in figure 3 and Dempster-Shefer theory, the confidence in sentences is propagated to their paragraphs, sections and the entire document XML documents. Then the elements with highest confidence values are selected and put in order in a rank list to be shown to the user. This method of combining is called “propagate-DS” method.

We have used another method for combining the evidences named “DS-propagate”. It assumes that,

if more than one sentence relates to the user’s query, first we should combine the evidences using DS theory of evidence, and then propagate the confidences gained to the higher levels.

The difference between these two approaches relies on the fact that, in “propagate-DS” approach, first we propagate the confidences to higher levels, then we utilize the DS theory for combining the evidences, whereas in the “DS-propagate” approach, we first combine the evidences by using the DS theory, then propagate the combined confidence value to higher level.

4- Experiments

We have experimented with two approaches for combining the evidences. The results (for average of all RP measures) are depicted in fig. 4 and fig. 5, respectively. The results show that we entertain relatively higher precision for the DS-propagate method. It seems that by using the DS theory first, then propagating the result to the higher levels, our precision will become higher.

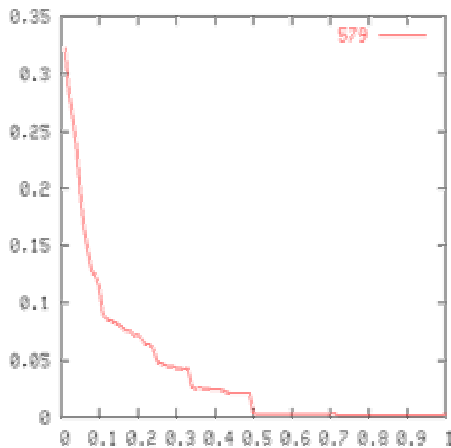


Fig 4. Average of all RP measures for DS-propagate approach

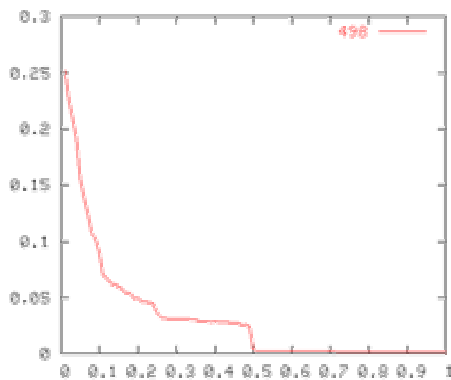


Fig 5. Average of all RP measures for propagate-DS approach

Our systems performs relatively better on “Exhaustivity” rather than on “Specificity” parameter. The figures for those motioned approaches are in fig. 6 and fig. 7, respectively.

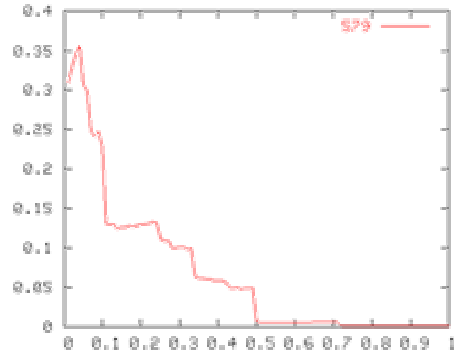


Fig. 6. RP (exhaustivity oriented with s=3,2) for DS-propagate approach

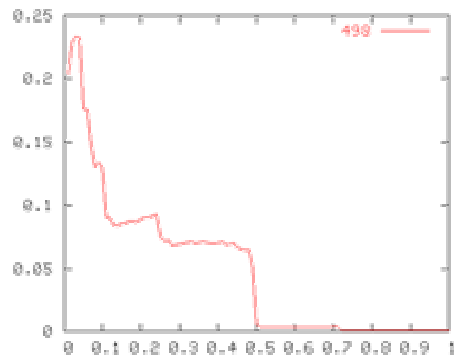


Fig. 7. RP (exhaustivity oriented with s=3,2) for propagate-DS approach

5. Conclusion

We presented a novel approach and implementation for finding, scoring and ranking of the meaningful units of retrieval in the context of XML information retrieval. A new specialized inference is added to the inferences of Collins and Michalski theory of Human Plausible Reasoning in order to handle XML information retrieval. Then by using the DS theory of evidence, we have combined different confidence values coming from different sources to estimate the final confidence in each element of an XML document element.

Currently we are analyzing the results of our experiments. Other experiments are underway with other methods of combining evidences such as fusion. It is possible also to develop other inferences for propagation of confidence from sentences to higher elements of the document the document tree.

6. References

- [1] A. Collins and M. H. Burstejn, "Modeling a theory of human plausible reasoning", *Artificial Intelligence III*, 1988.
- [2] G.A. Shafer, "Mathematical theory of evidence", *Princeton University Press*, 1976.
- [3] A. Collins and R. Michalski, "The logic of plausible reasoning A core theory", *Cognitive Science*, vol. 13, pp. 1-49, 1989.
- [4] F. Oroumchian, R.N. Oddy, "An application of plausible reasoning to information retrieval", *Proc. Of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich , pp. 244-252, August 1996.
- [5] F. Oroumchian, B. Arabi, E. Ashori, "Using plausible inferences and Dempster-shafer theory Of evidence for adaptive information filtering", *4th International Conference on Recent Advances in Soft Computing (RASC2002)*, Nottingham, United Kingdom, Dec 2002.
- [6] F. Oroumchian, B. Khandzad, "Simulating tutoring decisions by plausible inferences", *4th International Conference on Recent Advances in Soft Computing (RASC2002)*, Nottingham, United Kingdom, Dec 2002.
- [7] A. Jalali, F. Oroumchian, "Rich document representation for document clustering", *Coupling Approaches, Coupling Media and Coupling Languages for Information Retrieval Avignon (Vaucluse)*, France, vol. 1, pp. 802-808, April 2004.
- [8] K. Dontas, "An implementation of the Collins-Michalski theory of plausible reasoning", Master's Thesis, *University of Tennessee*, Knoxville, TN, August 1987.
- [9] C. J. Van Rijsbergen, "Toward an information logic", In N. J. Belkin & C. J. Van Rijsbergen (Eds.), *Proceedings Of The 12th Annual International SIGIR Conference*, Cambridge, MA, 1989.
- [10] <http://inex.is.informatik.uni-duisburg.de:2004/> accessed 7th of August 2004.
- [11] F. Oroumchian, "Information retrieval by plausible inferences: an application of the theory of plausible reasoning of Collins and Michalski", *Ph.D. Dissertation, School Of Computer And Information Science, Syracuse University*, 1995.
- [12] N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, "INEX: INitiative for the Evaluation of XML retrieval", In Ricardo Baeza-Yates, Norbert Fuhr, and Yoelle S. Maarek, editors, *Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval*, 2002.
- [13] B. Sigurbjörnsson, B. Larsen, M. Lamas, S. Malik, "INEX 2003 guidelines for topic development", *In Proceeding of INEX 2003 Conference*, Schloss Dagstuhl, Dec 2003.

Analyzing the Properties of XML Fragments decomposed from the INEX Document Collection

Kenji Hatano[†], Hiroko Kinutani[‡], Toshiyuki Amagasa[†]
Yasuhiro Mori[‡], Masatoshi Yoshikawa[‡], Shunsuke Uemura[†]

[†] Nara Institute of Science and Technology, Japan

{hatano, amagasa, uemura}@is.naist.jp

[‡] Ochanomizu University, Japan

kinutani@dblab.is.ocha.ac.jp

[‡] Nagoya University, Japan

mori@dl.itc.nagoya-u.ac.jp, yosikawa@itc.nagoya-u.ac.jp

ABSTRACT

In current keyword-based XML fragment retrieval systems, various granules of XML fragments are returned as retrieval results. The number of the XML fragments is huge, so that it causes adverse effects on index construction time and query processing time of the XML fragment retrieval systems if the XML fragment retrieval systems can not extract only answer XML fragments certainly. In this paper, we propose a method for determining XML fragments which are relevant in keyword-based XML fragment retrieval. We think it would help to improve overall performance of the XML fragment retrieval systems. The proposed method utilizes and analyzes statistical information of XML fragments based on a technique of quantitative linguistics. Moreover, our keyword-based XML fragment retrieval system runs on a relational database system. In this paper, we briefly explain implementation of our system.

Categories and Subject Descriptors

H.2.4 [System]: Relational databases; H.3.1 [Content Analysis and Indexing]: Linguistic processing; H.3.3 [Information Search and Retrieval]: Selection process; H.3.4 [Systems and Software]: Performance evaluation (efficiency and effectiveness)

General Terms

Relational database, XML Information retrieval, CO, Performance evaluation

Keywords

XML fragment retrieval system on relational database system, Analyzing properties of XML fragments

1. INTRODUCTION

Extensible Markup Language (XML) [5] is becoming widely used as a standard document format in many application domains. In the near future, we believe that a great variety of documents will be produced in XML. Therefore, in a similar way to developing Web search engines, XML information retrieval systems will become very important tools for users wishing to explore XML documents.

In the research area of XML retrieval, it is important to

propose a method for retrieving fragments of XML documents. XQuery [4], proposed by the World Wide Web Consortium (W3C), is known as a standard query language for XML fragment retrieval. Using XQuery, users can issue a flexible query consisting of both some keywords and XPath notations. If users already have knowledge of structure of XML documents, users can issue XQuery-style queries for XML fragment retrieval. Consequently, we afford to say that XQuery is suitable for searching data in XML documents¹.

At the same time, XML Query Working Group has been developing powerful full-text search functions [3, 2] to XQuery. This is because there are a lot of *document-centric* XML documents like articles in XML form, including structured information such as the names of authors, date of publication, sections, and sub-sections, as well as unstructured information such as the text contents of the articles. However, the document-centric XML documents like these have different XML schemas in each digital library, so that nobody can comprehend the structure of XML documents and can issue a formulated query like XQuery into XML fragment retrieval systems. Therefore, we believe that XML information retrieval systems should employ a much simpler form of query such as keyword search services without utilizing XQuery-style queries. Keyword search services enable users to retrieve needed information by providing a simple interface to information retrieval systems. In short, it is the most popular information retrieval method since users need to know neither a query language nor the structure of XML documents.

Because of the aforementioned background on XML fragment retrieval, much attention has recently been paid to a keyword-based XML fragment retrieval system. This type of XML fragment retrieval systems usually decomposes document-centric XML documents into XML fragments using their markup, and generates an index of decomposed fragments for searching. In spite of simple approach in XML fragment retrieval, this method enables to retrieve XML fragments related to keyword-based queries pretty well. However, XML documents are decomposed as far as possible using their

¹In this paper, we refer this type of XML documents to *data-centric* XML documents.

markup; thus index construction time and query processing time are too long compared with current document retrieval systems. This is because returning various granules and the huge number of XML fragments as retrieval results causes adverse effects on processing time unless XML fragment retrieval systems can extract only answer XML fragments certainly.

We believe that XML fragments required to keyword-based fragment retrieval are only part of decomposed fragments from document-centric XML documents. In short, we think there are a certain type of XML fragments which are never returned as retrieval results regardless of issued keyword-based queries. In particular, extremely small XML fragments are unlikely to become retrieval results of keyword-based query from the viewpoint of the information retrieval research area. Therefore, we will be able to perform XML fragment retrieval more efficiently than with current system if we can eliminate irrelevant fragments in XML fragment retrieval from index file. To cope with this problem, we have to determine XML fragments which are relevant in XML fragment retrieval extracted from document-centric XML documents.

In this paper, we propose a method for determining the relevant XML fragments to efficiently search XML fragments. Our method utilizes and analyzes statistical information of XML fragments decomposed from original documents based on a technique of quantitative linguistics. Our proposal holds the promise of not only reducing index construction time and query processing time of XML fragment retrieval systems, but also dealing with many types of document-centric XML documents, because statistical information does not depend on structures of XML documents. We also perform some experiments for verifying the effectiveness of our proposal.

2. RESEARCH ISSUES

Currently, we believe that there are two types of keyword-based XML fragment retrieval systems. In this paper, we refer to these two types of keyword-based XML fragment retrieval systems, *data-centric type* and *document-centric type* of XML fragment retrieval systems, for the sake of convenience. The former is based on structured or semi-structured database systems with keyword proximity search functions that are modeled as labeled graphs, where the edges correspond to the relationship between an element and a sub-element and to IDREF pointers [1, 10, 13]. Dealing with XML documents as XML graphs facilitates the development of keyword-based information retrieval systems, which are able to perform the retrieval processing efficiently. On the other hand, the latter has been developed in the research area of information retrieval [8, 9], and enables us to retrieve XML fragments without indicating element names of XML documents. The large difference of these two types of XML fragment retrieval systems derives from data propriety of their retrieval targets. In short, we consider that the former focuses mainly on XML documents which have data-centric view, whereas the latter deals with ones with document-centric view. In the meanwhile, almost all XML fragment retrieval systems currently assume the existence of DTD of XML documents in either research. It is true that DTD facilitates enhancing retrieval accuracy and query processing

time of their systems. However, there are some problems associated with searching heterogeneous XML fragments on the Web. Thus, other types of XML retrieval systems not utilizing DTD are required. Consequently, XML fragment retrieval systems in the future will have to deal with heterogeneous XML documents whose structures are not uniform.

To meet the needs of the new architecture of XML fragment retrieval systems, we have been developing a keyword-based XML fragment retrieval system [12]. Our system focuses on retrieval of document-centric XML documents rather than that of data-centric ones, and does not utilize any information about elements of XML documents, whereas almost all existing XML fragment retrieval systems take advantage of the information for querying and indexing of XML documents. In our approach, XML documents must be decomposed into their fragments and decomposed fragments must be utilized to generate an index file. XML is a markup language, so that XML documents can be automatically decomposed into their fragments using their markup [16]. However, a problem surfaces because this gives rise to an unmanageable profusion of XML fragments. In other words, it takes very long time to construct an index file and to search XML fragments related to a keyword-based query. For this reason, inspecting not all decomposed XML fragments, but the XML fragments which are relevant in XML fragment retrieval would be better for reducing index construction time and query processing time. In the next section, we explain the method for determining the relevant fragments in XML fragment retrieval based on a technique of quantitative linguistics.

3. ANALYSIS OF INEX TEST COLLECTION

Our research group has been analyzing statistical information of the INEX document collection since the last year. According to our analysis, it was notable that variances of the statistical information, especially variance of length of XML fragments, were too large. Therefore, we have focused on the length of XML fragments since we found this fact. In our INEX 2003 paper [11], we regarded small XML fragments as irrelevant ones in XML fragment retrieval, and verified the reasonability of our proposal using recall-precision curves. However, we just sketched the outline of our proposal and did not show good ground for adopting our proposal. In this section, consequently, we show the practical justification of our proposal.

3.1 Properties of XML Fragments

3.1.1 Quantitative linguistics

In our INEX 2003 paper, we determined threshold of the length of XML fragments, and regarded the XML fragments whose length could not meet the threshold as irrelevant ones in XML fragment retrieval. However, this approach required a lot of experiments to decide the threshold, so that it was inappropriate to adopt this approach for developing a large-scale XML fragment retrieval system. Therefore, we think that we have to decide the threshold systematically.

It is well known that statistical information of XML fragments, such as a number of tokens, length of XML fragments, and so on, is useful to decide the thresholds. In the research area of quantitative linguistics, the statistical

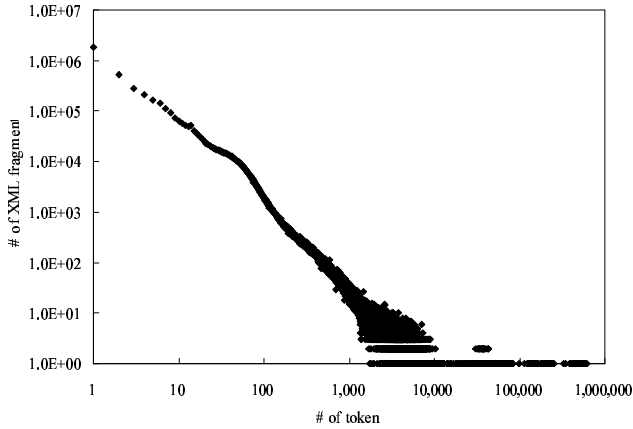


Figure 1: Relationship between n and $N(n)$.

information is often used. This is because analyzing the statistical information helps us to discover some rules in a document set, and the discovery of rules is essential for constructing a sound basis of a theory of terminology. In this research area, it is thought that examination to discover rules is similar to find out the systematic processes underlying a document set. For this reason, we employ a technique of quantitative linguistics to determine threshold using the number of tokens and XML fragments as statistical information, following the book [14]. Needless to say, not only the number of tokens and XML fragments, but also other mathematical or algebraic information can be utilized as statistical information. The reason for using such statistical information is that they can be extracted easily when our XML fragment retrieval system simultaneously analyzes XML documents and decomposes them into fragments.

Adopting a technique of quantitative linguistics, we can decide the threshold systematically.

3.1.2 Deciding relevant XML fragments

In the research area of quantitative linguistics, capturing properties of a document set is performed by analyzing statistical information. Utilizing the number of tokens and documents as statistical information, we can find a number of relationships. However, a small minority of documents have no relationship between the statistical information, so that it is said that the documents without relationship have anomalous property. Therefore, it is believed that such documents are not suitable for capturing properties of the document set, and should be disregarded in capturing properties.

We think that this concept can be utilized for determining irrelevant fragments in XML fragment retrieval. In short, if we are able to define a function between statistical information, XML fragments which do not follow the function can be regarded as irrelevant XML fragments. It is difficult to explain the process of determining irrelevant XML fragments on a conceptual basis, so that we describe the process using the following example.

Figure 1 shows log-log plots of the relationship between the number of tokens and XML fragments of the INEX docu-

Table 1: Comparison of APD with MPD in INEX 2003 relevance assessment.

	# of fragments	index construction (s)
APD	8,224,053	513,878
MPD	1,011,202	109,115
	query processing (s/topic)	average precision
APD	17.66	0.0707
MPD	5.27	0.1038

ment collection, where n is the number of tokens in each XML fragment and $N(n)$ is the number of XML fragments that contain n tokens. This figure shows that property of the INEX document collection is similar to that of Web document collection, because the log-log plots follow Zipf's distribution (or power-law distribution) [17]. Therefore, it is no wonder that statistics information of the INEX document collection follows the Zipf's distribution. However, it is difficult to determine whether XML fragments, in general, follow the Zipf's distribution or not.

From statistical point of view in quantitative linguistics, it is said that gaps between plots on Figure 1 cause a harmful effect on statistical information. Therefore, statistical information in plots with gaps is not used for capturing the property of the document set. In short, we afford to that XML fragments in these plots with gaps are irrelevant in XML fragment retrieval because we cannot capture the property of the document set accurately. As a result, we defined the relevant fragments in XML fragment retrieval as XML fragments in the plots in Figure 1 whose number of token is no fewer than 10, nor more than 10,000 in the case of adopting the INEX document collection. We think that this definition is sensible, because small XML fragments are not informative enough and large ones are too informative for users in keyword-based queries, so that small/large XML fragments are unlikely to be answers to the CO-topics.

3.1.3 Verification of XML fragments' properties

In order to verify the validity of the use of a technique of quantitative linguistics, we performed some experiments using the INEX 2003 relevance assessment. In these experiments, we measured average precisions, index construction time, query processing time, and the number of indexed XML fragments of the following two types of index files of our XML fragment retrieval system: the index files of all XML fragments (APD) and XML fragments except the irrelevant fragments described in Section 3.1.2 (MPD).

Figure 2 shows the recall-precision curves based on the INEX 2003 relevance assessment. We initially expected that recall-precision curves of APD and MPD were a very close each other, because the fragments which were judged as irrelevant in XML fragment retrieval did not rank in the top 1,500 of all. However, the recall-precision curve of MPD was higher than originally expected. Therefore, we think that the method proposed in Section 3.1.2 does not deteriorate the retrieval accuracy of XML fragment retrieval systems. In addition, the number of indexed XML fragments was significantly reduced by adopting our method, so that index construction time and query processing time were also reduced (see Table 1).

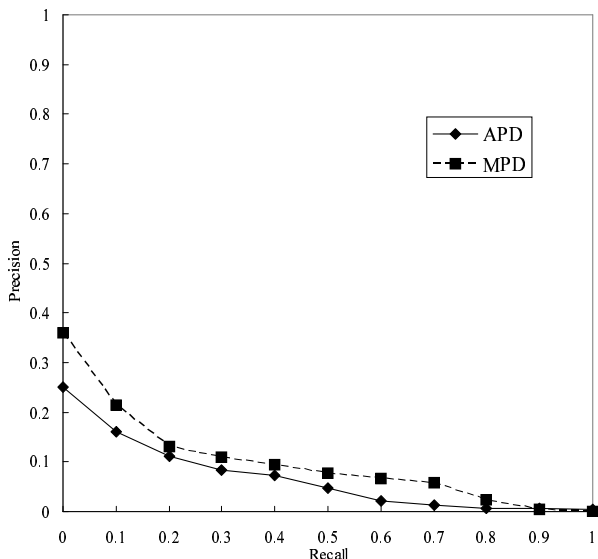


Figure 2: Recall-precision curves based on the INEX 2003 relevance assessment.

As afore-mentioned points, proposed method is useful not only to reduce index construction time and query processing time, but also to improve retrieval accuracy of XML fragment retrieval system. We think, therefore, proposed method can be also acceptable in INEX 2004 relevance assessment.

3.2 Experiments using INEX 2004 Relevance Assessment

Proposed method in this year worked well for reduction of index construction time and query processing time; thus we apply it to the INEX 2004 relevance assessment.

Figure 3 shows the recall-precision curves based on the INEX 2004 relevance assessment. Unlike the case of using the INEX 2003 relevance assessment, recall-precision curves of APD and MPD were a very close each other. The XML fragments which were judged as irrelevant in XML fragment retrieval based on proposed method did not rank in the top 1,500 of all; thus, the recall-precision curves were almost the same. Moreover, average precisions in the INEX 2004 relevance assessment were smaller than those in the INEX 2003. Our XML fragment retrieval system tends to retrieve relatively small XML fragments, so that it could not retrieve large XML fragments whose root node is `article`, `bdy`, or `fm`. As a result, exhaustively of our system tends to be small, while specificity of our system tends to be large, and average precision becomes small. We think this characteristic of our XML fragment retrieval system causes negative effects on average precision; therefore we have to propose new term weighting scheme for XML fragment retrieval. Currently, some term weighting schemes including ours have already published; however, they are not suitable for XML fragments which overlap each other. Consequently, we have to adopt another weighting scheme suitable for XML fragment

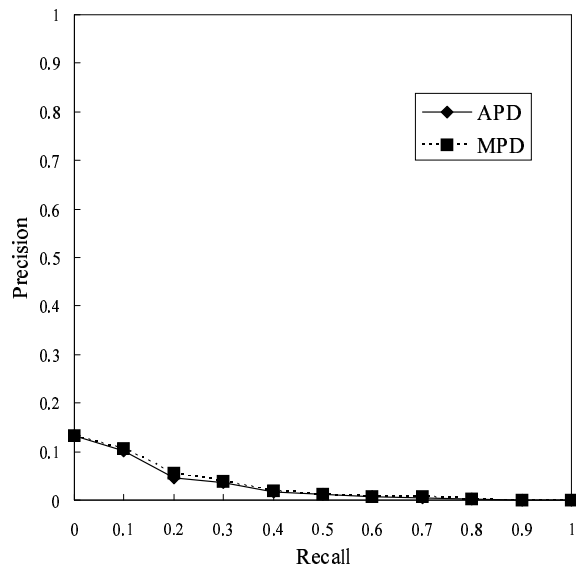


Figure 3: Recall-precision curves based on the INEX 2004 relevance assessment.

retrieval².

In order to ascertain the problems of proposed method, we also calculated the number of indexed XML fragments using proposed method of that of answer XML fragments in the relevance assessment. Table 2 shows the success ratio, which is the ratio of the number of answer XML fragments which were indexed by proposed method to that of all answer XML fragments. In this table, we show the topic IDs whose success ratios were in the bottom five of topics. Almost all success ratios of CO-topics of the INEX 2003/2004 relevance assessments were more than 90%; however, only the success ratios of these topics listed in Table 2 were remarkable small. In topic 185, especially, the higher the exhaustivity and the specificity, the smaller the success ratio. Moreover, the number of fragments, which were irrelevant in our method and were answers in the relevance assessment, was extremely large. We think that keyword-based queries (CO-topics) are usually issued for not specifying the hottest topics in contents, but just searching contents related to keywords. If this concept is true, the CO-topics whose answers contain small XML fragments are not suitable for the relevance assessment in XML fragment retrieval. Consequently, we think that not only controversial points of term weighting scheme for XML fragment retrieval, but also impertinent CO-topics for the INEX relevance assessment cause low average precision of our system.

On the other hand, the number of indexed XML fragments was significantly reduced by adopting our method in the same line with the case of INEX 2003, so that query pro-

²We think that length normalization of XML fragments [15] is one of the term weighting schemes in XML fragment retrieval. We believe that not only normalization of length of XML fragments, but also frequencies of tokens in XML fragments are important for improving average precision of our system.

Table 2: Success ratios of our method (worst 5).

topic ID	$(E, S) = (2, 3)$			$(E, S) = (3, 2)$			$(E, S) = (3, 3)$			total
	miss	all	success ratio	miss	all	success ratio	miss	all	success ratio	
187	378	554	0.32	1,028	1,463	0.30	646	848	0.25	0.50
166	9	15	0.40	5	48	0.90	27	62	0.56	0.76
192	5	42	0.88	0	0	N/A	1	10	0.90	0.81
194	3	4	0.25	0	5	1.00	4	11	0.64	0.83
179	0	6	1.00	3	5	0.40	0	0	N/A	0.88

Table 3: Comparison of APD with MPD in INEX 2004 relevance assessment.

	query processing (s/topic)	average precision
APD	25.48	0.0263
MPD	13.03	0.0286

cessing time was reduced as shown in Table 3. Therefore, XML fragment retrieval systems could perform index construction and query processing more efficiently than current systems if we adopt our method.

3.3 Discussions

Through the experiments based on INEX 2003 and 2004 relevance assessments, we found that index construction time and query processing time were reduced by adopting our method based on quantitative linguistics. Moreover, average precision of XML fragment retrieval system adopting our method did not become worse. As a result, proposed method helps to improve performance of the XML fragment retrieval systems. We are now working for improving average precision of our XML fragment retrieval system. We foresee that a novel term weighting scheme for XML fragment retrieval and a phrase match function enable to improve average precision of our XML fragment retrieval system.

4. IMPLEMENTING XML FRAGMENT RETRIEVAL SYSTEM ON RELATIONAL XML DATABASE

It goes without saying that not only accuracy, but also performance is the essential aspect of an XML fragment retrieval system. In fact, this is not an easy task, because we have to deal with several millions of fragments extracted from a document collection. In our project, we have been attempting to develop an XML fragment retrieval system based on relational databases. The reason for using relational databases is that we can utilize a variety of techniques, such as query optimization, storage management, and top-k ranking, for speeding up the process of XML fragment retrieval.

This section describes our first attempt for constructing such an XML fragment retrieval system. The system is based on a path-based relational XML database system, XRel [18], that is for storing and retrieving XML documents using off-the-shelf relational databases. In fact, we make an extension to XRel, that originally supports XPath as its basic query language, for supporting IR queries including CO- and CAS-topics.

4.1 An overview of XRel

4.1.1 The basics

XRel [18] is a scheme to realize XML databases on top of off-the-shelf relational databases. Using XRel, we can store any well-formed (or valid) XML documents in a relational database, and can retrieve XML fragments from the database using XPath expressions.

For storing XML documents, we shred the documents into small fragments so that they can be stored in relational tuples. Actually, we take the path-based approach, in that each node in an XML tree, such as element node, attribute node, and text node, is extracted and stored in a relational table with its *simple path expression* from the root and the *region* in the document. Here, a region is represented as a pair of integers (*start, end*), where *start* and *end* represent the starting and ending byte positions of the node in the XML file, respectively. This information is necessary and sufficient to retain the topology of XML tree, and we can therefore achieve lossless decomposition of XML documents into flat relational tables. An important notice here is that, for given regions, we can detect relationships among XML nodes, such as ancestor, descendant, precedes, and follows, by applying subsumption theorem³[18, 6]. Optionally, *depth*, that represents the depth of a node from the root, may be added as the third dimension in a region. In that case, we can additionally detect parent and child relations.

4.1.2 The schema design

The components extracted from an XML document are stored in relational tables. Actually, there are countless ways to design the relational schema. In XRel, we decided to use four kinds of tables according to the node types, namely, *Element*, *Attribute*, *Text*, and *Path*. In addition, metadata about XML files, such as location, size, and identifier, are stored in *Document* table. The actual schema definition of the tables are as follows:

```
Document (docID, filepath, length)
Element (docID, elementID, parentID, depth, pathID, \
        start, end, index, reindex)
Attribute (docID, elementID, pathID, start, end, value)
Text (docID, elementID, pathID, start, end, value)
Path (pathID, pathexp)
```

In this definition, metadata are stored in the *Document* table with unique IDs. Also, all possible path expressions are stored in the *Path* table as character strings with their unique IDs. The other tables refer to these values in terms

³Node *x* is an ancestor (descendant) of node *y* iff the region of *x* subsumes (is subsumed by, respectively) the region of *y*.

```

<vol no="1">
  <article>
    <title>TITLE1</title>
    <body>The first content.</body>
  </article>
  <article>
    <title>TITLE2</title>
    <body>The second <em>content.</em></body>
  </article>
</vol>

```

Figure 4: An example XML document.

of `docID` and `pathID` attributes. For the *Element* table, each element node is stored with its document ID (`docID`), path expression (`pathID`), and region (`start`, `end`, and `depth`). Additionally, `elementID`, that is the unique identifier of an element node, is included for efficiency reasons, although this information is not mandatory. Likewise, `parentID`, that refers to the `elementID` of its parent node, is defined so that parent nodes can easily be reached. The `index` (`reindex`) attribute represents (reverse) ordinal of nodes that share the same parent and the same path expression, and is used to speed up positional predicates, such as `/book/author[2]` (`/book/author[-2]`). For the *Attribute* and *Text* tables, all attributes, except for `value`, act like as in the *Element* table. The `value` attribute is to store textual values of attribute and text nodes.

Figure 5 demonstrates how an XML document in Figure 4 is decomposed and stored in the relational tables.

4.1.3 Query processing in XRel

For query retrieval, XRel supports XPath core, that is a subset of XPath [7], as its query language. Simply speaking, XPath core permits using “/” and “//” as location steps, and using typical predicate expressions. Given an XPath core expression, XRel translates it into an equivalent SQL query that operates on the relational tables. The point here is that the translated query can be processed solely by the underlying relational database system. Then, the query result is obtained in the form of result table, that is, in turn, reconstructed as result XML fragments.

For example, an XPath core query, “`//article/title[2]`,” can be expressed as:

```

SELECT  e1.docID, e1.st, e1.ed
FROM    Path p1, Element e1
WHERE   p1.pathexp LIKE '#%/article#/title'
AND     e1.pathID = p1.pathID
AND     e1.idx = 2
ORDER BY e1.docID, e1.st

```

We do not go into the details from the limitations of space, but more complicated queries containing node tests and/or predicates can be expressed in this way [18].

4.2 Supporting IR queries in XRel

4.2.1 Statistics

Although the above tables are sufficient to process XPath core queries, when considering INEX tasks, we need more information regarding IR statistics, in order to support IR queries like CO- and CAS-topics. To this end, we attempt

to maintain statistics of XML nodes, in addition to the basic tables of XRel. Those values include TF-IDF scores (including several variations), numbers of descendant elements, and various kinds of statistics. The concrete definition of the relational tables are as follows:

```

Token (docID, elementID, nodeFlag, token, articleNo, \
      tf, tfidf, tfidfMG, tfief, tfipf, tfOrder)
DescendantElementNum (docID, elementID, elementName, count)
ElementStatistics (docID, elementID, sentenceNum, termFreq, \
                  tokenFreq, wordFreq)

```

Let us take a closer look at the definitions. The *Token* table is for storing every occurrence of a distinct token. A token is stored with the document ID, element ID, and article ID where it appears, term frequency (`tf`), and several variations of term scores (`tfidf`, `tfidfMG`, `tfief`, and `tfipf`). `tfOrder` is used for ordering the tuples in the descending order of `tf`, so as to speed up table scans. The *DescendantElementNum* table maintains the number of descendant elements for each element. The *ElementStatistics* table is for storing various kinds of statistics regarding elements, such as numbers of elements, term frequencies, token frequencies, and word frequencies.

4.2.2 Processing CO-topics

Using the above tables as well as the basic XRel tables, we can express any CO-topics, in the form of `[key1, ..., keyl, +plus1, ..., +plusm, -minus1, ..., -minusn]`, as a SQL query:

```

SELECT docID, elementID, SUM(t.tfidf) result
FROM   token t
WHERE  t.token IN ('key_1', ..., 'key_l')
GROUP BY docID, elementID
HAVING (SELECT COUNT(*)
        FROM token
        WHERE token IN ('minus_1', ..., 'minus_n')
        AND   t.docID = token.docID
        AND   t.elementID = token.elementID ) = 0
AND
       (SELECT COUNT(*)
        FROM token
        WHERE token IN ('plus_1', ..., 'plus_m')
        AND   t.docID = token.docID
        AND   t.elementID = token.elementID ) = m
ORDER BY result DESC;

```

As can be seen, the calculation of TF-IDF is implemented in terms of an aggregation function. It should also be noticed that, in the translated query, “+key” and “-key” are expressed in terms of HAVING clause. The resulting query is sorted in descending order of TF-IDF scores, by the ORDER BY clause.

In the same way, we can express CO-topics with phrase match by using the `value` attribute in the *Text* table. However, this may not be realistic from the viewpoint of efficiency, due to the fact that the cost for approximate matching in SQL is quite expensive. Consequently, a naive implementation would cause serious performance degradation. Actually, we may need an additional index, that supports full-text search on text contents, to deal with phrase matching.

(a) Document			(c) Attribute					
docID	filepath	length	docID	elemID	pathID	st	ed	value
0	"/path/to/foo.xml"	203	0	0	1	1	1	"1"

(b) Element									
docID	elemID	parID	depth	pathID	st	ed	idx	reidx	
0	0	-1	1	0	0	202	1	1	
0	1	0	2	2	15	98	1	2	
0	2	1	3	3	29	49	1	1	
0	3	1	3	4	55	85	1	1	
0	4	0	2	2	102	195	2	1	
0	5	4	3	3	116	136	1	1	
0	6	4	3	4	142	182	1	1	
0	7	6	4	5	159	175	1	1	

(d) Text						(e) Path	
docID	elemID	pathID	st	ed	value	pathID	pathexp
0	2	3	36	41	"TITLE1"	0	"#/vol"
0	3	4	61	78	"The first content."	1	"#/vol#/@no"
0	5	3	123	128	"TITLE2"	2	"#/vol#/article"
0	6	4	148	158	"The second "	3	"#/vol#/article#/title"
0	7	5	163	170	"content."	4	"#/vol#/article#/body"
						5	"#/vol#/article#/body#/em"

Figure 5: A storage example of XRel.

4.3 Discussions

As discussed above, our system currently just supports XPath core and CO queries, and we therefore need further development for the purposes of extending its ability and improving the system performance. We are now working for improving the entire system performance. In the scheme, we use a novel technique to reduce the number of result candidates. Also, we are working for the support of CAS- (VCAS-) topics. Efficient execution of top-k ranking in CO- and CAS-topics is another important issue.

5. CONCLUSION

In this paper, we propose a method for determining XML fragments which are relevant in keyword-based XML fragment retrieval based on quantitative linguistics. Through some experimental evaluations, we find that proposed method helps to improve performance of our XML fragment retrieval system. Moreover, we describe our XML fragment retrieval system on relational database system which enables to query processing time of our system. If we can implement a phrase match function on our system, we can expect to improve average precision. Currently, we have a problem related to term weighting scheme suitable for XML fragment retrieval and query optimization with ranking function on relational database system. These problems are the immediate tasks of our project, so that we are going to solve these tasks in the near future. Originally, we are focusing on XML fragment retrieval without scheme information; thus we are going to address these problems with a view to the heterogeneous collection track of INEX project.

6. ACKNOWLEDGMENTS

This work is partly supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan, under grants #15200010, #16016243 and #16700103.

7. REFERENCES

- [1] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A System for Keyword-Based Search over Relational Databases. In *Proc. of the 18th International Conference on Data Engineering*, pages 5–16. IEEE CS Press, Feb./Mar. 2002.
- [2] S. Amer-Yahia, C. Botev, S. Buxton, P. Case, J. Doerre, D. McBeath, M. Rys, and J. Shanmugasundaram. XQuery 1.0 and XPath 2.0 Full-Text. <http://www.w3.org/TR/xmlquery-full-text/>, July 2004. W3C Working Draft 09 July 2004.
- [3] S. Amer-Yahia and P. Case. XQuery 1.0 and XPath 2.0 Full-Text Use Cases. <http://www.w3.org/TR/xmlquery-full-text-use-cases/>, July 2004. W3C Working Draft 09 July 2004.
- [4] S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language. <http://www.w3.org/TR/xquery>, Oct. 2004. W3C Working Draft 29 October 2004.
- [5] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible Markup Language (XML) 1.0 (Third Edition). <http://www.w3.org/TR/REC-xml>, Feb. 2004. W3C Recommendation 04 February 2004.
- [6] S.-Y. Chien, V. J. Tsotras, C. Zaniolo, and D. Zhang. Storing and querying multiversion XML documents using durable node numbers. In *Proc. of the 2nd International Conference on Web Information Systems Engineering*, pages 270–279, 2001.
- [7] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. <http://www.w3.org/TR/xpath>, Nov. 1999. W3C Recommendation 16 November 1999.

- [8] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. In *Proc. of 29th International Conference on Very Large Data Bases*, pages 45–56. Morgan Kaufmann, Sep. 2003.
- [9] N. Gövert, N. Fuhr, M. Abolhassani, and K. Großjohann. Content-Oriented XML Retrieval with HyREX. In *Proc. of the First Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 26–32. ERCIM, Mar. 2003.
- [10] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 16–27. ACM Press, June 2003.
- [11] K. Hatano, H. Kinutani, M. Watanabe, Y. Mori, M. Yoshikawa, and S. Uemura. Keyword-based XML Portion Retrieval: Experimental Evaluation based on INEX 2003 Relevance Assessments. In *Proc. of the Second Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 81–88, Mar. 2004.
- [12] K. Hatano, H. Kinutani, M. Yoshikawa, and S. Uemura. Information Retrieval System for XML Documents. In *Proc. of the 13th International Conference on Database and Expert Systems Applications*, volume 2453 of *LNCS*, pages 758–767. Springer, Sep. 2002.
- [13] V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword Proximity Search on XML Graphs. In *Proc. of the 19th International Conference on Data Engineering*, pages 367–378. IEEE CS Press, Mar. 2003.
- [14] K. Kageura. *The Dynamics of Terminology*. John Benjamins, 2002.
- [15] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length Normalization in XML Retrieval. In *Proc. of the 27th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 80–87. ACM Press, July 2004.
- [16] M. Kaszkiel and J. Zobel. Passage Retrieval Revisited. In *Proc. of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185. ACM Press, July 1997.
- [17] J. Nielsen. Do Websites Have Increasing Returns? <http://www.useit.com/alertbox/9704b.html>, Apr. 1997. Jakob Nielsen’s Alertbox for April 15, 1997.
- [18] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura. XRel: A Path-Based Approach to Storage and Retrieval of XML Documents using Relational Databases. *ACM Transactions on Internet Technology*, 1(1):110–141, June 2001.

An algebra for Structured Queries in Bayesian Networks

Jean-Noël Vittaut
LIP6, Paris, France

vittaut@poleia.lip6.fr

Benjamin Piwowarski
LIP6, Paris, France

bpiwowar@poleia.lip6.fr

Patrick Gallinari
LIP6, Paris, France

gallinar@poleia.lip6.fr

ABSTRACT

We present an algebra to represent structured information queries which can be used in a Bayesian Network framework. This framework allows us to consider *content-only* (CO) queries and *content-and-structure* (CAS) queries. We perform the retrieval task using inference in our network. The proposed model can adapt to a specific corpus through parameter learning. Thanks to this algebra, the representation of the information demand is independent from the structured query language. It allows us to answer both vague and strict structured queries.

Keywords

Bayesian networks, INEX, XML, Focused retrieval, Structured document retrieval, Algebra

1. INTRODUCTION

The goal of our model is to provide a generic system for performing different Information Retrieval (IR) tasks on collections of structured documents. We take an IR approach to this problem. We want to retrieve specific relevant document elements as an answer to a query. The elements may be any document or document part (full document, section(s), paragraph(s), etc.) indexed from the structural description of the collection. The Okapi and Bayesian Network (BN) models are described in section 2 and we give the results of these models for CO queries. In section 3, we present our algebraic approach for the evaluation of CAS queries, and we give official INEX 2004 results.

2. CONTENT ONLY QUERIES

2.1 Okapi model

In this section, we present the Okapi model which was also used for experiments on the INEX 2004 database. Then, we describe the way we use this model with Bayesian Networks. Lastly, we give the results of the experiments we conducted on the INEX corpus.

We used Okapi as a standalone model and also as a local baseline model for Bayesian Networks. It allows us to compute a local score in each doxel (a document element) of the database. Then, this score is used to order the results (if we use the Okapi model alone) or as a source of evidence for Bayesian Networks.

Bayesian Networks needs baseline models that give a score which can be interpreted as a probability (of relevance). So, we adapted Okapi [19] in order to:

- reach reasonable performances on the INEX corpus (and on a structured collection in general);

- compute a score which could be interpreted as a probability with this model.

The local score of a doxel x for a given query q , computed by the Okapi model, is defined by:

$$\text{Okapi}(q, x) = \sum_{j=1}^{\text{length}(q)} \omega_{j,x} \frac{(k_1 + 1)tf_{x,j}}{K_x + tf_{x,j}} \times \frac{(k_3 + 1)qtf_j}{k_3 + qtf_j}$$

Where k_1 and k_3 are constants, $\text{length}(q)$ is the number of terms in query q . This formula is similar to classical Okapi except for the index x appearing in ω , K and tf . Okapi makes use of different statistics relative to the document collection such as term occurrences or mean document length. Since for Structured Information Retrieval (SIR) elements to be retrieved are doxels and not plain documents, these statistics have to be adapted. Values $\omega_{j,x}$ and K_x are defined as follows:

- $\omega_{j,x} = \log\left(\frac{N - n_j + 0.5}{n_j + 0.5}\right)$. In Okapi N is the number of documents in the collection and the number of documents containing term j . There are different options for adapting these collection statistics to SIR. We will present here tests where these two values were defined respectively with respect to the classical document set ("document frequency") as in Okapi.
- $K_x = k_1 \left((1-b) + b \frac{dl}{avdl} \right)$ where b is a constant and in

Okapi dl is the document length and $avdl$ is the average document length. Here dl was replaced by the doxel length and one weighting scheme was tested for $avdl$: the average length taken respectively over all the doxel with the same tag ("tag").

We chose this peculiar weighting scheme as it allowed us to reach good performances when used by our BN model. As we said, we needed scores which can be interpreted as probabilities. Okapi score does not range between 0 and 1. The normalization of Okapi is discussed in [18] in the context of filtering, where it is proposed to make a regression of the original Okapi score via a logistic function. We used this idea here with the following transformation:

$$P(M_{\text{Okapi}}(x) = R|q) = \frac{1}{1 + e^{\alpha \times \text{Okapi}(q,x) / \text{length}(q) - \beta}}$$

This formula gives the normalized score for the local baseline variants of Okapi model. The α and β parameters were estimated on the whole INEX 2002 database. This score is dependant on the query length. Since the parameters of the logistic function should be valid for queries of varying length, this score was divided by the query length. We then computed the mean okapi score μ and the standard deviation σ for all the CO queries of INEX 2003. We then set α and β such that the probability $P(M_{\text{Okapi}}(x)=R|q)$ is 0.5 when the score is μ and 0.75 when the score is $\mu+\sigma$. These values were chosen empirically.

This is different from [18] where the parameters of the regression are estimated for each query. This would not be realistic here because of the increased complexity of SIR.

2.2 Bayesian Networks

2.2.1 Model

Let us consider a hierarchically structured collection like the INEX corpus. Documents are organised in a category hierarchy with corpus as the root node, journal collections as its immediate descendents, followed by journals, articles etc. We view retrieval for such a collection as a stochastic process in which a user goes deeper and deeper in the corpus structure: the user starts its search at the “root node” of all categories, and then selects one or several categories in which relevant documents should be. For each category, he or she selects subcategories and/or documents within these categories. This process is iterated until the user has found relevant and highly specific doxels.

The BN structure we used directly reflects this document hierarchy and retrieval will follow the above stochastic process. We consider that each structural part within the hierarchy has an associated random variable. The root of the BN is thus a “corpus” variable, its children the “journal collection” variables, etc. The whole collection is thus modelled as a large BN which reflects the doxel hierarchy in the collection.

Each random variable in the BN can take its values in a finite set. Existing BN models for flat [2] or structured [14] documents use binary values (R , $-R$). This is too limitative for SIR since quantifying an element relevance is more complex than for whole documents and should somewhat be related to the two dimensional scale (specificity, exhaustivity) proposed for INEX. We used a state space of cardinality 3, $\mathbf{V} = \{I, B, E\}$ with:

1. **I** for **Irrelevant** when the element is not relevant;
2. **B** for **Big** when the element is marginally or fairly specific;
3. **E** for **Exact** when the element has a high specificity.

In this model, relevance is a local property in the following sense: if we knew that an element is relevant, not relevant or too big, the relevance value of its parent would not bring any new information on the relevance of one of its descendants.

For any element X and for a given query q , the probability $P(X = E|q \wedge X\text{'s parent} = B)$ will be used as the final Retrieval

Status Value (RSV) of this element. Using the simpler RSV $P(X = E|q)$ led to poor performances with the BN. Our choice was partly motivated by the work of Crestani [4][5] and by preliminary experiments.

Besides these variables, there are two more types of random variables in the BN. The first one corresponds to the query need, it is denoted Q and its realization q . Q is a vector of word frequencies taking its values in a multidimensional real space. This random variable is always observed (known). Document textual information is not directly modelled in this BN for complexity reasons. Instead a series of baseline IR models will be used to compute local relevance scores for each doxel given a query. For each local baseline model, this score will only depend on the doxel content and on the query. It is then independent from the context of the doxel in the XML tree. The global score for each doxel will then combine these local scores and will also depend on the doxel context in the BN – the parent’s relevance. These local baseline models have been adapted from classical (flat) retrieval IR models. In the experiments presented here one variant of the Okapi model were used for baseline: the okapi with standard document frequency and a length normalisation over elements with the same tag. In the BN model a random variable is associated to each local scorer and doxel. Let $M(X)$ denote the random variable associated to the local baseline model and doxel X and m its realization. As in classical IR this variable will take two values: R (relevant) and $-R$ (not relevant), i.e. $m \in \{R, -R\}$. The local relevance score at X given query q for the baseline model will be $P(M(X) = R|q)$. Note that it is straightforward to add new baseline models; in the following, all the formulas were adapted to the case where we have only one baseline model.

Based on the local scores $M(X)$ and on the BN conditional probabilities, BN inference is then used to combine evidence and scores for the different doxels in the document model. In our tree like model, the probability that element X is in state I , B or E depends on its parent state and on fact that the local baseline models have judged the element as relevant or not relevant. The probability for X to be in a given state $v \in \mathbf{V}$ is then:

$$P(X = v|q) = \sum_{v_Y \in \mathbf{V}, m \in \{R, -R\}} P(X = v_X | Y = v_Y, M(X) = m) P(M(X) = m|q)$$

In this expression, the summation is over all the possible values of v_Y , m (v_Y can take any value in $\mathbf{V} = \{I, B, E\}$, and each m_i can take values $R, -R$). The conditional probability is expressed as follows:

$$P(X = v_X | Y = v_Y, M = m) = F_X(\Theta, v_X, v_Y, m) = \frac{e^{\theta_{c_X, v_X, v_Y, m}}}{\sum_{v \in \mathbf{V} = \{I, E, B\}} e^{\theta_{c_X, v, v_Y, m}}}$$

Where the $\theta_{c_X, v_X, v_Y, m}$ are real values to be learned. There is one such parameter for each tag category c_X and value set v_X, v_Y, m . All the doxels sharing the same value set c_X, v_X, v_Y, m will share this θ parameter. The denominator ensures that conditional probabilities sum to 1.

2.2.2 Training algorithm

In order to learn the parameters and to fit to a specific corpus, we used a training criterion based on the relative order of elements. We used all the assessed CO topics from the INEX 2003 dataset. The criterion to be minimised was:

$$Q(\Theta) = \sum_q w(q) \sum_{i,j} e^{(RSV(i,q) - RSV(j,q))s_q(i,j)}$$

where the weighted q summation is over the set of all training queries and the i and j ones are over the set of all doxels in the training set. $RSV(i,q)$ is the score of the element X_i and s_q is defined as follows:

$$s_q(i,j) = \begin{cases} 1 & \text{if } X_i \text{ is "better" than } X_j \text{ for query } q (X_i >_q X_j) \\ -1 & \text{if } X_j \text{ is "better" than } X_i \text{ for query } q (X_i <_q X_j) \\ 0 & \text{otherwise } (X_i =_q X_j) \end{cases}$$

The order (“better than”) between the elements depends on the assessments for a given query q . For instance, a highly specific and exhaustive element is “better than” a fairly exhaustive and highly specific one. We used the following partial order between assessments (from “best” to “worst”):

1. Highly specific and highly exhaustive
2. Highly specific and fairly exhaustive
3. Highly specific and marginally exhaustive
4. All other assessment including “not assessed”

The score of an element in the criterion formula is either $P(X = E|q)$ for the model **BN1** or $P(X = E|q \wedge X \text{'s parent} = B)$ for the model **BN2**. The latter is more complex but more related to our scoring algorithm. The weight $w(q)$ was chosen in order to normalize the contribution of different topics: even if the number of assessments were different, this normalization ensured that each topic had the same influence on the criterion. The criterion is minimal when all the elements are ordered according to our partial order.

In order to optimize the criterion, different gradient algorithms could be used. For the experiments we used a simple gradient descent algorithm where the learning rate (epsilon) was automatically set by a line search; for this latter, we use the Armijo algorithm. The number of steps was chosen so as to optimize the performance with respect to the ERR metric. For BN1, a maximum was reached after 195 iterations while for BN2 a maximum was reached after 700.

2.3 Experiments

Three official runs were submitted to INEX'04:

- **Okapi** In this run, we used the Okapi weighting scheme; every volume (and not every doxel) in the INEX corpus was considered as a document while the average document length used in the Okapi formula was local: for every doxel, the average document length was the average length of the doxels with the same tag.
- **BN1** In this run, we submitted the doxel retrieved with the BN which is described in 2.2. The former Okapi

model was used as a base model, $P(X = E|q)$ was used as the score of an element for the learning process, and $P(X = E|q \wedge X \text{'s parent} = B)$ was used as the score of an element.

- **BN2** In this run, we also submitted the doxel retrieved with the BN which were learnt with a different mapping between tag names. $P(X = E|q \wedge X \text{'s parent} = B)$ was used as the score of an element both for learning and testing.

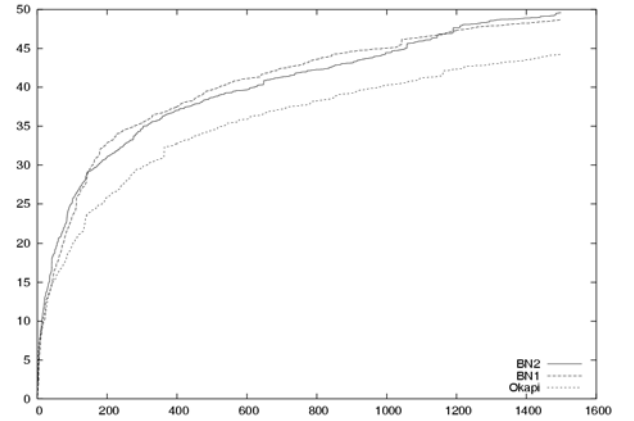


Figure 1: CO official runs (Generalized recall)

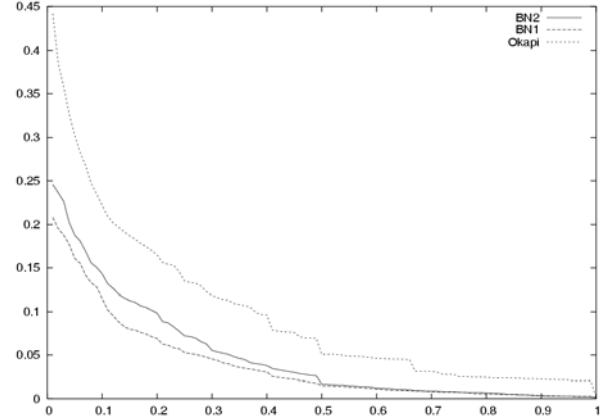


Figure 2: CO official runs (average of all RP measures)

Table 1: CO official runs

	Generalized recall	Average of all RP measures
Okapi	40.74	0.10
BN1	46.45	0.04
BN2	45.97	0.05

The results are summarized in Figure 1, Figure 2 and Table 1. For generalized recall, BN models are clearly above Okapi model whereas curves are inverted for the average of all RP measures.

Contradictions between the different measures do not allow us to conclude which model is better.

With respect to the experiments we have done the two previous years, this ranking criterion seems the most promising one – in INEX 2002 and 2003 we used a maximum likelihood algorithm (EM) which was not well fitted to this task. However, the partial order should be refined so as to be even more close to the “ideal” criterion which is user related. We also want to investigate other criterions such as the cross-entropy.

3. CONTENT AND STRUCTURE QUERIES

In this section, we present the algebra we have used to answer Vague Content and Structure Queries (VCAS) starting from the scores of BN Model or standalone Okapi model. We only give some elements to understand the way we use this algebra in the specific case of NEXI queries. A more detailed description of the algebra is given in [17]. At last, we give the results of the experiments on INEX 2004 for the Okapi model and Bayesian Networks.

3.1 Algebra

3.1.1 Introduction

In INEX, queries are expressed in a query language (NEXI) which is very similar to XPath in which a *vague* operator (about) is introduced in order to allow for queries in a similar fashion than in information retrieval. Such languages can be used to express query needs that mix possibly vague content and structure constraints. XPath is for XML documents what SQL is for databases: it is a language that describes which information should be retrieved from XML documents. In traditional databases, this request is usually mapped into an algebra which in turn is used as a query plan. This query plan is closely related to physical operations that will give the answers to the query. In databases, the result of a formula of the algebra is a set of tuples. In XML databases, the result is a set of elements.

Defining or choosing an algebra is very important to answer complex query needs. This is proved by the important number of works within the semi-structured database field which definite algebras like for example [15][1]. Such approaches are also used in INEX [13]. Our algebra is very closely related to the algebra defined by Fuhr and Grossjohan [8] when we defined our own algebra for XML retrieval. As in classical IR, SIR aim is to retrieve the set of document elements that fulfill a given query need. This query need is very often imprecise. The algebra we define here can be used to answers *vague* queries that have constraints on both content and structure and make use of the Bayesian Networks framework that we use for CO queries.

3.1.2 Algebra

Besides classical operators of the set theory like the intersection, the union and the complementary, our algebra use structural operators like:

- $\overline{desc}(x)$ (descendant or self)
- $\overline{anc}(x)$ (ancestor or self)
- other operators we do not mention because they are useless with the specification of NEXI language.

We denote X the set of all doxels. We introduce three functions:

1. $R(q)$ which returns the set of doxels which are answers to the query need q . That is, a doxel is in the set with a given probability.
2. $comp(t, comparison)$ which returns the set of doxels x where $comparison(x, t)$ is true. We have used $=, \neq, \leq, \geq, <, >$ comparisons.
3. $label(x)$ which returns the label of the doxel (the tag name). The function $label^{-1}(l)$ returns the set of doxels which have a label l (This function is used for SCAS queries). In order to process VCAS queries, we can replace the latter function by a vague one called $invlabel(l)$ which returns a set of labels with a given probability.

The algebra is defined on the set $P(X)$ (the set of all the part of the set of doxels). We use the operator “ \circ ” to compose the different functions defined on $P(X)$ which take values in $P(X)$.

With all these operators and functions, we are able to answer structured queries.

3.1.3 Probabilistic interpretation

In the previous section, $R(q)$ returns the set of doxels that are answers to the query q . In Information Retrieval (IR), the answers to a query are not well defined: the query is expressed in vague terms, and the real query need cannot be easily defined. We thus have to define $R(q)$ as a “*vague*” set in order to compute the answer to a query that contains predicates like *about*.

In our approach, as in the probabilistic interpretation of fuzzy sets [6], a set $A \subset X$ is not anymore defined strictly. We denote such a set by A_v (v for *vague*). A_v is defined by a probability distribution on subsets of X . The case where probability $P(A_v = A) = 1$ means that the set A_v is strict and not vague (the concept of fuzzy set is thus more general than the concept classical set). An element a belongs to A_v with a probability $P(a \in A_v)$ which is formally defined by:

$$P(a \in A_v) = \sum_{A \subset X, a \in A} P(A_v = A)$$

We define recursively the fact that a doxel belongs to a vague set:

$$x \in \varphi(A_v) \equiv \bigvee_{x' \in X, x \in \varphi(\{x'\})} x' \in A_v$$

Lastly, intersection and union operators can also be transformed in logical formulas:

$$x \in A_v \cap B_v \equiv (x \in A_v) \wedge (x \in B_v)$$

$$x \in A_v \cup B_v \equiv (x \in A_v) \vee (x \in B_v)$$

3.1.4 Algebraic expression of a CAS query

In order to convert a NEXI query into an algebraic expression, we briefly define the way we decompose the NEXI Queries used in INEX, which can be easily extended to XPath like queries.

A NEXI query is read from left to right. The different components are separated by two slashes “//” which are not within brackets. The query $//L_0[F_0]//L_1[F_1]...//L_n[F_n]$ can be decomposed in $//L_i[F_i]$ elements.

Each component is itself composed of three parts:

1. **The axis** ($//$). This axis is an abbreviation of the */descendant-or-self* :: axis in XPath. It defines a set with respect to a given doxel x . For the first component of the XPath, this set is defined by the document d . For the first component of an XPath within a filter, the set of selected doxels is evaluated with respect to the document d or to the filtered doxel. For any another component, the selection is made with respect to the set of doxels selected by the previous component ;
2. **The label** (L_i). It filters the set of doxels selected by the axis and keep only those which have the label a . When the label is *, the “axis” set is not filtered ;
3. **The filter** (F_i) that express a boolean condition on doxels. It returns the subset of those doxels which fulfill the boolean conditions expressed in the filter. An XPath can be used in the filter: it is relative to the context path and take the form of

$$//L_0//L_1...//L_n$$

The filter is a condition which can be true or false for one doxel.

An algebraic expression is defined on the parts of the set of doxels. Each part of the query (axis, label L_i , filter F_i) is a component that can be processed separately:

- An axis is transformed into the structural operator $\Psi_A(//) = \overline{desc/}$ except for the first component of the XPath which is transformed into $\Psi_A^{(0)}(//) = \overline{desc/}(d)$.
- A label (or a set of labels) L_i is transformed into a function Ψ_L that selects a subset of doxels which have a label L_i in the set:

$$\begin{aligned} \Psi_L(L_i): \text{P}(X) &\mapsto \text{P}(X) \\ X &\mapsto X \cap \text{label}^{-1}(L_i) \end{aligned}$$

where we handle the special case of * by defining $\text{label}^{-1}(*) = X$

- As for the filter F_i , the transformation is more complex and is denoted Ψ_F :

$$\begin{aligned} \Psi_F(F_i): \text{P}(X) &\mapsto \text{P}(X) \\ X &\mapsto X \cap \Psi'_F(F_i) \end{aligned}$$

where Ψ'_F is the function which transforms a filter in the set of doxels that fulfill the conditions expressed in the filter.

With these notations, the query:

$$p = //L_0[F_0]//L_1[F_1]...//L_n[F_n]$$

is the result of the evaluation of the algebraic expression:

$$\begin{aligned} \Psi(d, p) &= \Psi_F(F_n) \circ \Psi_L(L_n) \circ \Psi_A(//) \\ &\dots \\ &\circ \Psi_F(F_1) \circ \Psi_L(L_1) \circ \Psi_A(//) \\ &\circ \Psi_F(F_0) \circ \Psi_L(L_0) \circ \Psi_A^{(0)}(//) \\ &= \Psi'_F(F_n) \cap \text{label}^{-1}(L_n) \\ &\quad \left(\dots \right. \\ &\quad \left. \cap \overline{desc/} \left(\overline{desc/} \left(\Psi'_F(F_1) \cap \text{label}^{-1}(L_1) \right) \right) \right. \\ &\quad \left. \left. \cap \overline{desc/} \left(\Psi'_F(F_0) \cap \text{label}^{-1}(L_0) \right) \right) \right) \end{aligned}$$

We do not detail the way we evaluate Ψ'_F . We use a similar method than above in which:

- predicates *about* are transformed into a $R(q)$ function.
- comparisons are transformed using $\text{comp}(t, \text{operator})$ function.

3.2 Experiments

To compute the union or the intersection of two vague sets, we used the probabilistic and/or operators defined below:

- $P(a \in A \wedge b \in B) = P(a \in A)P(b \in B)$
- $P(a \in A \vee b \in B) = P(a \in A) + P(b \in B) - P(a \in A)P(b \in B)$

Complement remains $P(a \notin A) = 1 - P(a \in A)$. We have also tested min/max and Lukasiewicz operators, but they were outperformed the probabilistic operator.

In order to introduce vagueness into the query structure, we used the following labelling function:

$$\text{invlabel}(l) = \bigcup_{x \in X, \text{label}(x) = l} \text{pa}(x) \cup \{x\} \cup \{y \in X, \text{pa}(y) = x\}$$

where we supposed that all the doxels from this set have the same probability of being labelled l . We have also tested the labelling function $\text{label}^{-1}(l)$ function, and other simple strategies to introduce vagueness into structure (not considering tag names), but they were outperformed by $\text{invlabel}(l)$.

- Three official runs were submitted to INEX'04; the models we used to computed the probability of relevance are the same that in section 2.3.

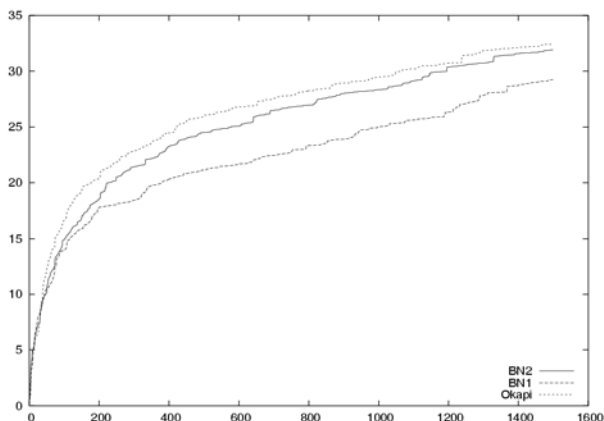


Figure 3: VCAS official runs (Generalized recall)

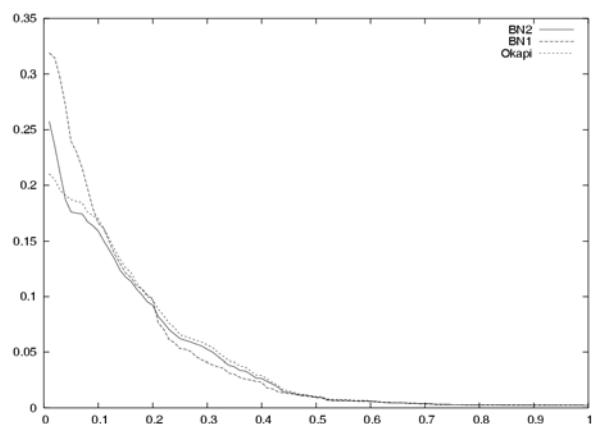


Figure 4: VCAS official runs (average of all RP measures)

Table 2: VCAS official runs

	Generalized recall	Average of all RP measures
Okapi	33.14	0.05
BN1	27.97	0.05
BN2	31.67	0.04

The results are summarized in Figure 3, Figure 4 and Table 2. Okapi model outperform BN1 and BN2 models but not significantly. We remark results are inverted in comparison with experiments on CO queries.

Our algebra can answer all INEX VCAS and also more complex structured queries. Nevertheless, the connection between CO queries and VCAS queries is not clear because the best model for CO queries is not the best one for VCAS queries. The Okapi model gives better results for structured queries than for content only queries. Moreover, the choice of union and intersection functions for aggregation like min/max, Lukasiewicz or probabilistic is not also clear regarding the base models we use.

4. CONCLUSION

We introduced a new BN model whose conditional probability functions are learnt from the data via a gradient descent algorithm. The BN framework has some advantages. Firstly, it can be used in distributed IR, as we only need the score of the parent element in order to compute the score of any its descendants. Secondly, it can use simultaneously different baseline models: we can therefore use specific models for non textual media (image, sound, etc.) as another source of evidence.

We have described a new algebra we have used to process *content-and-structure* queries. This algebra is a generic way to represent structured queries and can be easily used with the IR system based on Bayesian Networks we have developed.

Our system can answer CO and VCAS queries. The model has still to be improved, tuned and developed. In particular, we should improve the baseline models and fit them to the specificities of CO or VCAS queries. We show this algebra allows answering VCAS queries and we have to investigate new ways of including vagueness in the structure of queries.

5. REFERENCES

- [1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.
- [2] Callan J, Croft WB, and Harding SM (1992) The INQUERY Retrieval System. In A. Min Tjoa and Isidro Ramos, editors, Database and Expert Systems Applications, Proceedings of the International Conference, pages 78-83, Valencia, Spain, 1992. Springer-Verlag.
- [3] Y. Chieramella, P. Mulhem, and F. Fourel. A Model for Multimedia Information Retrieval. Technical report, IMAG, Grenoble, France, July 1996.
- [4] Crestani F, de Campos LM, Fernandez-Luna JM and Huete JF (2003), A multi-layered Bayesian network model for structured document retrieval, *ECSQARU*, LNAI 2711, Springer-Verlag,74-86.
- [5] Crestani F, de Campos LM, Fernandez-Luna JM and Huete JF (2003), Ranking Structured Documents Using Utility Theory in the Bayesian Network Retrieval Model, In *SPIRE (String Processing and Information Retrieval)*, Brazil, 2003, pp. 168-182.
- [6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from incomplete data via de EM algorithm. *The Journal of Royal Statistical Society*, 39:1{37, 1977.
- [7] N. Fuhr and T. Rölleke. HySpirit - a Probabilistic Inference Engine for Hypermedia Retrieval in Large Databases. In H.-J. Schek, F. Saltor, I. Ramos, and G. Alonso, editors, *Proceedings of the 6th International Conference on Extending Database Technology (EDBT)*, Valencia, Spain, 1998. Springer, Berlin.
- [8] N. Fuhr and K. Grossjohann. XIRQL: A query language for information retrieval in XML documents. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *The 24th International Conference on Research and Development*

- in *Information Retrieval*, New Orleans, Louisiana, USA, Sept. 2001. ACM.
- [9] T. Grabs and H.-J. Schek. ETH Zrich at INEX: exible information retrieval from XML with PowerDB-XML. Dec. 2002.
- [10] G. Kazai, M. Lalmas, and T. Rölleke. A Model for the Representation and Focussed Retrieval of Structured Documents based on Fuzzy Aggregation. In *String Processing and Information retrieval (SPIRE 2001) Conference*, Laguna de San Rafael, Chile, Sept. 2001.
- [11] M. Lalmas. Dempster-Shafer's Theory of Evidence Applied to Structured Documents: Modelling Uncertainty. In *Proceedings of the 20th Annual International ACM SIGIR*, pages 110-118, Philadelphia, PA, USA, July 1997. ACM.
- [12] M. Lalmas and E. Moutogianni. A Dempster-Shafer indexing for the focussed retrieval of a hierarchically structured document space: Implementation and experiments on a web museum collection. In *6th RIAO Conference, Content-Based Multimedia Information Access*, Paris, France, Apr. 2000.
- [13] J. List, V. Mihajlovic, A. P. de Vries, and G. Ram´irez. The TIJAX XML-IR system at INEX 2003. In *N. Fuhr, M. Lalmas, and S. Malik, editors, INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop*, Dagstuhl, Germany, Dec. 2003.
- [14] Myaeng SH, Jang DH, Kim MH and Zhoo ZC (1998) A Flexible Model for Retrieval of SGML documents. In *W. Bruce Croft, Alistair Moffat, C.J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 138-140, Melbourne, Australia, August 1998. ACM Press, New York.
- [15] G. Navarro and R. Baeza-Yates. Proximal nodes: A model to query document databases by content and structure. *ACM TOIS*, 15(4):401–435, Oct. 1997.
- [16] B. Piwowarski, G.-E. Faure, and P. Gallinari. Bayesian networks and INEX. In *Proceedings of the First Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX)*, DELOS workshop, Dagstuhl, Germany, Dec. 2002. ERCIM.
- [17] B. Piwowarski and P. Gallinari. An algebra for probabilistic XML Retrieval. In *The First Twente Data Management Workshop*, 2004
- [18] S. Robertson. Threshold setting and performance optimization in adaptive Filtering. *Information Retrieval*, 5(2-3):239-256, April-July 2002.
- [19] Walker S and Robertson SE (1999) Okapi/Keenbow at TREC-8. In *E. M. Voorhees and D. K. Harman, editors, NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, Maryland, USA, November 1999.

IR of XML documents – A collective Ranking Strategy

Maha Salem
Faculty of Electrical Engineering,
Computer Science and Mathematics
University of Paderborn
Warburger Str. 100
33098 Paderborn
Germany
MahaSalem@web.de

Alan Woodley
Centre for Information Technology
Innovation
Faculty of Information Technology
Queensland University of Technology
GPO Box 2434 Brisbane Q 4001
Australia
ap.woodley@student.qut.edu.au

Dr Shlomo Geva
Centre for Information Technology
Innovation
Faculty of Information Technology
Queensland University of Technology
GPO Box 2434 Brisbane Q 4001
Australia
s.geva@qut.edu.au

ABSTRACT

Within the area of Information Retrieval (IR) the importance of appropriate ranking of results has increased markedly. The importance is magnified in the case of systems dedicated to XML retrieval, since users of these systems expect the retrieval of highly relevant and highly precise components, instead of whole document retrieval. As an international, coordinated effort to evaluate the performance of Information Retrieval systems, the Initiative for the Evaluation of XML retrieval (INEX) encourages participating organisation to run queries on their search engines and to submit their result for the annual INEX workshop. In previous INEX workshops the submitted results were manually assessed by participants and the search engines were ranked in terms of performance. This paper presents a Collective Ranking Strategy that is supposed to facilitate the derivation of a ranking of participating search engines and moreover provides a system that outperforms all other search engines.

Keywords

Information Retrieval, Document Standards, Digital Libraries

1. INTRODUCTION

Modern society unceasingly produces and uses information. All technical activity – in science, industry, commerce or government – now takes place in such a complex environment that it must be based on specially acquired information. At the same time, every act gives rise to information, and recorded knowledge grows rapidly. To find the relevant information sought within the huge mass of information now available becomes ever more difficult. If information is supposed to be accessible it must be organized [1].

The specific nature of information has called for the development of many new tools and techniques for information retrieval. Modern information retrieval (IR) deals with storage, organisation and access to text, as well as multimedia information resources. The concept of information retrieval requires that there are some documents containing information that have been organised in an order suitable for easy retrieval [2].

Within the area of information retrieval, keyword search querying has emerged as one of the most effective paradigms for IR, especially over HTML documents in the World Wide Web. One of the main advantages of keyword search querying is its simplicity – users do not have to learn a complex query language, and can issue queries without any prior knowledge about the structure of the underlying data. Since the keyword search query interface is very flexible, queries may not always be precise and can potentially return a large number of query results, especially in large document collections. As a consequence, an important requirement for keyword search is to rank the query results so that the most relevant results appear first.

Despite the success of HTML-based keyword search engines, certain limitations of the HTML data model make such systems ineffective in many domains. These limitations stem from the fact that HTML is a presentation language and hence cannot capture much semantics. The XML (eXtensible Markup Language) data model addresses this limitation by allowing for extensible element tags, which can be arbitrarily nested to capture additional semantics. Information such as titles, references, sections and sub-sections are explicitly captured using nested, applicationspecific XML tags, which is not possible using HTML.

Given the nested, extensible element tags supported by XML, it is natural to exploit this information for querying. One approach is to use sophisticated query languages based on Xpath to query XML documents. While this approach can be very effective in some cases, a disadvantage is that users have to learn a complex query language and understand the schema of underlying XML.

Information retrieval over hierarchical XML documents, in contrast to conceptually flat HTML documents, introduces many new challenges. First, XML queries do not always return whole documents, but can return arbitrarily nested XML elements that contain the information needed by the user. Generally, returning the “deepest” node usually gives more context information. Second, XML and HTML queries differ in how query results are ranked. HTML search engines usually rank documents partly based on their hyperlinked structure [3]. Since XML queries can return nested elements, as against

entire XML documents, ranking has to be done at the granularity of XML elements, which requires complicated computing due to the fact that the semantics of containment links (relating parent and child elements) is very different from that of hyperlinks. As a consequence, techniques for computing rankings exclusively based on hyperlinks are not directly applicable for nested XML elements [4].

This paper presents an approach for effective ranking of XML result elements in response to a user query by considering the results of several other search engines and producing a collective ranking on the basis of some sort of a vote. The hypothesis is that the resulting system will outperform all search engines delivering the results it is based on.

1.1 Overview of INEX

Extensible Markup Language (XML) has become a widely accepted standard for the representation and exchange of data, attracting growing attention in digital libraries, scientific data repositories and on the web. The widespread use of XML documents led to the development of appropriate information retrieval (IR) methods for XML documents in the form of XML retrieval systems. These systems exploit the logical structure of documents, which is explicitly represented by the XML markup, to retrieve document components, instead of entire documents, in response to a user query. This means that an XML retrieval system needs not only to find relevant information in the XML documents, but also determine the appropriate level of granularity to return to the user, and this with respect to both content and structural conditions [5]. The expansion in the field of information retrieval caused the need to evaluate the effectiveness of the developed XML retrieval systems.

To facilitate research in XML information retrieval systems the INitiative for the Evaluation of XML retrieval (INEX) has established an international, coordinated effort to promote evaluation procedures for content-based XML retrieval. INEX provides a means, in the form of a large XML test collection and appropriate scoring scheme for the evaluation of XML retrieval systems [6]. The test collection consists of XML documents, predefined queries and assessments. The scoring scheme is based upon two dimensions: *specificity* (reflects the relevancy of a particular XML component) and *exhaustiveness* (measures whether a relevant component contains suitable coverage). These values are quantised to the traditional metrics of *precision* (the probability that a result element viewed by a user is relevant) and *recall* (the total number of relevant components returned). In turn, these metrics are combined to form a recall/precision curve.

Together they provide a means for qualitative and quantitative comparison between the various competitors participating at INEX. Each year the competitors' systems are ranked according to their overall effectiveness.

2. RANKING OF RESULTS

Ranking of results has a major impact on users' satisfaction with search engines and their success in retrieving relevant documents. While searches may retrieve thousands of hits, search engine developer claim their systems place items that best match the search query at the top of the results list.

Since users often do not have time to explore more than the top few results returned, it is very important for a search engine to be able to rank the best results near the top of all returned results. A study conducted by [7] indicates that 80% of users only view the first two pages of results. The user may consider a number of factors in deciding whether or not to retrieve a document. Regardless of relevance-ranking theory, users have an intuitive sense of how well the relevance ranking is working, and a key indicator of this intuitive satisfaction is the number of distinct query words that a document contains. E.g., a document containing only two query words from an eight-word query should not be higher ranked than a document containing all eight words [8].

2.1 Collective Ranking

As described before, the INEX workshop is run once a year and is generally based on the following steps:

1. Participating organisations contribute topics (end user queries) and a subset of topics is selected for evaluation.
2. The topics are distributed to participants who run their search engines and produce a ranked list of results for each topic.
3. The results are pooled together (disassociated and duplicates eliminated).
4. The pooled results are individually assessed by the original topic contributors, who act as end users manually assessing the relevance of the results in terms of exhaustiveness and specificity.
5. The search engines are ranked in terms of performance (recall/precision) using several metrics.
6. Results are returned to participants who in turn write up and present their systems and discuss it at the workshop.

During the last two years the execution of step 4 (assessment of topics by human assessors) has emerged as a very time-consuming procedure which led to the idea of a "Collective Ranking Strategy". The idea is to take the entire set of results from all search engines and produce a collective ("committee") ranking by taking some sort of a vote.

The collectively ranked results are to be evaluated against the assessed pool of results (as determined by the human assessors). The hypothesis is that it may be possible to outperform any single system by taking account of the results from all systems. If this hypothesis is verified, then as a consequence manual assessment of pooled results by

human assessors (step 4) may be no longer required. Instead, a relative comparison of submissions with the collective ranking results will be sufficient to derive a ranking of all search engines. Moreover, this would also prove the assumption that you can derive a better performing search engine by solely considering results of several other search engines.

2.2 Strategy

Several strategies were tested and the specifics of the Collective Ranking were determined. During the testing phase it became obvious that the simplest strategy led to best results. The eventually applied Collective Ranking Strategy is described by the following algorithm, which is to be applied separately for both CAS and CO topics:

```

For each topic  $t_1 \dots t_n$ :
{
  For each submission  $s_1 \dots s_m$  for topic  $t_i$ :
  {
    Take the top  $x$  result elements;

    Assign a value  $p_i$  (points for ranking position) to each rank  $r_i \in [1 \dots x]$  of the top  $x$  submitted result elements applying the following formula:

     $p_i := (x - r_i) + 1$ ;
  }

  Compute a total result element score res_scorei for each unique submitted result element as follows (m being the number of submissions):

  
$$\forall x_i \in \text{result elements: } \text{res\_score}_i := \sum_{i=1}^m p_i^k$$


  Rank the result elements according to the assigned result element score  $\text{res\_score}_i$  in descending order;

  In the format of a submission file write the top 1500 result elements of the ranked list into the Collective Ranking output file;
}

```

Within this algorithm, x (\triangleq number of top ranked result elements taken from each submission for each topic) and k (\triangleq weighting for p_i) are variables whose optimal values are to be determined according to the best possible results in the testing phase.

The basic idea of this strategy is to take into account both the number of *occurrences* and the *ranking* position of each result element submitted by the participants' search engines. With reference to the number of occurrences, the summation in the algorithm makes sure that, the more often the same result element appears in the submitted result lists of various search engines, the higher it is rated and eventually ranked. This becomes evident considering an implication provided by the algorithm: If a particular result element is not returned in a search engine's top 100 results, it receives 0 points for the ranking position (p_i). With respect to the consideration of the ranking position,

the definition and incorporation of p_i (points for ranking position) makes sure that, the higher the ranking position of the same result element in each submitted result list is, the bigger the value p_i for each occurrence will be.

As the Collective Ranking is derived from a descending list of the top 1500 result element scores, the bigger the value p_i for each occurrence of a particular result element and as a consequence the bigger the result element score res_score_i (derived from the summation of p_i) is, the better the final ranking position of this particular result element in the Collective Ranking will be.

3. TESTING

The Collective Ranking algorithm was tested using different values for the variables x and k , in order to identify the optimal combination of these values. In the testing phase it became evident that the bigger the depth value x (\triangleq number of top ranked result elements taken from each submission for each topic) is, the bigger the applied value for k (\triangleq weighting for p_i) is supposed to be in order to obtain optimal results.

Furthermore, the algorithm was tested comparing results when considering all submissions and merely considering the top ten ranked submissions (as determined by INEX 2003), respectively. It became obvious that considering all submissions instead of solely considering those submissions ranked as the top ten in INEX 2003 led to better results while retaining the same values for x and k .

Figure 3.1 presents an example of the different effect on results when considering all submissions and only top ten submissions of INEX respectively. Figure 3.2 and 3.3 show examples of test results with different values for x and constant value for k and vice versa respectively.

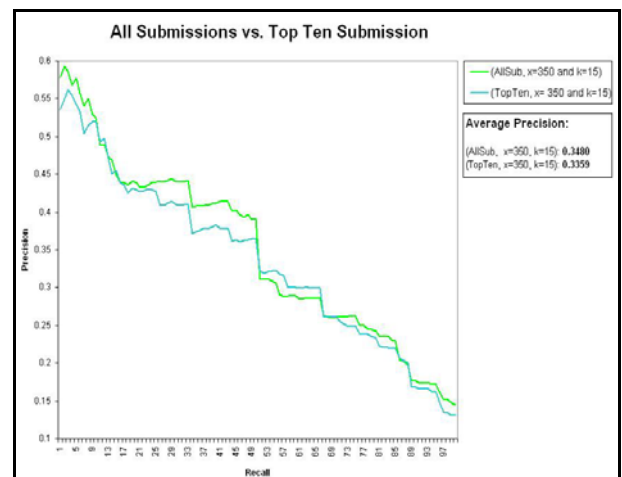


Figure 3.1: Comparison of results considering all submissions / only top ten submissions of INEX

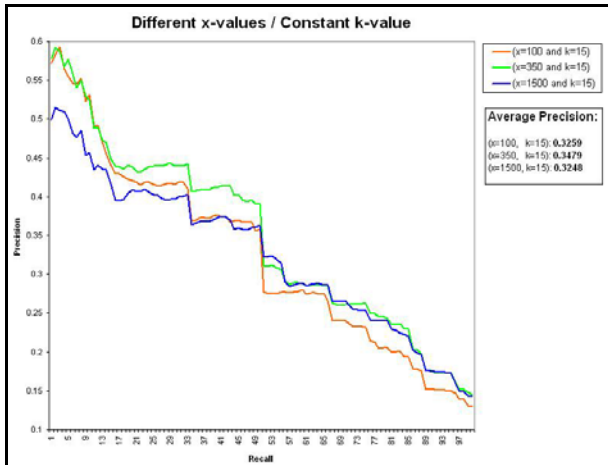


Figure 3.2: Example of test results with different values for x and constant value for k

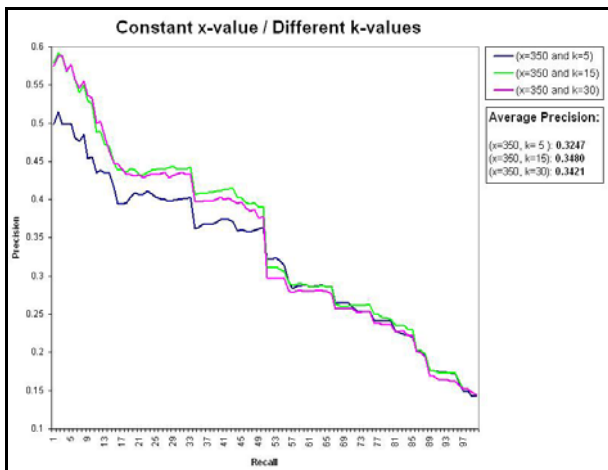


Figure 3.3: Example of test results with constant value for x and different values for k

4. RESULTS

After executing the described algorithm for both CAS and CO topics and submitting the obtained Collective Ranking result file to the INEX evaluation software the hypothesis was verified: The recall/precision curve as well as the average precision of the Collective Ranking outperformed all other systems' submissions.

In this context, best results for the different tasks and quantisations were achieved with the following values for x and k :

- CAS strict: $x = 400$ and $k = 18$
- CAS generalised: $x = 1000$ and $k = 30$
- CO strict: $x = 1500$ and $k = 39$
- CO generalised: $x = 1500$ and $k = 39$

Table 4.1 displays the best values of average precision achieved by the Collective Ranking in comparison with the best ranked submissions of participants in INEX 2003.

The Precision/Recall curves represented in Figures 4.1 to 4.4 demonstrate the performance of the Collective Ranking (displayed in red colour) in comparison with all other submissions of INEX 2003.

Table 4.1: Comparison of best values of Avg. Precision of participants/Collect. Ranking (*Univ. of Amsterdam)

Task	Quantisation	Avg. Precision - Best Value	
		Participants	Collect. Ranking
CAS	Strict	0.3182 [*]	0.3480
CAS	Generalised	0.2989 [*]	0.3177
CO	Strict	0.1214 [*]	0.1339
CO	Generalised	0.1032 [*]	0.1210

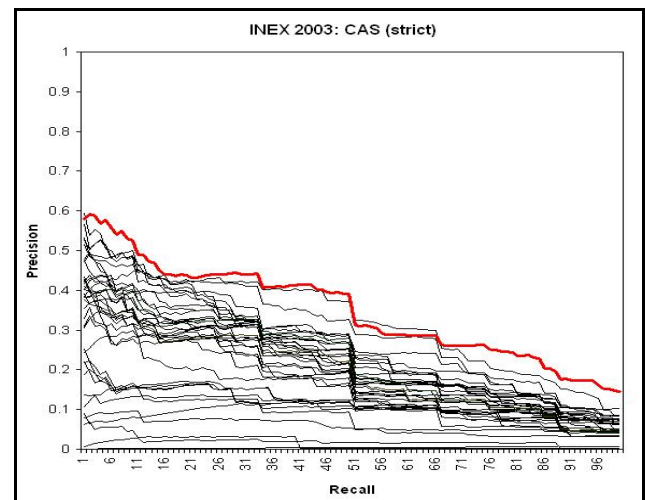


Figure 4.1: Results – CAS (strict)

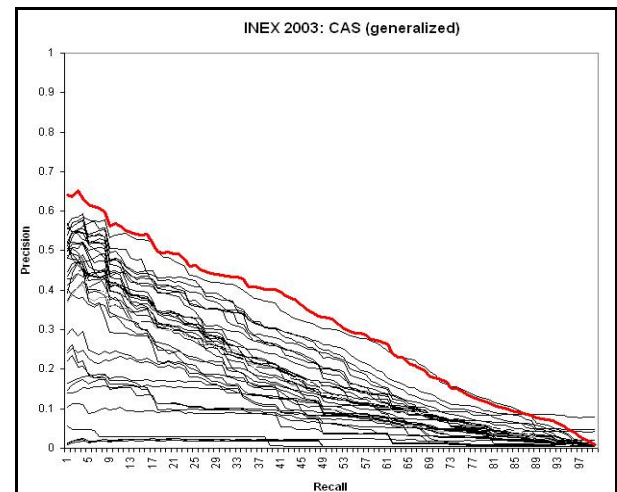


Figure 4.2: Results – CAS (generalised)

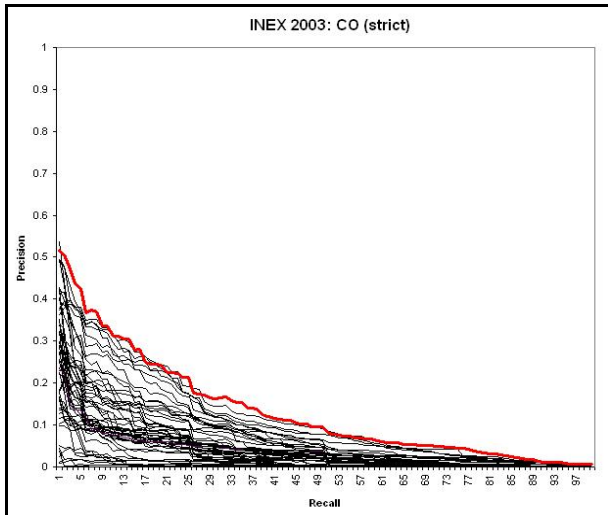


Figure 4.3: Results – CO (strict)

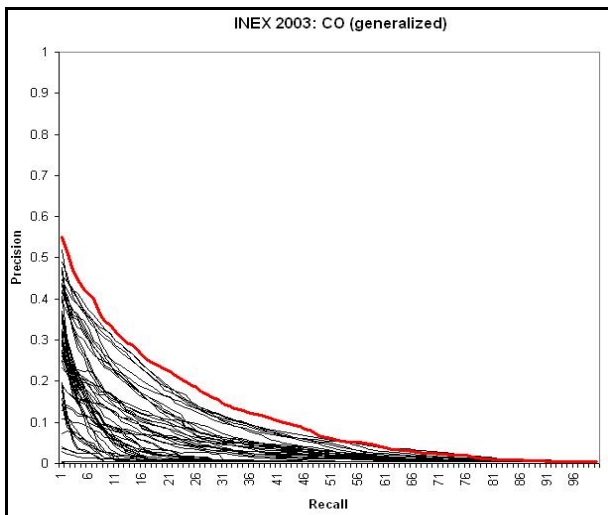


Figure 4.4: Results – CO (generalised)

5. OUTLOOK AND FUTURE WORK

The development and implementation of a Collective Ranking Strategy as presented in this thesis and the results obtained establish a basis for copious future work. This chapter gives an overview of challenges and ideas of approaches for further research on this topic.

5.1 Realistic Assessment

In order to identify the extent of possible improvement regarding the Collective Ranking, a programme indicating the notionally maximum performance was implemented, which is based on the following idea:

During the assessment process of the INEX workshop human assessors determine the relevancy of result elements returned by the participants' systems and pooled together in the pool of results in terms of exhaustiveness and specificity. While exploring the XML files of the INEX document collection with respect to the result elements returned for a certain topic, assessors may add elements of the XML files that were not returned by any

participating system but, however, are considered relevant to the pooled results. This procedure yields a pool of results referred to as the “*Official Perfect Pool of Results*”, which provides the basis for the *INEX Official Assessment Files* that are required for the evaluation of the search engines' performance. These assessment files suggest an *idealish* ranking of particular result elements for each topic representing a guideline for the assessment of the actually returned results. This ranking is referred to as “*idealish*” since elements from some relevant articles might not be included in the top 100 results from any submission, hence are not in the pool of results at all. Adding these elements to the pool would make it theoretically possible to achieve an even better performance. However, due to the fact that some result elements contained in those idealish assessment files are manually added and not returned by any single system, it is not realistic to expect the search engines to actually retrieve these result elements. Therefore, it is equally unlikely that the Collective Ranking system could perform as good as the official ideal results, since it is solely based on the results actually returned by the participants' search engines.

In order to set a more realistic benchmark to identify the (theoretically) best possible performance of the Collective Ranking system, a so-called “*Realistic Perfect Pool of Results*” is to be established. The appendant programme developed to derive the required *Realistic Assessment Files* eliminates all result elements not actually submitted by any participant's search engine from the “*idealish*” Official Perfect Pool of Results.

Figures 5.1 to 5.4 display the performance of the Collective Ranking system compared with the precision/recall curves of the “*Official Perfect*” and “*Realistic Perfect*” Results. They reveal the remarkably big capability of improvement regarding the Collective Ranking Strategy.

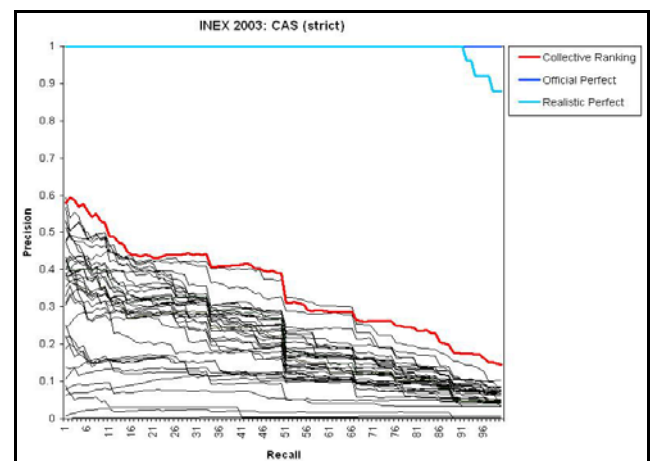


Figure 5.1: Collective Ranking compared with “*Official Perfect*” and “*Realistic Perfect*” results (CAS – strict)

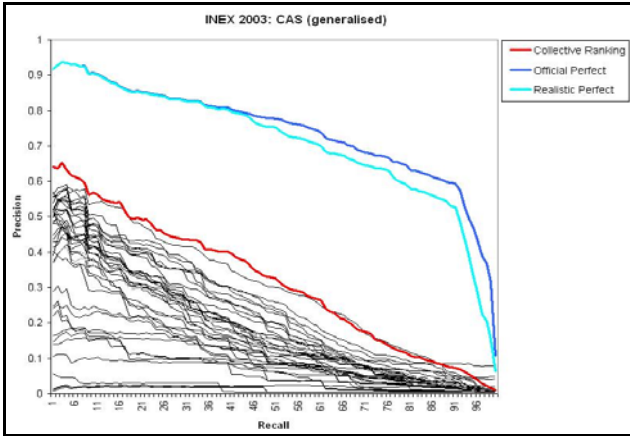


Figure 5.2: Collective Ranking compared with “Official Perfect” and “Realistic Perfect” results (CAS-generalised)

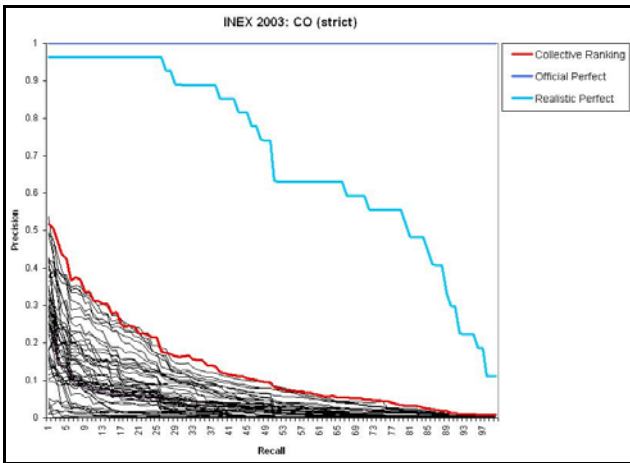


Figure 5.3: Collective Ranking compared with “Official Perfect” and “Realistic Perfect” results (CO-strict)

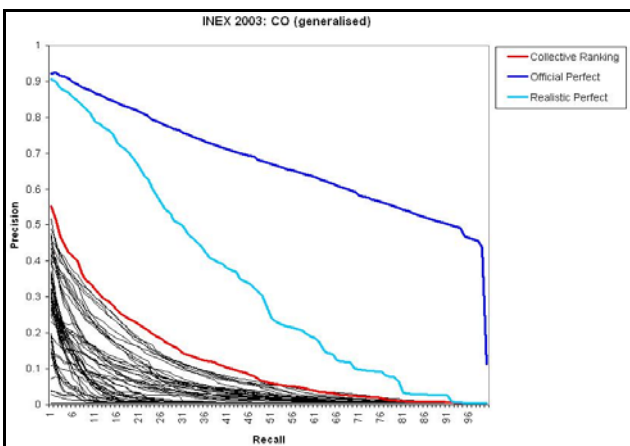


Figure 5.4: Collective Ranking compared with “Official Perfect” and “Realistic Perfect” results (CO-generalised)

Surveying these results it is particularly striking to see that the precision/recall curves of the Realistic Perfect Results

are remarkably better performing than the Collective Ranking, although the Realistic Perfect “system” avails itself of the same source – solely consisting of result elements returned by INEX participants – that is also available for the Collective Ranking system. This emphasises the crucial importance of successful ranking of returned results and therefore represents a point of origin for further examinations.

5.2 Modification of Algorithm

A possible approach to improve the performance of the Collective Ranking system is the modification of the algorithm applied for the implementation of the Collective Ranking Strategy. In this context two practical ideas are described as follows:

1. Quality Factor:

The main idea is the introduction and implementation of a so-called Quality Factor which represents an iterative assignment of a value q_i (0, 1] to each submission depending on its performance in relative comparison with the Collective Ranking. In this regard the definition of the result element score res_score_i (currently derived from the summation of p_i only) would be the following:

$$\forall x_i \in \text{result elements: } res_score_i := \sum_{i=1}^m (p_i^k * q_i^j)$$

Initially, for the first run q_i equals 1 for every submission. After this initial run, a first ranking of submissions can be derived from relative comparison with the Collective Ranking and an individual value for q_i (0, 1] can be assigned for each submission applying the following formula (with m = number of submissions and sr_i = rank of submission i according to submission ranking derived from previous run compared with Collective Ranking):

$$q_i := (m - sr_i + 1) / m$$

This means for example, if there are ten submissions, the submission ranked first achieves the value ($q_i = 1$) for its individual quality factor whereas the submission ranked tenth will be assigned a quality factor value of ($q_i = 0.1$) only. Consequently, as the Collective Ranking is derived from a descending list of the top 1500 result element scores res_score_i , the bigger the value p_i and the bigger the value q_i for each occurrence of a particular result element is, the better the final ranking position of this particular result element in the Collective Ranking will be.

Implementing the idea of a quality factor q_i would emphasise the impact of better performing

submissions and as a consequence might lead to a better performance of the Collective Ranking system.

2. Improvement of Ranking:

As the Realistic Pool results have revealed that most of the result elements contained in the official INEX assessment files have actually been submitted by participants and as a consequence must be accessible for the Collective Ranking, it becomes obvious that an improved ranking of results for both the INEX submissions and the Collective Ranking could be the key for noticeable improvement of performance. However, at present it is not quite clear yet how this idea can be translated into successful methods.

5.3 Automatic Testing

At this stage, values identified best for x and k applied in the Collective Ranking programme are based on results derived from experimental testing. However, since possible values for x can range from 1 to 1500 and appropriate values for k can theoretically range from 0 to infinite, it was not possible to test all possible combinations of these two values. Therefore it is conceivable that better “optimal” combinations may be identified by using automated testing methods which in turn requires the assignment of an adequate implementation.

5.4 Automatic Assessment

At the present time, *INEX Assessment Files* that are used for the evaluation of submissions are derived from assessments conducted by human assessors who work through the INEX document collection to identify relevant result elements. Since this has emerged as a very time-consuming procedure, future work and development with respect to the Collective Ranking could benefit the INEX workshop at such a rate that human assessments might eventually be replaced by assessment and ranking of submissions derived from a relative comparison of those submissions with the Collective Ranking. For this purpose, however, *Automatic Assessment Files* are to be established within the scope of further research and testing.

6. CONCLUSION

The results achieved within the scope of this research project by the development and implementation of a Collective Ranking Strategy may benefit the future procedure of the INEX workshop since – although not yet a suitable substitute for human assessments of results – a ranking of participating search engines can now be derived without manual assessment.

The hypothesis stated at the beginning of this project, suggesting that it may be possible to outperform any single system by taking account of the results from all systems was verified. Moreover it was proven that an outperforming search engine can be developed on the

basis of other search engines’ results. However, the results derived from the implementation of the Realistic Pool Assessment Programme revealed that there is still much room for improvement. Therefore, ample research on the reasons for the performance of the Collective Ranking system will be required in order to identify means to improve the current results.

These conclusions will provide a basis for further research on this topic, especially for the automatic assessment and ranking of search engines, and may be considered a starting point for the exploration of new challenges regarding ranking strategies within this area of modern Information Retrieval.

7. REFERENCES

- [1] B. C. Vickery. “The Need for Information”. In *Techniques of Information Retrieval*, p 1, London, 1970.
- [2] G. G. Chowdhury. “Basic concepts of information retrieval systems”. In *Introduction to Modern Information Retrieval*, pp 1-2, London, 2004.
- [3] S. Brin, L. Page. “The Anatomy of a Large-Scale Hypertextual Web Search Engine”, WWW Conf., 1998.
- [4] L. Guo, F. Shao, C. Botev, J. Shanmugasundaram. “Ranked Keyword Search over XML Documents”, p.1, San Diego, CA, June 9-12, 2003.
- [5] N. Fuhr and S. Malik. Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2003. In *INEX 2003 Workshop Proceedings*, pp 1-11, Schloss Dagstuhl, Germany, December 15-17, 2003.
- [6] N. Fuhr, N. Gövert, G. Kazai and M. Lalmas. “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)*, pp 1-15, Schloss Dagstuhl, Germany, December 9-11, 2002.
- [7] Jansen, J. Bernard, A. Spink, J. Bateman, T. Saracevic. “Real Life Information Retrieval: a Study of User Queries on the Web”. In *SIGIR Forum 32 No. 1*, pp. 5-17, 1998.
- [8] M. B. Koll. “Automatic Relevance Ranking: A Searcher's Complement to Indexing”. In *Indexing, Providing Access to Information: Looking Back, Looking Ahead, Proceedings of the 25th Annual Meeting of the American Society of Indexers*, pp 55-60, Alexandria, VA, May 20-22, 1993.

TRIX 2004 – struggling with the overlap

Jaana Kekäläinen
Dept. of Information Studies
33014 University of Tampere
Finland
jaana.kekalainen@uta.fi

Marko Junkkari
Dept. of Computer Sciences
33014 University of Tampere
Finland
junken@cs.uta.fi

Paavo Arvola
Dept. of Computer Sciences
33014 University of Tampere
Finland
paavo.arvola@uta.fi

Timo Aalto
Dept. of Information Studies
33014 University of Tampere
Finland
timo.aalto@uta.fi

ABSTRACT

In this paper, we present a new XML retrieval system prototype employing structural indices and a *tf*idf* weighting modification. Our runs for INEX 2004 test a) emphasizing the *tf* part in weighting and b) allowing overlap to different degrees in run results. It seems that increasing the overlap percentage leads to a better performance. Emphasizing the *tf* part enables us to increase exhaustivity of the run results.

Categories and Subject Descriptors

H.3 [Information Storage and Retrieval]: Retrieval models, performance evaluation

General Terms

Performance, Design, Experimentation.

Keywords

XML, Information retrieval, Relevance ranking, Overlap

1. INTRODUCTION

TRIX (Tampere retrieval and indexing system for XML) is aimed for full scale XML retrieval. Extensibility and generality for heterogeneous XML collections have been the main goals in designing TRIX. We started from scratch and the first prototype was implemented during four months in the summer 2004. This prototype is able to manipulate CO queries but not CAS queries. However, with the CO approach of TRIX we achieved tolerable ranking for VCAS runs in INEX 2004.

One idea of XML is to distinguish the content (or data) element structure from stylesheet descriptions. From the perspective of information retrieval, stylesheet descriptions are typically irrelevant. However, in the INEX collection these markups are not totally separated. Moreover, some elements are irrelevant for information retrieval. Thus, we preprocessed the INEX collection so that we removed the irrelevant parts from the collection. We classified these irrelevant parts into three classes. First, there are elements which possess relevant content but the tags are irrelevant. Tags which only denote styles, such as boldface or

italic, inhere in this class. These tags were removed but the content of the elements was maintained. Second, there are elements whose content seems irrelevant but their tags are necessary in order to maintain the coherent structure of documents. For example we appraised the content of `<sgmlmath>` and `<math>` elements to inhere in this class. Third, there are elements having irrelevant content and whose tags are not necessary in structural sense. These elements, such as `<doi>` and `<en>`, were removed.

The main goal of the preprocessing of the INEX collection was to achieve a structure in which the content element has a natural interpretation. In the terminology of the present paper, the content element means an element that has own textual content. The ranking in TRIX is based on weighting the words (keys) with a *tf*idf* weighting modification, in which length normalization and *idf* are based on content elements instead of documents.

The overlap problem is an open question in XML information retrieval. On one hand, it would be ideal that the result list does not contain overlapping elements [3]. On the other hand, the metrics of INEX 2004 encourage for large overlapping among results. In this paper, we examine how the ranking of runs depends on the degree of overlap. For this, we have three degrees of overlap:

1. No overlapping is allowed. This means that any element (or document) is discarded in the ranking list if its subelement (descendant) or superelement (ancestor) appears in the result list.
2. Partial overlapping is allowed. The partial overlapping means that the immediate subelements and superelement are not allowed in the result list relating to those elements which have a higher score.
3. Full overlapping is allowed.

In this report we present the performance of two slightly different weighting schemes and three different overlapping degrees for both CO and VCAS tasks. The report is organized as follows: TRIX is described in Chapter 2, the results are given in Chapter 3, and discussion and conclusions in Chapter 4.

2. TRIX 2004

2.1 Background

The manipulation of XML documents in TRIX is based on structural indices [2]. In the XML context this way of indexing is known better as Dewey ordering [7]. To our knowledge the first proposal for manipulating hierarchical data structures using structural (or Dewey) indices is found in [5]. The idea of structural indices is that the topmost element is indexed by $\langle 1 \rangle$ and its immediate subelements by $\langle 1,1 \rangle$, $\langle 1,2 \rangle$, $\langle 1,3 \rangle$ etc. Further the immediate subelements of $\langle 1,1 \rangle$ are labeled by $\langle 1,1,1 \rangle$, $\langle 1,1,2 \rangle$, $\langle 1,1,3 \rangle$ etc. This kind of indexing enables analyzing any hierarchal data structure in a straightforward way. For example, the superelements of the element labeled by $\langle 1,3,4,2 \rangle$ are found from indices $\langle 1,3,4 \rangle$, $\langle 1,3 \rangle$ and $\langle 1 \rangle$. In turn, any subelement related to the index $\langle 1,3 \rangle$ is labeled by $\langle 1,3,\xi \rangle$ where ξ is a non-empty subscript of the index.

In TRIX we have utilized structural indices in various tasks. First, documents and elements are identified by them. Second, the structure of the inverted file for elements is based on structural indices. Third, algorithms for partial and full overlapping are designed based on them.

2.2 Weighting Function and Relevance Scoring

In TRIX the weighting of keys is based on a modification of the BM25 weighting function [1, 6].

$$w_k = \frac{kf_e}{kf_e + v * \left((1-b) + b * \frac{ef_c}{\sqrt{ef_k}} \right)} * \frac{\log\left(\frac{N}{n}\right)}{\log N}$$

where w_k is the weight assigned to a key k in element e , kf_e is the number of times k occurs in the element, ef_c is the number of all content subelements of the element e , ef_k is the number of content subelements of e containing k , n is the number of content elements containing k , N is the total number of content elements, v and b are constants for tuning the weighting. The length normalization for the element is based on the ratio of its all content subelements and content subelements containing the key.

The weights are combined to a relevance ranking score for each element by taking an average of the weights of keys – or query fragments – appearing in queries and elements. By query fragments we refer to phrases or +/- operations. Beside the average we also used a fuzzy operation called Einstein's sum [4]:

$$w_{k1,k2} = \frac{w_{k1} + w_{k2}}{1 + w_{k1} \cdot w_{k2}}$$

Unlike average this operation is associative, which means that if \oplus denotes Einstein's sum then $w_{k1} \oplus w_{k2} \oplus w_{k3} = (w_{k1} \oplus w_{k2}) \oplus w_{k3} = w_{k1} \oplus (w_{k2} \oplus w_{k3})$.

The '+' prefix in queries is used to emphasize the importance of a search key. In TRIX the weight of the key was increased by taking a square root of the original weight:

$$w_k^+ = \sqrt{w_k}$$

This works because the weights are scaled between 0 and 1. The '-' prefix in queries denotes an unwanted key. In TRIX the weight of such a key was decreased by changing the weight to its negation:

$$w_k^- = -w_k$$

TRIX does not support proximity searching. However, it is possible to demand keys to appear in the same content element by giving the keys in quotes: " k_1, \dots, k_n ". In this case the calculation of the weights is affected as follows:

$$w_{k1\dots kn} = \frac{k_{1\dots n}f_e}{k_{1\dots n}f_e + v * \left((1-b) + b * \frac{ef_c}{\sqrt{ef_{k1\dots kn}}} \right)} * \frac{\log\left(\frac{N}{n}\right)}{\log N}$$

where $k_{1\dots n}f_e = \min\{k_1f_e, k_2f_e, \dots, k_nf_e\}$.

2.3 Implementation

The TRIX is implemented in C++ for Windows/XP but the implementation has not been bound to these operating systems. In other words, the TRIX prototype could be operated in UNIX/LINUX as well. In implementing the present TRIX prototype, we have paid attention for effective manipulation of XML data structures based on structural indices. However, the efficiency has not been the main goal of TRIX, and for optimizing of the code we have not used any tricks.

The TRIX prototype has two modes: online mode and batch mode. In the online mode the user can run CO queries in the default database (XML collection). The batch mode enables running a set of CO queries. In this mode queries are saved in a text file. Running the CO queries of INEX 2004 in the batch mode takes about 40 minutes in a sample PC (Intel Pentium 4, 2.4 GHz, 512MB of RAM).

The command based user-interface of the TRIX prototype is tailored for testing various aspects XML information retrieval. This means that a query can be run with various options. For example, the user can select:

- the method (average or Einstein's sum) used in combining of weights,
- the degree of overlap (no overlapping, partial overlapping or full overlapping), and
- the values of the constants.

For example the command string

```
TRIX -e -o b=0.1 queries2004co.txt
```

means that Einstein's sum is used in combination of weights (parameter -e), full overlapping is allowed (parameter -o) and the b value is 0.1. Finally, queries2004co.txt denotes the file from which the query set, at hand, is found. Actually, there is no assumption of ordering for the parameters of a query. For example, the command string

```
TRIX -o queries2004cs.txt b=0.1 -e
```

is equivalent with the previous query.

The online mode of TRIX is chosen by the command

```
TRIX
```


After this command the user may give his/her query.

3. RESULTS

For INEX 2004 we submitted both CO and VCAS runs though our system supports only CO queries. In both cases, the title field was used in automatic query construction. Phrases marked in titles were interpreted as ‘TRIX phrases’ in queries, i.e. all the phrase components were demanded to appear in the same element. Yet, all the components were added as single keys to queries as well. In VCAS queries the structural conditions were neglected and all keys were collected into a flat query. Word form normalization for the INEX collection and queries was Porter stemming, and a stoplist of 419 words was employed.

We tested in both CO and VCAS runs the effects of a) tuning of the constant b in the weighting scheme and b) overlap in the results.

3.1 CO Runs

In the official submissions of the CO queries we tried both average and Einstein’s sum in relevance scoring. The results were so similar that we report the results based on average only. Further, in our official submissions two overlap degrees were tested: no overlapping and partial overlapping. Later on we added the full overlapping case.

Table 1. Scores and Rankings of CO runs

	B	Score	Ranking
no overlapping	0.4	0.0198	45
	0.1	0.0239	42
partial overlapping	0.4	0.0443	31
	0.1	0.0487	25
full overlapping	0.4	0.0831	11
	0.1	0.0957	10

Table 1 shows the effect of different overlaps and tuning of b to the aggregate score and rank. Decreasing b has a slight positive effect on the aggregate score and rank. When the different metrics are considered, it is obvious that small b values enhance the dimension of exhaustivity at specificity’s expense. Figures 3 and 5 in the appendix show a specificity-oriented metric, quantization s_3e_{321} , and there average precision decreases as the b decreases. Figures 4 and 6 in the appendix show an exhaustivity-oriented metric, quantization e_3s_{321} , which shows that average precision increases as b decreases.

The effect of overlap is more substantial: allowing the full overlapping changes the aggregate rank from 45th to 11th when $b=0.4$, or from 42nd to 10th when $b = 0.1$. Figure 1 illustrates the increase in the aggregate score when overlap percentage increases (compare Figures 3a and 5a, and 3b and 5b, etc. in the appendix). Whether the change in the result lists is desirable from the user’s point of view is questionable because it means returning several overlapping elements from the same document in a row.

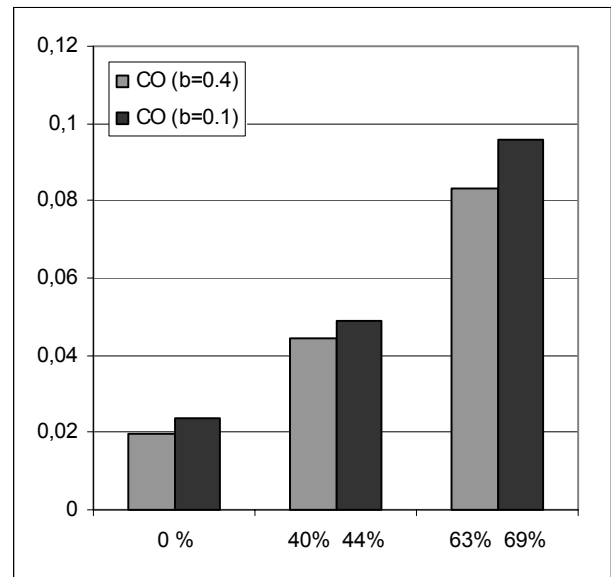


Figure 1. Scores and overlapping of CO runs

3.2 VCAS Runs

The results of the VCAS runs are very similar to CO runs. Decreasing b value gives better exhaustivity-oriented results but impairs specificity. Increasing the overlap enhances effectiveness. Both these tactics have a positive effect on the aggregate score (see Table 2).

Table 2. Scores and Rankings of VCAS runs

	b	Score	Ranking
no overlapping	0.4	0.269	30
	0.1	0.0308	30
partial overlapping	0.4	0.0384	25
	0.1	0.0421	22
full overlapping	0.4	0.0607	11
	0.1	0.0754	7

Figure 2 shows the overlap percentages for different VCAS runs. Also here the benefits of allowing the overlap are evident though not as remarkable as for CO queries.

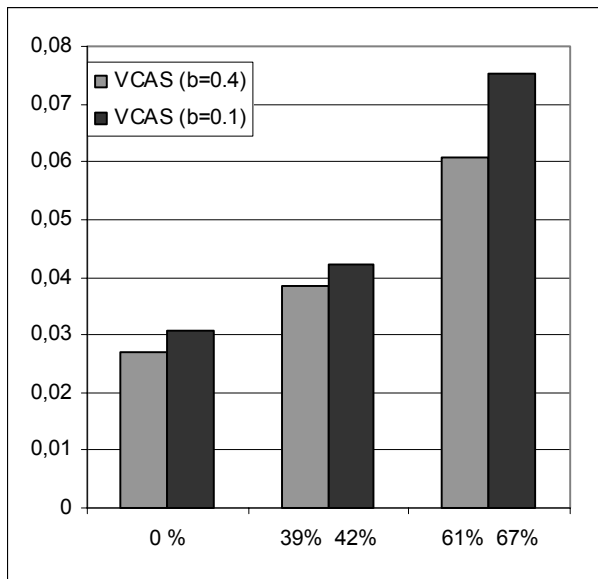


Figure 2. Scores and overlapping of VCAS runs

4. DISCUSSION AND CONCLUSIONS

TRIX is an XML retrieval system which employs a modification of *tf*idf* weighting, using the number of content subelements in element length normalization. In the present mode it only supports CO queries but we aim at introducing a query language for content and structure queries. Our original design principle was not to allow overlap in results. Because only the titles of the topics – providing a very terse description of the information need – were allowed in query construction, and we did not expand the queries, mediocre effectiveness was to be expected.

When the official results were distributed, we found out that our official submission allowing partial overlap yielded a better performance than runs with no overlap. We then tested runs with full overlap and a slightly tuned weighting scheme. These tests show that strengthening the *tf* part in our weighting scheme (by decreasing *b*) enables us to strengthen the exhaustivity in results. Further, allowing the full overlap leads to a dramatic improvement in the aggregate score and rank. In our case, the improvement in effectiveness was not necessarily an improvement from the user's point of view, because it led to a massive repetition.

Since TRIX does not support querying with structural conditions we submitted VCAS runs processed similarly as CO runs. Surprisingly our success with the VCAS task was not worse than with the CO task. However, if structural conditions are not considered when assessing the relevance, it is understandable that CO and VCAS tasks resemble each other.

Our further work with TRIX is aimed at introducing a query expansion or enhancing module. Incapability to deal with short content queries is a well-known disadvantage. Also, a CAS query language allowing also document restructuring is under construction.

5. ACKNOWLEDGMENTS

This research was supported by the Academy of Finland under grant number 52894.

6. REFERENCES

- [1] Hawking, D., Thistlewaite, P., and Craswell, P. ANU/ACSys TREC-6 experiments. In *Proc. of TREC-6*, 1998. [online, cited 21.11.2004.] Available at: <URL: <http://trec.nist.gov/pubs/trec6/papers/anu.ps>>
- [2] Junkkari, M. PSE: An object-oriented representation for modeling and managing part-of relationships. *Journal of Intelligent Information Systems*, to appear.
- [3] Kazai, G., Lalmas, M., and de Vries, A.P. The overlap problem in content-oriented XML retrieval evaluation. In *Proc. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Sheffield, UK, 2004, 72-79.
- [4] Mattila, J.K. *Sumean Logiikan Oppikirja: Johdatusta Sumean Matematiikkaan*. Art House, Helsinki, 1998.
- [5] Niemi, T. A seven-tuple representation for hierarchical data structures. *Information systems*, 8, 3 (1983), 151-157.
- [6] Robertson S.E., Walker, S., Jones, S., Hancock-Beaulieu, M.M., Gatford, M. Okapi at TREC-3. In *NIST Special Publication 500-226: Overview of the Third Text REtrieval Conference (TREC-3)*. 1994. [online, cited 21.11.2004] Available at: <URL: <http://trec.nist.gov/pubs/trec3/papers/city.ps.gz>>
- [7] Tatarinov, I., Viglas, S.D., Beyer, K., Shanmugasundaram, J., Shekita, E., and Zhang, C. Storing and querying ordered XML using a relational database system. In *Proc. of the SIGMOD Conference*, 2002, 204-215.

APPENDIX

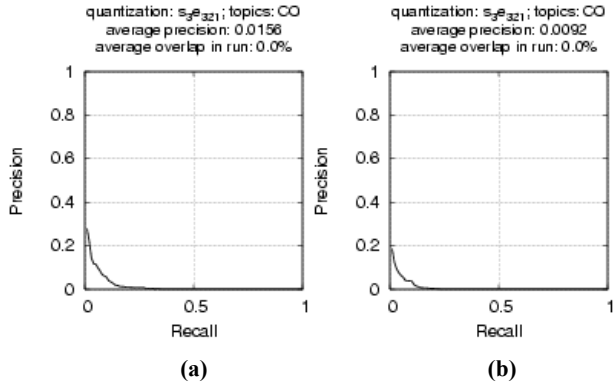


Figure 3. CO without overlap. Quantization: s_3e_{321} (a) $b = 0.4$, rank 39/70; (b) $b = 0.1$, rank 46/70

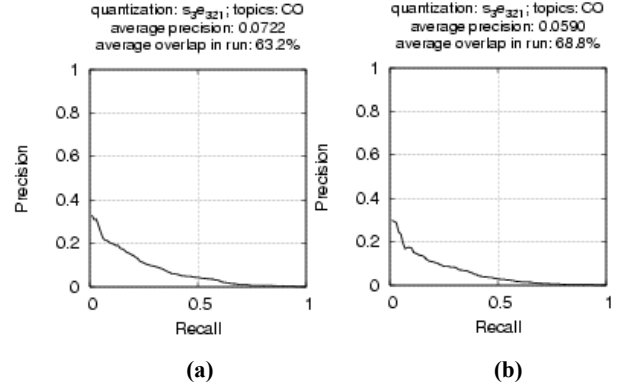


Figure 5. CO with full overlap. Quantization: s_3e_{321} (a) $b = 0.4$, rank 8/70; (b) $b = 0.1$, rank 12/70

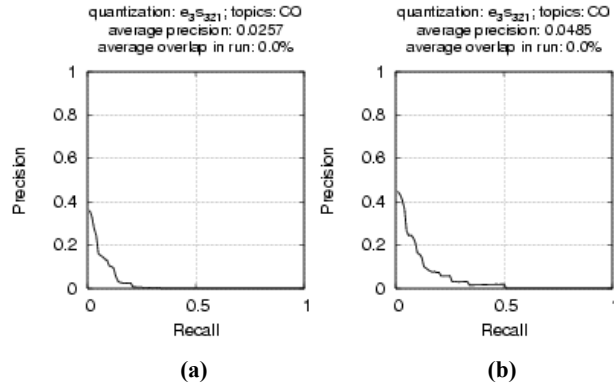


Figure 4. CO without overlap. Quantization: e_3s_{321} (a) $b = 0.4$, rank 45/70; (b) $b = 0.1$, rank 39/70

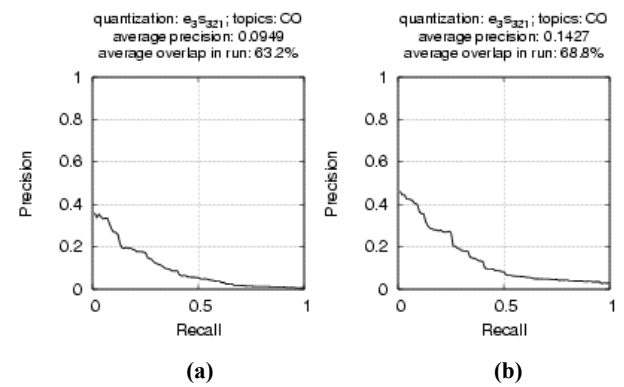


Figure 6. CO with full overlap. Quantization: e_3s_{321} (a) $b = 0.4$, rank 17/70; (b) $b = 0.1$, rank 11/70

Merging XML Indices

Giambattista Amati, Claudio Carpineto and Giovanni Romano
Fondazione Ugo Bordoni, Via Baldassarre Castiglione 59, 00142, Rome, Italy
{gba,carpinet,romano}@fub.it

ABSTRACT

Using separate indices for each element and merging their results has proven to be a feasible way of performing XML element retrieval; however, there has been little work on evaluating how the main method parameters affect the results. We study the effect of using different weighting models for computing rankings at the single index level and using different merging techniques for combining such rankings. Our main findings are that (i) there are large variations on retrieval effectiveness when choosing different techniques for weighting and merging, with performance gains up to 102%, and (ii) although there does not seem to be any best weighting model, some merging schemes perform clearly better than others.

1. INTRODUCTION

We focus on the Content Only (CO) task and try to extend information retrieval (IR) techniques to deal with XML documents. As each XML document is formed by several nested elements and the goal is to retrieve the most relevant elements, IR ranking models must be expanded with element level statistics. However, at INEX 2003, Mass and Mandelbrod [7] showed that, for XML documents, the use of classical IR statistics involving element and term frequencies is not straightforward and may easily lead to inconsistencies and errors, due to the nesting of elements.

To overcome this problem, one can compute weights at the most specific level and propagate such weights upwards using augmentation factors [5]. Another approach, which does not rely on user parameters, is to use a separate index for each type of elements and compute rankings at the single index level. Such rankings are then merged to return a combined result [7].

In this paper we aim at experimenting with the latter approach, extending previous work in two directions. Our goal is to study whether the choice of the weighting model and the merging technique affect the retrieval performance of

XML indices in the INEX environment, and to evaluate relative merits and drawbacks of different parameter choices.

We consider three weighting models with a different theoretical background that have proved their effectiveness on a number of tasks and collections. The three models are deviation from randomness [3], Okapi [11], and statistical language modeling [13].

The merging problem is tackled by combining the relevance scores associated with each index through different normalization techniques. We consider five schemes; namely, normalization by query score, maximum score, standard norm, sum norm, and Z-score norm.

In the following we first present the weighting models and the normalization schemes. Then we describe the experimental setting and discuss the results. Finally, we provide some conclusions.

2. WEIGHTING MODELS

For the ease of clarity and comparison, the document ranking produced by each weighting model is represented using the same general expression, namely as the product of a document-based term weight by a query-based term weight:

$$sim(q, d) = \sum_{t \in q \wedge d} w_{t,d} \cdot w_{t,q}$$

Before giving the expressions for $w_{t,d}$ and $w_{t,q}$ for each weighting model, we report the complete list of variables that will be used:

f_t	the number of occurrences of term t in the collection
$f_{t,d}$	the number of occurrences of term t in document d
$f_{t,q}$	the number of occurrences of term t in query q
n_t	the number of documents in which term t occurs
D	the number of documents in the collection
T	the number of terms in the collection
λ_t	the ratio between f_t and T
l_d	the length of document d
l_q	the length of query q
$avr\ l_d$	the average length of documents in the collection

2.1 Okapi

To describe Okapi, we use the expression given in [11]. This formula has been used by most participants in TREC and CLEF over the last years.

$$w_{t,d} = \frac{(k_1 + 1) \cdot f_{t,d}}{k_1 \cdot \left[(1 - b) + b \frac{l_d}{avr_l_d} \right] + f_{t,d}}$$

$$w_{t,q} = \frac{(k_3 + 1) \cdot f_{t,q}}{k_3 + f_{t,q}} \cdot \log_2 \frac{D - n_t + 0.5}{n_t + 0.5}$$

2.2 Statistical Language Modeling (SLM)

The statistical language modeling approach has been proposed in several papers, with many variants (e.g., [6], [9]). Here we use the expression given in [13], with Dirichlet smoothing.

$$w_{t,d} = \log_2 \frac{f_{t,d} + \frac{\mu \lambda_t}{l_d + \mu}}{l_d + \mu} - \log_2 \frac{\mu}{l_d + \mu} - \log_2 \lambda_t + \frac{l_q}{|q \wedge d|} \cdot \log_2 \frac{\mu}{l_d + \mu}$$

$$w_{t,q} = f_{t,q}$$

2.3 Deviation From Randomness (DFR)

Deviation from randomness has been successfully used at TREC, for the Web track [1], and CLEF, for the monolingual tasks [2]. It is best described in [3].

$$w_{t,d} = (\log_2(1 + \lambda_t) + f_{t,d}^* \cdot \log_2 \frac{1 + \lambda_t}{\lambda_t}) \cdot \frac{f_t + 1}{n_t \cdot (f_{t,d}^* + 1)}$$

$$w_{t,q} = f_{t,q}$$

with

$$f_{t,d}^* = f_{t,d} \cdot \log_2 \left(1 + \frac{c \cdot avr_l_d}{l_d} \right)$$

3. MERGING METHODS

Most of IR work on method combination has focused on merging multiple rankings with overlapping documents, whereas combining disjoint rankings has not received much attention. If training information is available, one can learn cut-off values for each ranking [12] or give a value to each index [4]. As in this case we did not have access to prior data (this is our first participation in INEX), we use combination techniques that do not require such data.

One simple approach would be to combine the original scores into a large ranked list, without modifying the scores. However, such an approach would not work, due to the different scales of the scores yielded by each index. In fact, the relevance scores used in our experiments are not probabilities

and the statistics on which they are based are relative to indices of varying size. Thus, the scores need to be normalized, and the merging problem essentially becomes a normalization problem.

Normalization can be done in different ways (see for instance [8]). We test five approaches, which feature different properties in terms of shift and scale invariance and outlier tolerance. Such approaches are detailed in the following.

3.1 Q

The raw scores of the elements retrieved in response to query Q by index i are divided by $sim(q, q)$, which is the score of the query itself (as if it were a document in the index) according to the weighting model of index i . This technique, denoted here by Q , has been used in [7].

3.2 Max

The raw scores are divided by the maximum score in the corresponding index. Note that each index will produce one element with normalized score = 1. In the combined ranking, these topmost elements are ordered according to their original value. This normalization scheme will be referred to as *Max*.

3.3 MinMax

This scheme consists of shifting the minimum raw score to zero and scaling the maximum to one, i.e.

$$\frac{score - minimum}{maximum - minimum}$$

This scheme will be denoted by *MinMax*.

3.4 Sum

A normalized score is obtained by shifting the minimum raw score to zero and the sample sum to one; i.e.,

$$\frac{score - minimum}{\sum_N scores - N \cdot minimum}$$

This scheme will be denoted by *Sum*.

3.5 Z-score

This is the classical *standard score*, denoted *Z-score*. It is derived by subtracting the sample mean from raw scores and then dividing the difference by the sample standard deviation, i.e.,

$$\frac{score - mean}{\sigma}$$

4. EXPERIMENTAL SETTING

As also pointed out in [7], the great majority of highly relevant elements are taken from the set: {article, bdy, abs, sec, ss1, ss2, p, ip1}, because they represent more meaningful results for a query. We intended to build a separate index for each of these elements; however, a bug in the program for building the indices of more specific elements (i.e., paragraphs) and a tight schedule prevented us from doing so. In the experiments reported here we use only 5 types of

Table 1: Average precision (strict quantization) by weighting method and by normalization scheme.

	DFR	Okapi	SLM
Q	0.0695	0.0769	0.0927
Max	0.0903	0.0963	0.0931
MinMax	0.0853	0.0911	0.0848
Sum	0.0713	0.0806	0.0492
Z-score	0.1018	0.0987	0.0938

elements: {article, abs, sec, ss1, ss2}. Even the runs actually submitted by us to INEX for evaluation had the same limitation.

Each index was built as follows. We identified the individual words occurring in the elements of interest, ignoring punctuation and case; thus, a strict single-word indexing was used. The system then performed word stopping and word stemming, using Porter algorithm [10].

At run time, we ran each INEX 2004 CO topic against all 5 indices and computed the ranking associated with each index. Only the title topic statement was considered. For each query and for each index, we then computed three rankings, one for each weighting model. The choice of the parameters involved in the weighting models was as follows.

DFR	$c = 2$
Okapi	$k_1 = 1.2, k_3 = 1000, b = 0.75$
SLM	$\mu = 1000$

Then, for each query, we merged the index level rankings of each weighting model using the five normalization schemes described above.

5. RESULTS

We computed in all 15 rankings, i.e., three weighting models times five normalization schemes. In order to evaluate the retrieval effectiveness, we focus on strict relevance; i.e., on highly exhaustive and specific (E3S3) elements. This choice was partly motivated by the importance of this task for an XML information retrieval system, partly by the observation that a E3S3 relevance judgement may better reflect the will of the assessor rather than the rules enforced by the evaluation system, which were found to produce a proliferation of small irrelevant elements labeled as (partially) relevant.

The results are shown in Table 1; performance was measured using average precision averaged on the 25 topics with nonempty E3S3 elements.

Before discussing the results, we would like to make one general comment about the absolute value of the strict quantization figures at INEX 2004. Our impression is that the results have been penalized by a large number of elements that have probably been mistakenly labeled as strictly relevant (E3S3) for some topics. For instance, there are as many as 288 E3S3 "it" elements and 68 E3S3 "tmath" elements associated with one single topic. Also, the 55% of all E3S3 elements (i.e., 1429 out of 2589 elements) is associated with

only two topics. Even though evaluation of precision may not be so much affected by these spurious elements, because they will probably not be highly ranked by the retrieval systems, this will definitely downweight the recall part of the evaluation measures.

Turning to the relative behaviour of the different methods tested in the experiments, the results in Table 1 show that there was a huge variability in retrieval effectiveness. The worst performance was obtained by the pair SLM/Sum, with an average precision of 0.0492; the best performance by the pair DFR/Z-score, with an average precision of 0.1018 (+102%). Incidentally, the submitted runs were obtained using DFR with Q, with official scores very similar to that reported here (0.0695).

If we look at the behaviour of the weighting models when the normalization scheme is kept constant, we see that no weighting model clearly won. DFR achieved the best results for Z-score, Okapi for Max, MinMax, and Sum, and SLM for Q. In most cases (i.e., for Max, MinMax, and Z-score), the results were comparable.

The results for the normalization schemes reveal a more interesting pattern. The most important finding is that Z-score achieved the best performance for each weighting model, with notable performance improvements over the results obtained by the other normalization schemes using the same weighting model. In particular, for DFR, the average precision grows from 0.0695 with Q to 0.1018 with Z-score, and for Okapi, it grows from 0.0769 with Q to 0.0987 with Z-score.

The results in Table 1 also show that Max was consistently ranked as the second best normalization scheme for each weighting model, although with more comparable performance improvements than Z-score. The other three normalization schemes showed a mixed behaviour.

The results presented so far were obtained considering the full set of relevance judgements. As our system only deal with five types of elements, all the other relevant elements cannot be actually retrieved. So it may be interesting to see what happens if we remove from the relevance set all the elements other than those used by the system. This should give an idea about the results that this method might obtain if we expanded the number of indices to include at least the body and paragraph elements. On the other hand, it must be considered that not all indices are alike; chances are that there are proportionally fewer small elements (e.g., paragraphs) that are relevant, so it may be more difficult to find them.

If we consider only the elements dealt with by our system, we get 614 E3S3 elements (rather than 2589). In Table 2, we show the retrieval effectiveness of the weighting/merging methods relative to such a restricted set of E3S3 elements, in which only the elements {article, abs, sec, ss1, ss2} have been kept.

If we compare the results in Table 1 with those in Table 2, we see that passing from unrestricted to restricted relevance judgements roughly doubles the retrieval performance. The

Table 2: Average precision (strict quantization) by weighting method and by normalization scheme on the restricted relevances.

	DFR	Okapi	SLM
Q	0.1352	0.1500	0.1673
Max	0.1716	0.1791	0.1651
MinMax	0.1594	0.1654	0.1479
Sum	0.1520	0.1517	0.0911
Z-score	0.2080	0.2033	0.1807

improvement might seem smaller than one might expect by judging from the decreasing in the number of relevant elements. Consider that the system retrieves the same elements in the same order in both situations, so the change in average precision only depends on the different number of relevant elements per query. As this number roughly reduces to one fourth (from 2589 to 614), it may be somewhat surprising to see that the average precision just doubled, rather than becoming four times greater. In fact, we checked that most of the relevant elements other than those considered by our system are concentrated in a very small number of topics. For instance, 323 out of the 691 E3S3 paragraphs are associated with just one query.

The results in Table 2 confirm the main findings obtained for the unrestricted relevances. The main differences are that DFR achieved the best performance for two normalization schemes rather than for one and that the performance variations were slightly less marked.

On the whole, our results suggest that while the different weighting models achieved comparable retrieval performance the normalization schemes differed considerably, with Z-score showing a superior performance. This raises the question of why Z-score worked better. One explanation is that Z-score is based on aggregate statistics, which are more robust (e.g., with respect to outliers). However, this is not completely satisfying, because Sum is also based on aggregate statistics and it did not score so well. A better understanding of why some methods perform better than others would probably require a deeper analysis of the ranking data. For instance, as standard scores are especially appropriate for data that are normally distributed, one can hypothesize that the ranking data follow this distribution.

Our results also suggest that certain combinations of weighting and merging work particularly well (e.g., DFR and Z-score) or particularly badly (e.g., SLM and Sum); an analysis of the mutual relationships between weighting models and merging schemes is another issue that deserves more investigation.

6. CONCLUSIONS

The main indication of our experiments is that there is much scope for improving the performance of XML retrieval based on separate indices. We showed that an appropriate choice of the weighting model and normalization scheme may greatly improve the retrieval effectiveness of this technique.

One direction for future work is to use more queries and evaluation measures, incorporating past statistics about distribution of relevant elements across element types to improve combination of results. As one shortcoming of using separate indices is that the relationships between the elements in different indices are not taken into account, future work will also consider how to discriminate between nested retrieval results

7. REFERENCES

- [1] G. Amati, C. Carpineto, and G. Romano. FUB at TREC-10 Web Track: A Probabilistic Framework for Topic Relevance Term Weighting. In *Proceedings of the 10th Text REtrieval Conference (TREC-10)*, NIST Special Publication 500-250, pages 182–191, Gaithersburg, MD, USA, 2001.
- [2] G. Amati, C. Carpineto, and G. Romano. Comparing weighting models for monolingual information retrieval. In *Working Notes for the CLEF 2003 Workshop*, pages 169–178, Trondheim, Norway, 2003.
- [3] G. Amati and C. J. van Rijsbergen. Probabilistic models of information retrieval based on measuring divergence from randomness. *ACM Transactions on Information Systems*, 20(4):357–389, 2002.
- [4] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington, USA, 1995.
- [5] N. Fuhr and K. GrossJohann. XIRQL: A query language for information retrieval in XML documents. In *Proceedings of SIGIR 2001*, pages 172–180, New Orleans, LA, USA, 2001.
- [6] D. Hiemstra and W. Kraaij. Twenty-one at TREC-7: Ad hoc and cross-language track. In *Proceedings of the 7th Text REtrieval Conference (TREC-7)*, NIST Special Publication 500-242, pages 227–238, Gaithersburg, MD, USA, 1998.
- [7] Y. Mass and M. Mandelbrod. Retrieving the most relevant XML components. In *Proceedings of the INEX 2003 Workshop*, pages 53–58, Schloss Dagstuhl, Germany, 2003.
- [8] M. Montague and J. Aslam. Relevance score normalization for metasearch. In *Proceedings of the 10th International ACM Conference on Information and Knowledge Management*, pages 427–433, Atlanta, Georgia, USA, 2001.
- [9] J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, 1998.
- [10] M. F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

- [11] S. E. Robertson, S. Walker, and M. M. Beaulieu. Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC, and Interactive track. In *Proceedings of the 7th Text REtrieval Conference (TREC-7), NIST Special Publication 500-242*, pages 253–264, Gaithersburg, MD, USA, 1998.
- [12] E. Voorhees, N. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Reasearch and Development in Information Retrieval*, pages 172–179, Seattle, Washington, USA, 1995.
- [13] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342, New Orleans, LA, USA, 2001.

The Utrecht Blend: Basic Ingredients for an XML Retrieval System

Roelof van Zwol

Centre for Content and
Knowledge Engineering
Utrecht University
Utrecht, the Netherlands

roelof@cs.uu.nl

Frans Wiering

Centre for Content and
Knowledge Engineering
Utrecht University
Utrecht, the Netherlands

frans.wiering@cs.uu.nl

Virginia Dignum

Centre for Content and
Knowledge Engineering
Utrecht University
Utrecht, the Netherlands

virginia@cs.uu.nl

ABSTRACT

Exploiting the structure of a document allows for more powerful information retrieval techniques. In this article a basic approach is discussed for the retrieval of XML document fragments. Based on a vector-space model for text retrieval we aim at investigating various strategies that influence the retrieval performance of an XML-based IR system.

The first extension of the system uses a schema-based approach that takes into account that authors tag their text to emphasise on particular pieces of content that are of importance. Based on the schema used by the document collection, the system can easily derive the children of mixed content nodes. Our hypothesis is that those child nodes are more important than other nodes.

The second approach discussed here is based on a horizontal fragmentation of the inverse document frequencies, used by the vector space model. The underlying assumption states that the distribution of terms is related to the semantical structure of the document. However, we observed that the IEEE collection is not a good example of semantic tagging.

The third approach investigates how the performance of the retrieval system can improve for the 'Content Only' task by using a set of a-priori defined cut-off nodes that define 'logical' document fragments that are of interest to a user.

1. INTRODUCTION

The upcoming XML standard as a publishing format provides many new challenges. One of these challenges, the scope of INEX [2], is the retrieval of structured documents. This requires new techniques that extend current developments in text retrieval. Not only should an XML retrieval

system be equipped with an adequate text retrieval strategy, it is also required that the system is capable to take the document structure into account during the retrieval process.

The structure of the XML document is not only used to refine the query formulation process, it also allows to retrieve more accurate the relevant pieces of information that a user is interested in. For the ad hoc track of INEX, two tasks are defined that take these aspects into account: the *Content Only* (CO) task and the '*Vague Content and Structure* (VCAS) task [4]. The aim of both tasks is to retrieve relevant document fragments. The difference lays in the query formulation. The CO task uses only a keyword specification, as commonly used for text retrieval and the well-known Internet search engines. The VCAS task, however, also takes the document structure into account for the query formulation, using the NEXI specification.

The challenge is thus to build the best content-based XML retrieval system that allows for the retrieval of relevant text fragments, while taking the structure of the XML documents into account. Our personal aim is more modest, since we are primarily interested in the effect of our hypothesis on the retrieval performance of an XML retrieval system. Therefore we have built a retrieval system, that is based on the vector space model for text retrieval and use a strict interpretation of the structural constraints, formerly referred to as the *strict content and structure* (SCAS) task.

We have three hypothesis that we want to put to the test. First of all our aim is to investigate whether the retrieval performance of our default XML retrieval system can be improved by taking into account that the author uses markup (structure) to emphasise on particular pieces of text that are of extra importance, i.e. bold/italic text, itemised lists, or enumerations. Focusing on the XML structure, examples of these text fragments are typically found within *mixed-content* nodes. The content model of a mixed-content node contains a mixture of text and child-elements. Using the DTD or XML-schema definition the content type of nodes can easily be determined. In this article we refer to this as the *schema-based* run.

Another hypothesis that we want to investigate here, takes

into account that some terms will occur more often within certain XML document fragments, than in other document fragments. Adjusting the term weights taking this distribution into account will increase the performance of the ranking of the retrieval strategy. This hypothesis has already been tested successfully in the context of XML and semantical schemas [9]. The vector space model consists of two components: a document statistic, i.e. the term frequency (tf), and a collection statistic, i.e. the inverse document frequency (idf). These two statistics are calculated for each term in the document collection. However, the inverse document frequencies are no longer calculated over the entire document, but for small text fragments. Assume now that some terms occur less frequently in abstract, than in other parts of the document. As a result the idf, and thus the term weight, of those terms is valued relatively low compared to other terms in the abstract. Using a fragmented document frequency, where the idf is calculated per XML element name corrects this problem. Our experience is that for semantically tagged XML documents an increase in retrieval performance can be achieved, when the query consists of two or more query terms [9]. We refer to this strategy as the *fdf* run.

The third hypothesis focuses on the CO task. For the CO task it is not specified in the query, which document fragments should be returned by the system. Returning entire documents as the result of a query will result in a low performance according to the specificity quantisation [3], since it is likely that only small portions of the XML document will contain relevant information. To deal with this we have defined a cutoff node set, that consists of XML elements that provide a partial logical view on the XML document. When retrieving XML document fragments this node set is used to return smaller fragments, that have a higher specificity of the content in relation to the query terms. We refer to this strategy as the *cutoff* run.

1.1 Organisation

In the remainder of this article we first discuss the approach used to index the XML collection in Section 2. In Section 3 the different retrieval strategies for querying XML documents is discussed for the different runs that we have submitted for INEX 2004. The results of our system are presented in Section 4, together with the unofficial runs that we computed with improved performance of the vector space model. Finally we come to our conclusions in Section 5.

2. INDEXING THE XML COLLECTION

To index the IEEE XML document collection the XML structure of each document is analysed and a text retrieval strategy is implemented. In Section 2.1 the indexing of the index structure is discussed, while in Section 2.2 the text retrieval component is described.

2.1 Processing XML Structures

To index the XML collection the structure of each document is analysed as follows. The nodes are numbered using the method described in Table 1. This resembles an approach adopted by others [5], however we have chosen not to number the individual terms within a text fragment, but to refer to a text fragment as a whole.

<ElementA> ¹
TextFragmentA ²
<ElementB> ³
TextFragmentB ⁴
</ElementB> ⁵
<ElementC> ⁶
TextFragmentC ⁷
</ElementC> ⁸
<ElementB/> ⁹
TextFragmentD ¹⁰
</ElementA> ¹¹

Table 1: XML example. illustrating the numbering of nodes

Furthermore we keep track of parent-child relations for each node. All node information is stored in the **Element** table, as shown in Figure 2. This table contains the following information about element nodes: A unique id, the element name, a reference to its parent, a pointer to the document containing the element, and the unique path leading to the element node. Finally, for each element node the start and end positions are stored, as explained above.

Whenever the indexer encounters a text fragment, a new id is generated and stored in the table **TextFragment**. A reference to the parent node, its position in the document, the number of terms, i.e. the length, and a pointer to the document URI is stored. The text fragment is then handed to the text indexer.

Document

id	uri
----	-----

Element

id	name	parent	document	path	start	end
----	------	--------	----------	------	-------	-----

Textfragment

id	parent	position	length	document
----	--------	----------	--------	----------

Term

content	fragment	tf	tfidf
---------	----------	----	-------

Table 2: Internal data structure

2.2 Processing Text Fragments

The text retrieval component of our indexing system is based on vector space model [1]. This component analyses the rather small text fragments according to the following steps:

- **pre-processing.** A number of basic text operations are called during the pre-processing step. Among these are lexical cleaning, stop word removal and stemming [1].
- **indexing.** Using a bag of terms approach the frequencies of the terms occurring in the text fragment are calculated. After processing a text fragment, all the terms are stored in the **Term** table. For each term, its content, a reference to the corresponding text fragment and the term frequency is stored in the database.
- **post-processing.** Once all documents have been indexed the collection statistics are calculated. For each unique term in the collection the inverse document fre-

quency is calculated as:

$$idf(t) = \log\left(\frac{N}{n(t)}\right), \quad (1)$$

with N being the total number of unique terms, and $n(t)$ the number of text fragments in which term t occurs.

Later on, we also used a normalised tf factor [7]. The ntf factor reduces the range of the contributions from the term frequency of a term. This is done by compressing the range of the possible tf factor values. The ntf factor is used with the belief that mere presence of a term in a text should have a default weight. Additional occurrences of a term could increase the weight of the term to some maximum value. To compute this factor we used:

$$ntf(t) = 0.5 + 0.5 * \frac{tf(t)}{\max tf(t)} \quad (2)$$

$tf(t)$ contains the raw term frequency for the term, while $\max tf(t)$ provides the maximum term frequency found in that text fragment.

The tfidf for each term in **Term** is then calculated as:

$$tfidf(t) = \frac{(tf(t) * idf(t))}{l} \quad (3)$$

Where l is the length of the text fragment. NB. this is not a standard way to normalise the term weights for the length of the text fragments.

3. QUERYING THE XML COLLECTION

For INEX we submitted six runs, as discussed below. They all use the same vector space model, with the exception of the fdf runs. Furthermore, we believe that this implementation of the vector-space model leaves plenty of room for improvement. When discussing the results, we will show some simple modifications that improve the retrieval performance of our system. Our interest in this experiment focuses mainly on the effect of using different XML-based mechanisms for calculating the relevances of the document fragments retrieved by our system. The following official runs were computed for the INEX 2004 topic set:

3.1 Content and Structured XML retrieval

The so called vague content and structure (VCAS) topics are defined using the NEXI specification [8]. Our system implements the NEXI grammar for these types of topics and evaluates the NEXI queries by following the path expressions and narrowing down the possible set of results. In fact our system enforces that the path constraints defined by the topic are computed in a strict fashion, according to the SCAS specification. We computed the following three runs for the VCAS ad hoc task:

- **33-VCAS-default.** Our default approach to compute a ranking of the retrieved documents simply determines a set of possible document fragments for the first structural constraint, and assigns a textual relevance of ‘0’ to them. If a filter clause is available, this set is narrowed down, according to the conditions defined in the filter. If an *about*-clause is defined within

that filter, a relevance ranking of the document fragments is obtained by the system. This basic approach is followed for all VCAS runs submitted. The variance between the runs is determined by the implementation of the *about*-clause.

Consider for example the following NEXI-query, presented in Table 3.

```
//article[about(./abs,classification)]//sec[about(.,
experiment compare)]
```

Table 3: NEXI example: INEX 2004, topic 132.

During the first step a set of **article**-fragments is retrieved, having a relevance score of ‘0’. The next step is to evaluate the *about*-filter, narrowing down the set of articles to those containing an **abstract**, which contains the word ‘*classification*’. The relevances computed by the about function are then summed and associated with the corresponding **article**-fragments. For this set, the second path-constraint is computed, which in this case results in a set of **sec**-nodes, which inherit the relevances computed for the parent **article** nodes. Again the about-filter is evaluated and the relevances are added to the existing relevance scores of the retrieved **sec** nodes.

For the default run the relevances for the document fragment are simply calculated by filtering all the relevant terms from the **TERM** table, using only the *positive* query terms. The relevance for each document fragment, defined in the offset of the about clause, is then calculated by summing over the terms of the text fragments that are contained within the start- and end position of the document fragment.

- **33-VCAS-schema.** The structural constraints for this run are computed similar to the default run. However the about function uses a weighing function, that increases the weight of those nodes which are considered of more importance.

The underlying hypothesis is that authors writing text use markup to emphasise on particular pieces of content that they find of more importance. Simple examples are those text fragments containing bold and italic text. A reader’s attention is automatically drawn whenever a bold or italic text fragment is seen. In XML, this markup is typically found within *mixed-content* nodes. Mixed content nodes are nodes that allow both text fragments and additional markup to be used in a mixed context. In our case, we are interested in the set of child nodes found within such *mixed-content* nodes. Using the DTD, or XML-schema definition this node set can be easily computed.

To compute the relevances of the XML document fragments the system first has to derive the set with text fragments containing relevant terms. If one or more ancestor nodes are contained in the set with mixed-content nodes a multiplication factor, i.e. 2, 4, 8, or ..., is added to the weight of that text fragment, depending on the number of mixed-content nodes that are found. Next, the relevance for each document fragment is calculated by summing over the terms of the

text fragments that are contained within the start- and end position of the document fragment.

- **33-VCAS-fdf.** This run uses an alternative way of calculating the term weights. The vector space model uses a combination of two statistics to calculate the term weights, i.e. the term frequencies and the inverse document frequencies. The inverse document frequency is a collection measure, that determines how frequently a term occurs in different documents of the collection. For the 'fragmented document fragments'-run (fdf) we have used a fragmented version of the inverse document frequencies (ifdf).

The underlying assumption for this fragmentation is that if the XML structure of the document is not merely based on presentation, but defines a semantic structure for the content contained in the document, it is likely that some terms, associated with the semantic structure will appear more often in certain document fragments than other terms.

For example, in text fragments discussing cultural information about a destination, the term '*church*' is more likely to appear, than in text fragments that discuss sports activities¹. Consider now the following information request: 'Find information about basketball clinics in former churches', the term church is an important query term in this search, however the *idf* for the query term '*church*' will be relatively low if the document collection contains both cultural- and sports descriptions of destinations. We have found that the retrieval performance improves significantly [9], when using the fdf approach. The retrieval strategy, based on the ifdf, is capable of ranking the relevant documents higher in the ranking, if the query consists of two or more query terms. In fact, increasing the amount of query terms will result in a higher retrieval performance.

3.2 Content Only XML retrieval

For the CO task we have defined four runs.

- **33-CO-default.** The content only runs are mainly driven by the text retrieval component. The positive query terms defined for each content only topic are used to find relevant text fragments. The term weights found in each text fragment are summed over the corresponding parent node of each text fragment.

In the next step the result set is grouped and summed per document. As a result the smallest common document fragment that can be retrieved for each document is returned as the result of a query. This approach ensures that no redundancy is possible between the document fragments retrieved by the system.

This approach has two advantages: no redundancy in the retrieved document fragments, and the retrieved fragments should score high on the exhaustiveness measure. This also introduces the drawback of this approach: together with the relevant information a lot

¹This example is based on the Lonely Planet collection, where the tagging of content is semantically organised[9].

of 'garbage' is retrieved, resulting in poor performance from a specificity point of view.

- **33-CO-schema.** This run is a combination of runs 33-CO-default and 33-VCAS-schema. It uses the multiplication scheme for the children of the mixed-content nodes, and the combinational logic as defined for the default approach described above. In this way, for each document the smallest document fragment is returned that contains all relevant text fragments.
- **33-CO-cutoff.** From a user point of view not all document fragments that can be retrieved are logical units. To facilitate this, we have defined a set of nodes that provide the users logical document fragments. The aim here is to find a balance between the exhaustiveness and specificity measures. For the IEEE collection we have defined a cutoff-node set containing five nodes: *fm, abs, sec, bib, article*. The article element forms the root node of many documents and should always be there, to prevent losing documents from the result set. After retrieving the relevant text fragments, the parent nodes are retrieved and (child) results merged into larger document fragments, until a node is found that is contained in the set with cutoff-nodes.
- **33-CO-fdf²** This run is also a combination of two other runs: 33-VCAS-fdf, and 33-CO-default. Instead of the default tfidf weights this run uses the ttfidf index, as explained in Section 3.1

4. RESULTS

In this section we will first present the results CO task and then the results for the VCAS task. All plots and measures were calculated using the on-line evaluation tool [6].

4.1 CO task

We first discuss the results of the official run for the CO task in Section 4.1.1. To improve on the performance for the CO task we need a better retrieval strategy for the text retrieval component. In Section 4.1.2 we investigate the effect of minor modifications in the vector space model on the retrieval performance for the CO task. Finally we compare and discuss the performance for all CO runs in Section 4.1.3.

4.1.1 Official runs

Figure 1 gives an overview of the performance of our CO runs. The *CO-default*-run performed best when evaluated using the strict quantisation measure. Slightly better performed the run *CO-schema*, while using the *e3_s32* quantisation, which illustrates that this approach is best used, when searching for exhaustive document fragments. On the other hand, the *CO-cutoff*-run performed best for the *s3_e32* quantisation measure. This was expected, since the aim of this approach was to return smaller logical document fragments, that would score better on the specificity scale.

These aspects are better illustrated in Figure 2. The average over all RP measures is showed in the top-left corner. On average, the best performance with the official runs

²For the official INEX runs, this approach is left out, since only six runs per participant were permitted.

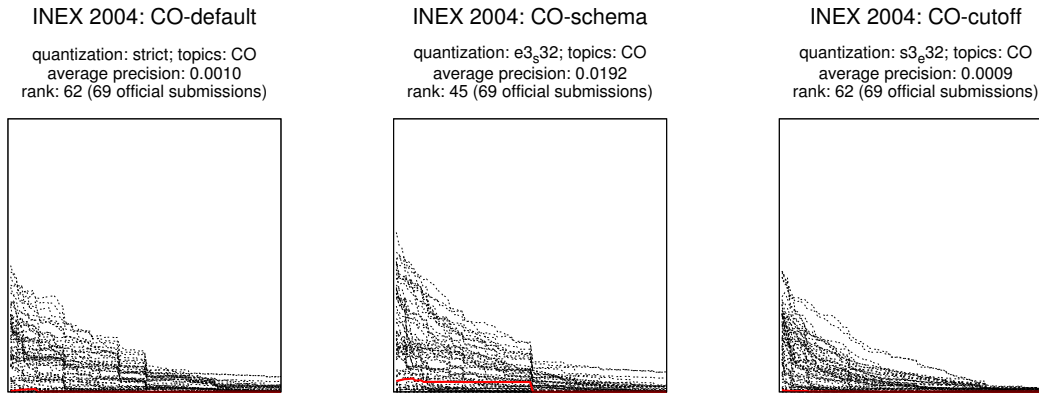


Figure 1: Official runs for the CO task - best performances

was obtained with *CO-schema*, while the *CO-cutoff*-run performed worst. Surprisingly however, the run *CO-cutoff* performed best when looking at the expected ratio of relevance (bottom-right) for the generalised recall, and slightly better when evaluation is based on the specificity quantisation. The top-right graph shows that for the CO task, it makes sense to include the markup added by the author to emphasise certain terms in the text into the ranking process.

4.1.2 Comparing the variation in the vector-space models

Due to the time constraints of INEX, our vector space model is not as sophisticated as we desired. The vector space model used for the official runs, uses a naive approach to take the length of the text fragment into account, i.e. its simply divides the term weights by the length of the text fragment. To see what the effect of the text retrieval component of our system is on this, we have also constructed a set of runs where the vector space model is not taking the length of the text fragment into account, and a set where the vector space is extended with normalisation of the term frequencies. Normalisation of the term frequencies is explained in Section 2.2. In Figure 3 two graphs are shown. The left graph computes the recall-precision based on the e3_s32 quantisation. This stresses the differences between the three variations of the vector space that are computed for the default runs, i.e.. *CO-default* (the official run), *CO-default-tfidf* (not using length normalisation), and *CO-default-ntf* (the variant that is based on normalised term frequencies). Using normalised term frequencies improves the retrieval performance at the lower recall levels (0.0 - 0.1). Not taking the length of the text fragments into account also provides a significant improvement at the recall levels 0.1 - 0.5.

The graph on the righthand-side plots the expected ratio of relevance for the generalised recall. It shows that on average the ntf-approach has the best performance, but the differences are only marginal.

4.1.3 Overall comparison

In Figure 4, we have computed the graphs for all nine variations of the CO runs. The average over all RP measures show that *CO-schema* performs best at the lower recall lev-

els (0.0 - 0.1), and that the ntf-runs have a positive effect on the precision at th recall levels 0.1-0.5. Looking at the generalised recall, it is obvious that the cutoff-runs perform significantly better than the others. As expected the cutoff run also performs better, when looking at the s3_s32 specificity quantisation. But when the focus is more on the exhaustive quantisation (e3_s32) measure, the schema-based (mixed-content) runs perform better.

4.2 VCAS task

We first discuss the results of the official run for the VCAS task in Section 4.2.1. To improve on the performance for the VCAS task we feel that we need a better retrieval strategy for the text retrieval component. In Section 4.2.2 we investigate the effect of minor modifications in the vector space model on the retrieval performance for the VCAS task.

4.2.1 Official runs

Figure 5 gives an overview of the performance of our VCAS runs. The *VCAS-default*-run performed best when evaluated using the s3e32 quantisation measure. Not surprising, since the implementation of our system uses the strict content and structure approach. The same is true for the *VCAS-fidf*-run. For the *VCAS-schema*-run the best performance is gained using the exhaustiveness quantisation measure. The differences between the runs however are marginal.

4.2.2 VCAS task - Comparing the variation in the vector-space models

Again additional runs were computed to investigate the effect of the same modifications, as discussed in Section 4.1.2, to the text retrieval component. Figure 6 provides two graphs showing that contrary to the CO runs, the results of the VCAS runs are not dominated by the text retrieval component. Here the performance is mainly determined by the structural constraints defined for the query.

5. CONCLUSIONS

Our goal within INEX was to investigate the influence of the three hypothesis on the retrieval performance. Obviously our system does not belong to the best performing systems.

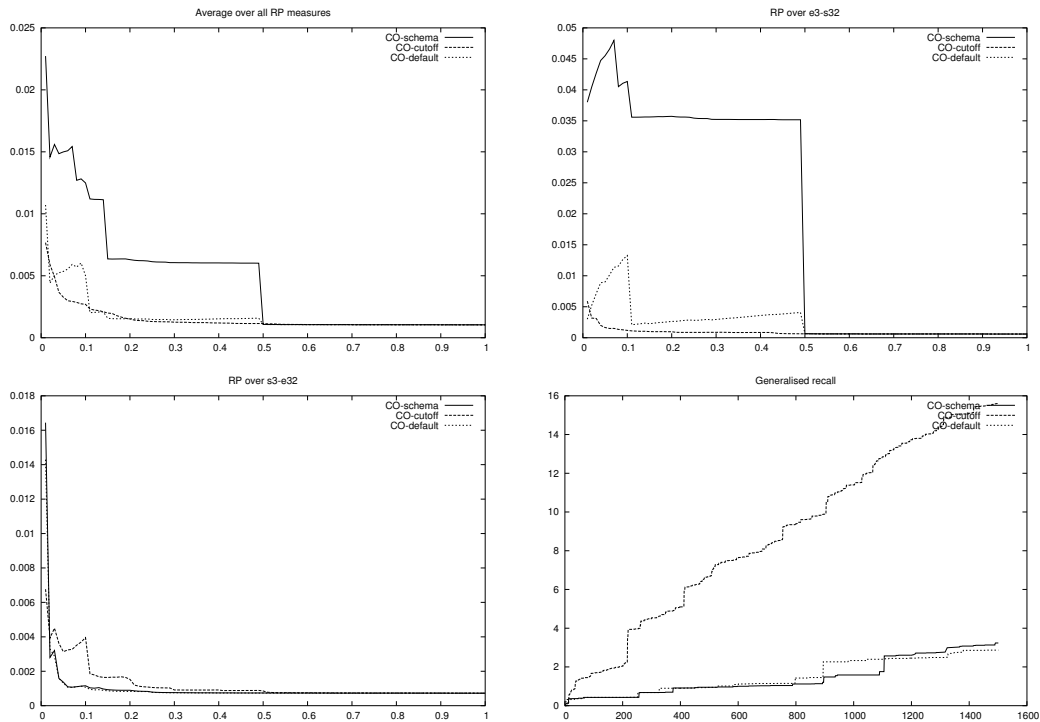


Figure 2: Official runs for the CO task - comparison

Various explanations can be found. Probably the most influential for the CO task, is the lack of a good implementation of the text retrieval component. With respect to the VCAS task we have restricted ourselves to SCAS and not VCAS. It seems that the text retrieval component is less influential in that case, but nevertheless major improvements can be achieved there as well.

However the comparison between the runs that we submitted for the CO task, clearly showed that, if authors use markup to emphasise particular pieces of content that they find of more importance, it makes sense to increase the weights of those document fragments to improve the retrieval performance. The results show that more relevant document fragments are ranked higher in the result list.

On the other hand we can increase the specificity of the retrieved document fragments, by using a so called cutoff node set. The system then returns smaller document fragments that are more relevant for the given topic.

Finally, the runs that were using the fragmented document frequencies (fdf) did not increase the retrieval performance of our system. We feel that this is mainly caused by the absence of a semantical markup of the content of the IEEE document collection. We therefore plead that the XML tagging should not be related to the functional role of the element, but rather have a semantic role.

6. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[2] N. Fuhr, N. Kazai, and M. Lalmas. INEX: Initiative for the evaluation of XML retrieval. In *In Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*, 2000.

[3] G. Kazai. Report of the inex'03 metrics working group. In *In Proceedings of the Second INitiative for the Evaluation of XML Retrieval (INEX) Workshop*, pages 184–190, Dagstuhl, Germany, 2003.

[4] M. Lalmas and S. Malik. Inex 2004 retrieval task and result submission specification, June 2004. http://inex.is.informatik.uni-duisburg.de:2004/internal/pdf/INEX04_Retrieval_Task.pdf.

[5] J. List and A. de Vries. Cwi at inex 2002. In *In Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*. ERCIM Workshop Proceedings, 2002.

[6] S. Malik and M. Lalmas. <http://inex.lip6.fr/2004/metrics/official.php>, 2004.

[7] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[8] A. Trotman and R. A. O'Keefe. The simplest query language that could possibly work. In *Proceedings of the Second Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*. ERCIM Publications, 2004.

[9] R. van Zwol. *Modelling and searching web-based document collections*. Ctiti ph.d. thesis series, Centre for Telematics and Information Technology (CTIT), Enschede, the Netherlands, 26 April 2002.

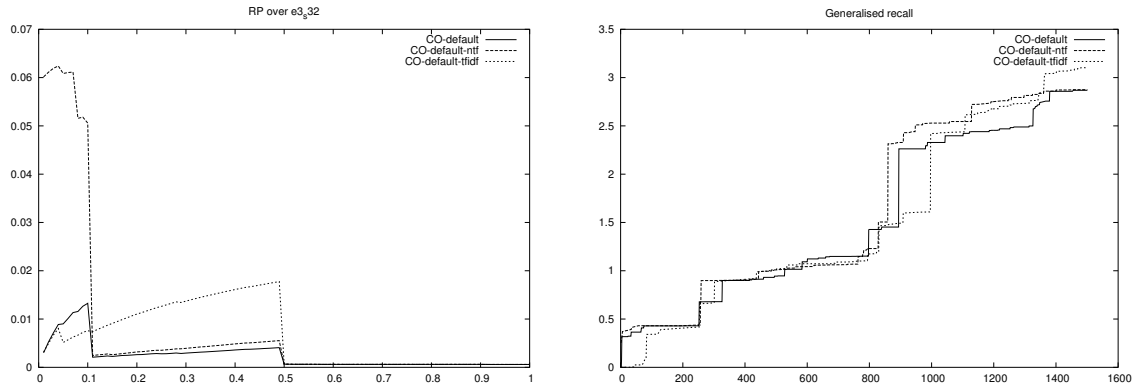


Figure 3: CO task - Variation in vector space model - comparison

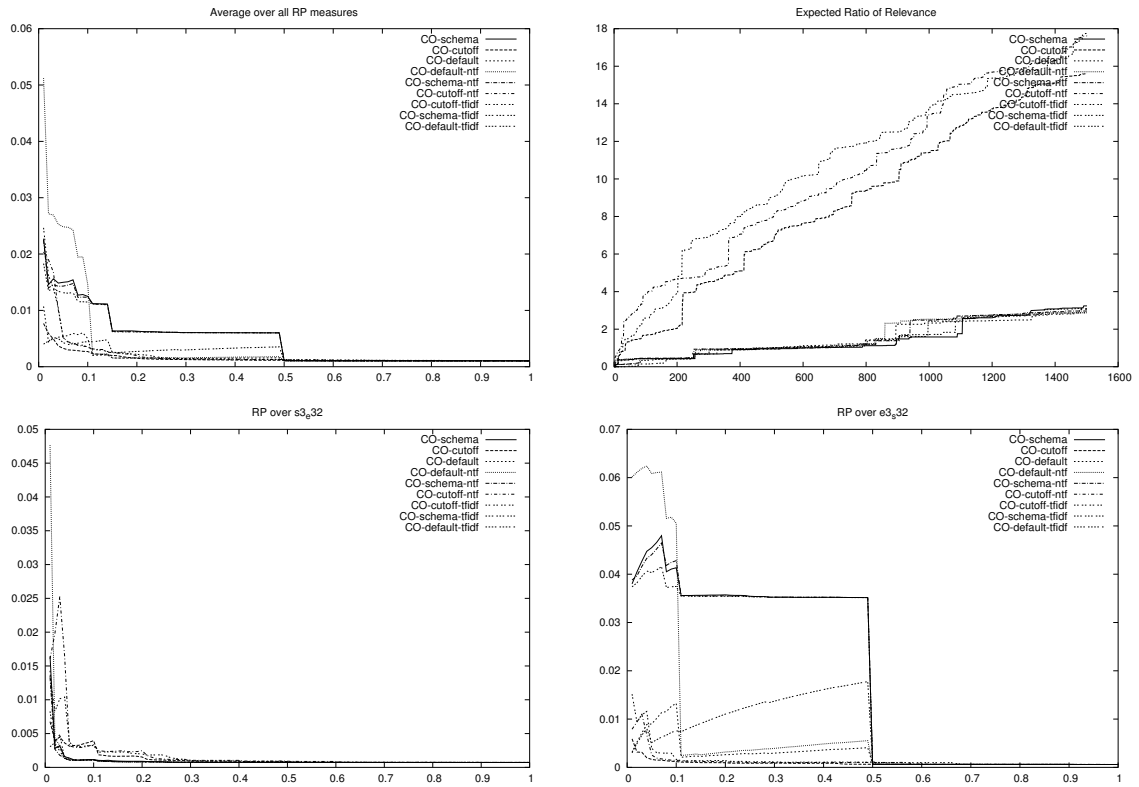
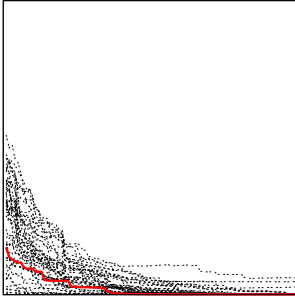


Figure 4: CO task - Overall comparison

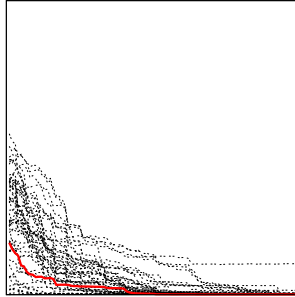
INEX 2004: VCAS-default

quantization: s3_s32; topics: VCAS
average precision: 0.0219
rank: 37 (52 official submissions)



INEX 2004: VCAS-schema

quantization: e3_s32; topics: VCAS
average precision: 0.0218
rank: 38 (52 official submissions)



INEX 2004: VCAS-tdf

quantization: s3_s32; topics: VCAS
average precision: 0.0221
rank: 36 (52 official submissions)

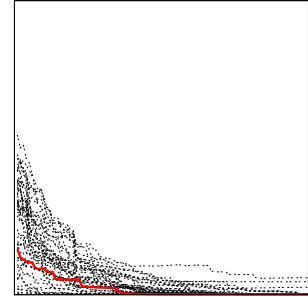


Figure 5: Official runs for the VCAS task - best performances

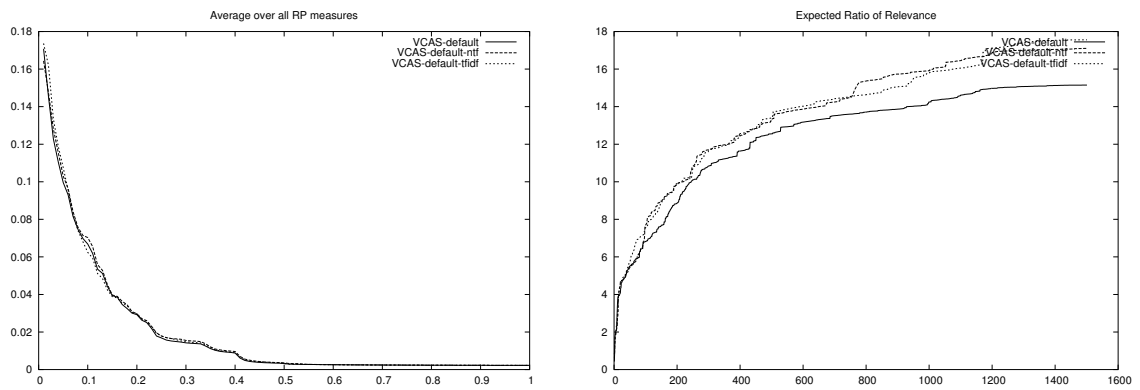


Figure 6: VCAS task - Variation in vector space model - comparison

Hybrid XML Retrieval Revisited

Jovan Pehcevski
RMIT University
Melbourne, Australia
jovanp@cs.rmit.edu.au

James A. Thom
RMIT University
Melbourne, Australia
jat@cs.rmit.edu.au

Anne-Marie Vercoustre
INRIA
Rocquencourt, France
anne-marie.vercoustre@inria.fr

ABSTRACT

In this paper, we report on the participation of the RMIT University group in the INEX 2004 ad-hoc track. Our preliminary analysis of CO and VCAS relevance assessments identifies two complementary cases of modified relevance assessments: General and Specific. Further analysis of the General relevance assessments reveal two categories of retrieval topics: Broad and Narrow. We design runs that follow a hybrid XML approach and implement two retrieval heuristics with different level of overlap among the result elements. We show that for the initial INEX 2004 test collection the overlap CO runs outperform the non-overlap runs, and the heuristic which favours less specific over more specific result elements performs best. Importantly, we present results which show that, in a scenario where users prefer compound and non-overlapping answers to their queries, the choice of using a plain full-text search engine is still a very effective choice for XML retrieval.

Keywords

XML Search & Retrieval, eXist, Zettair, INEX

1. INTRODUCTION

INEX 2004 explores two types of ad-hoc retrieval topics: Content-Only (CO) topics and Vague Content-And-Structure (VCAS) topics. Forty CO topics are used in the CO ad-hoc sub-track, while thirty-five VCAS topics are investigated in the VCAS ad-hoc sub-track.

CO topics do not refer to the existing document structure. An XML retrieval system using these topics may return elements with varying sizes and granularity, prompting a revisit of the issue of *length normalisation* for XML retrieval [4]. Moreover, a large proportion of overlapping result elements may be expected, since the same textual information in an XML document is often contained by more than one element. This *overlap problem* is particularly apparent during evaluation, where the “overpopulated and varying recall base” contains a substantial number of mutually overlapping elements [6].

VCAS topics enforce restrictions on the existing document structure and explicitly specify the target element (such as article, section or paragraph). However, the structural constraints in a VCAS topic need not be strictly matched. This means that not only are the restrictions on document structure *vague* restrictions, but also that the target element could also represent any element considered *likely to be rel-*

evant to the information need. Thus, the same retrieval strategies for CO topics may also be used for VCAS topics, since CO topics may be considered as *loosely restricted* VCAS topics.

The system we use for the ad-hoc track in INEX 2004 follows a *hybrid XML approach*, utilising the best features from Zettair¹ (a full-text search engine) and eXist² (a native XML database). The hybrid approach is a “fetch and browse” [1] retrieval approach, where full articles considered likely to be relevant to a topic are first retrieved by Zettair (the *fetch* phase), and then the most specific elements within these articles are extracted by eXist (the *browse* phase) [9].

The above approach however resulted in rather poor system performance for INEX 2003 CO topics, where Zettair performed better than our initial hybrid system. We have since developed a retrieval module that utilises the structural information in the eXist list of answer elements, and identifies and ranks *Coherent Retrieval Elements* (CREs) [8]. We show elsewhere that this hybrid-CRE system produces performance improvements for the (V)CAS topics [7]. Different heuristic combinations may be used by the CRE module, mainly to determine the final rank of each CRE.

For the INEX 2004 CO sub-track, we use our hybrid system to explore which CRE heuristic combination yields the best retrieval performance, and to investigate whether having non-overlapping result elements in the answer list has an impact on system performance.

For the INEX 2004 VCAS sub-track, we also investigate which retrieval choice — plain queries; queries with structural constraints and no explicitly specified target element; or queries with both structural constraints and a target element — results in a more effective VCAS retrieval.

The remainder of this paper is organised as follows. In Section 2 we undertake a preliminary analysis of the INEX 2004 relevance assessments to identify the types of highly relevant elements. By analysing the relevance assessments for the CO and VCAS topics, we aim to understand what users — or the topic authors who later assess the relevance of returned answer elements — consider to be the most useful. In Section 3 we provide a detailed description of the runs we consider for the CO and the VCAS sub-tracks. In Section 4 we

¹<http://www.seg.rmit.edu.au/zettair/>

²<http://exist-db.org/>

```

<file file="ic/2000/w4036">
<path path="/article[1]" E="3" S="3"/>
. . . . .
<path path="/article[1]/bdy[1]" E="3" S="3"/>
. . . . .
<path path="/article[1]/bdy[1]/sec[3]" E="3" S="3"/>
<path path="/article[1]/bdy[1]/sec[3]/ss1[1]" E="3" S="3"/>
<path path="/article[1]/bdy[1]/sec[3]/ss1[2]" E="3" S="3"/>
<path path="/article[1]/bdy[1]/sec[3]/ss1[3]" E="3" S="3"/>
. . . . .
<path path="/article[1]/bdy[1]/sec[4]" E="3" S="3"/>
<path path="/article[1]/bdy[1]/sec[4]/ss1[2]" E="3" S="3"/>
. . . . .
</file>

```

Figure 1: An extract from the INEX 2004 CO relevance assessments

present results of our CO and VCAS runs. These results reflect different retrieval scenarios based on our analysis of the INEX 2004 relevance assessments. Finally, we conclude in Section 5.

2. ANALYSIS OF INEX 2004 RELEVANCE ASSESSMENTS

Analysing the INEX 2004 CO and VCAS relevance assessments we observe that since neither topic restricts the answer elements, the final answer list may contain elements of different types with varying sizes and granularity. The names of some element types in the XML document collection correspond as follows: **article** to a full article, **abs** and **bdy** to article abstract and article body, **sec**, **ss1** and **ss2** to section and subsection elements, and **p** and **ip1** to paragraph elements. We expect that **article** elements may represent preferable answers for some topics, while for other topics more specific elements may be preferable over **article** elements.

2.1 CO relevance assessments

Figure 1 shows an extract from the INEX 2004 CO relevance assessments. Values for the two INEX relevance dimensions, *exhaustivity*³ (how many aspects of the topic are covered in the element), and *specificity*⁴ (how specific to the topic is the element), are assigned to an **article** and elements within **article** for assessing their relevance to a CO topic.

The focus of our analysis is on *highly relevant* elements. These are elements that — for a given topic — have been assessed as both highly exhaustive and highly specific (E3S3) elements. In Figure 1 there are 8 such elements, including the article itself. These answer elements represent the most useful retrieval elements, even though there is a substantial amount of overlap between them. Following our previous analysis of INEX 2003 relevance assessments [8], we identify two distinct types of highly relevant elements: *General* and *Specific*. Note that, unlike the INEX definitions for exhaustivity and specificity, the definitions for General and Specific (highly relevant) elements result from our analysis as follows.

³E represents the level of exhaustivity (values between 0-3)

⁴S represents the level of specificity (values between 0-3)

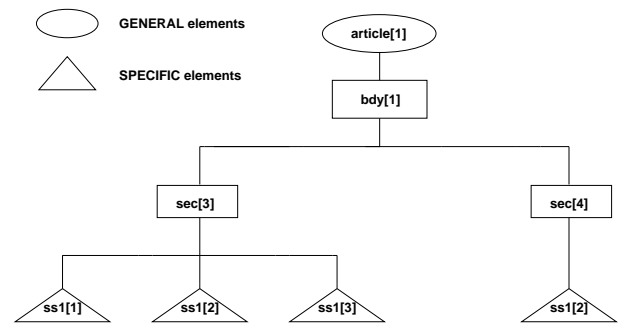


Figure 2: A tree-view example of GENERAL versus SPECIFIC elements.

General:

“For a particular article in the collection, a *General* element is the least-specific highly relevant element containing other highly relevant elements” [8].

Based on the above definition, **article[1]** is the only General element in the example in Figure 1. However, an article may contain several General elements if the article as a whole is not highly relevant. Figure 2 shows a tree representation of all the highly relevant elements shown in Figure 1. The General element is the element shown in ellipse.

Specific:

“For a particular article in the collection, a *Specific* element is the most-specific highly relevant element contained by other highly relevant elements” [8]. In Figure 2, the Specific elements are the highly relevant elements shown in triangles.

When there is only one highly relevant element in an article, that element is both a General and a Specific element.

There are 40 CO topics in INEX 2004 (numbers 162-201). We use version 3.0 of the INEX 2004 relevance assessments, where 34 of the 40 CO topics have their relevance assessments available. Of these, 9 topics do not contain highly relevant (E3S3) elements. Consequently, a total of 25 CO topics are used in our analysis.

Figure 3 shows the overall distribution of the most frequent highly relevant elements (including full articles) that appear in more than half the CO topics. The figure shows three distinct cases when relevance assessments consider all highly relevant elements (*Original* relevance assessments), General highly relevant elements only (*General* relevance assessments) and Specific highly relevant elements only (*Specific* relevance assessments), respectively. The *x*-axis contains the names of the six highly relevant elements that appear in more than half the CO topics (in the case of Original relevance assessments). The *y*-axis contains the number of overall occurrences of each element.

In the case of Original relevance assessments, **p** and **sec** elements occur most frequently, with 691 and 264 overall occurrences, respectively. The **ss1** and **ip1** elements come next, followed by **article** and **bdy** with 99 and 89 overall

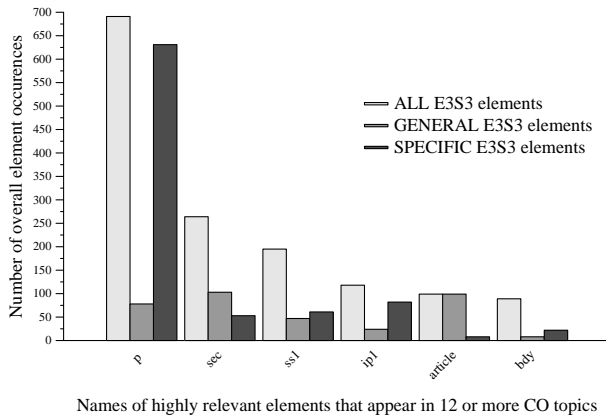


Figure 3: Overall distribution of highly relevant elements that appear in more than half the INEX 2004 CO topics, for three distinct cases of relevance assessments.

occurrences. The latter implies that in most cases when a `bdy` was assessed as highly relevant, the parent `article` is also likely to have been assessed as highly relevant too.

For General relevance assessments, one may expect that the situation should change in favour of the least specific and highly relevant elements. However, in this case `sec` elements are most frequent with 103 overall occurrences, followed by `article` elements with 99 occurrences (however the `article` occurrences are distributed across 16 topics, whereas there are 15 topics where `sec` elements occur). Surprisingly, `p`, `ss1` and `ip1` follow next, with 78, 47 and 24 overall occurrences, respectively. By looking at the number of `bdy` elements, we notice that there are 8 occurrences (distributed across 6 topics) where, when a `bdy` was assessed as highly relevant, the parent `article` has *not* been assessed as highly relevant.

The last case shown in Figure 3 is for Specific relevance assessments. As expected, the situation changes here in favour of the most specific elements, with `p` elements being most frequent. The `ip1`, `ss1`, `sec` and `bdy` come next, followed by only 8 occurrences of `article` elements. The 8 occurrences are distributed across 4 topics, where these `article` elements were the most specific elements assessed as highly relevant.

The two distinct cases of relevance assessments, General and Specific, typically model different user (retrieval) behaviours. Indeed, in the absence of empirically-based models for expected user behaviour, the former case reflects users that prefer compound and more informative answers for their queries, whereas the latter case reflects users that prefer specific, more focused answers for their queries. The knowledge obtained from the above statistics may therefore be appropriately utilised by an XML retrieval system, particularly because distinct cases of relevance assessments favour different types of highly relevant elements.

Topic categories

In the following analysis we consider the case of General relevance assessments. Our aim is to distinguish those CO retrieval topics that seek to mostly retrieve less specific el-

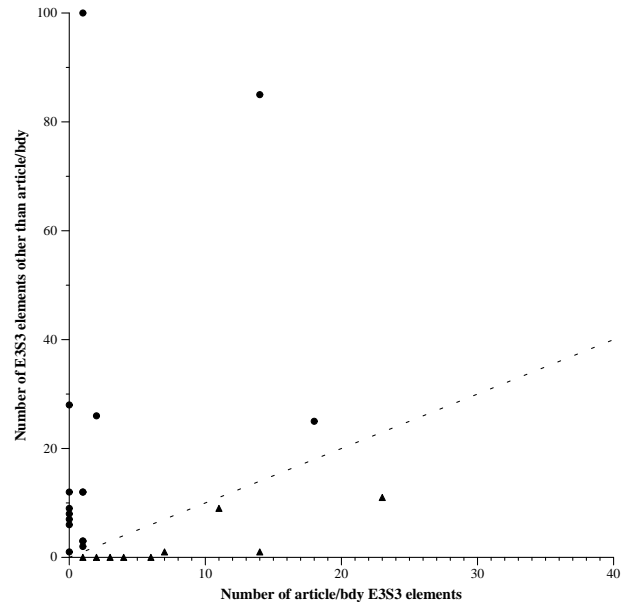


Figure 4: Categories of INEX 2004 CO topics when relevance assessments consider General (highly relevant) elements only.

ements (such as `article` and `bdy`), from those that mostly retrieve other, more specific elements. Consider Figure 4: a point on this graph represents a CO topic. The x -axis shows the total number of General `article` and `bdy` elements contained by a CO topic, whereas the y -axis shows the total number of General elements other than `article` and `bdy` contained by the same topic. For example, the CO topic depicted at coordinates (23,11) contains 23 highly relevant `article/bdy` elements and 11 highly relevant elements other than `article/bdy`.

We use this graph to identify two different categories of INEX 2004 CO topics. The first category, shown as full triangles on the graph and located below the dashed line, favours larger, less specific elements as highly relevant answers. There are 9 such topics (numbers 164, 168, 175, 178, 183, 190, 192, 197 and 198). We refer to these as *Broad* topics.

The second category, shown as full circles on the graph, favours smaller, more specific elements as highly relevant answers. There are 16 such topics. We refer to these as *Narrow* topics.

The above topic categorisation cannot easily be derived in the other two assessment cases, that is, for either the Original or the Specific relevance assessments. However, further analysis shows that four topics (numbers 168, 178, 190 and 198) clearly belong to the Broad category even in this two cases. We observed in our previous work a varying behaviour of an XML retrieval system when its performance is measured against different categories of CO topics [8]. Indeed, it has also been experimentally shown to be a valid observation for a fragment-based XML retrieval system [3]. Thus, it is likely to be useful to distinguish between different categories of CO topics.

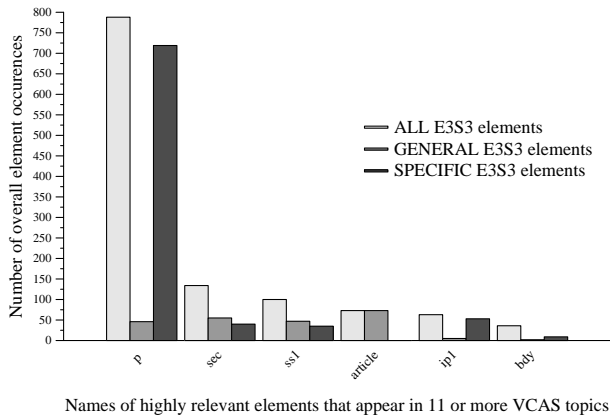


Figure 5: Overall distribution of highly relevant elements that appear in more than half the number of VCAS topics, for three distinct cases of relevance assessments.

2.2 VCAS relevance assessments

There are 35 VCAS topics in INEX 2004 (numbers 127-161). We use version 3.0 of the INEX 2004 relevance assessments, where 26 (out of 35) VCAS topics have their relevance assessments available. Of these, 4 topics do not contain highly relevant (E3S3) elements, and so we limit our analysis to a total of 22 VCAS topics.

Figure 5 shows the overall distribution of the most frequent highly relevant elements that appear in more than half the VCAS topics. The figure also shows three distinct cases of Original, General and Specific relevance assessments.

Since the VCAS relevance assessments have been done in much the same way as those for the CO topics, it is not surprising that graphs in Figures 5 and 3 show similar statistics. In both cases, the number of overall occurrences of `p` elements (in the case of Original and Specific relevance assessments) is far greater than the numbers of all the other elements. Nevertheless, there are some differences for the number of overall occurrences of `article` and `bdy` elements. In the case of VCAS Original relevance assessments, the number of `article` elements is much greater than that of the `bdy` elements (73 `article` occurrences across 12 topics, compared to 36 `bdy` occurrences across 11 topics). For VCAS General relevance assessments, `article` elements are the most frequent among all the other more specific elements. In the case of VCAS Specific relevance assessments, the number of `article` elements is zero, whereas there are 9 highly relevant `bdy` elements, which are distributed across 3 topics.

Topic categories

As for the CO topics, we use the case of General relevance assessments to identify two different categories of INEX 2004 VCAS topics. The first category of topics favours less specific elements as highly relevant answers. There are 6 such topics (numbers 130, 131, 134, 137, 139 and 150), which we refer to as *Broad* topics. The second category of topics favours more specific elements as highly relevant answers. There are 16 such topics, referred to as *Narrow* topics.

An interesting observation is that only two VCAS topics of the Broad category (137 and 139) explicitly ask for retrieving `article` or `bdy` elements in their titles (that is, these elements represent their *target* elements). This is not the case with the other four Broad topics, where two topics ask for `sec` (134 and 150), one asks for `abs` (131), and one asks for `p` (130). Further analysis also shows that surprisingly, the last topic belongs to the Broad category even in the other two cases of Original and Specific relevance assessments.

The above analysis clearly shows that highly relevant elements for VCAS topics do not necessarily represent target elements. We believe that distinguishing between categories of VCAS topics is, similar to the case for the CO topics, important information that an XML retrieval system should use.

3. RUNS DESCRIPTION

3.1 Background

All the runs we consider for the INEX 2004 ad-hoc track are based on the hybrid XML retrieval approach. To determine the ranks of CREs in the final list of answer elements, the CRE module in our hybrid system uses a combination of the following XML-specific heuristics:

1. the number of times a CRE appears in the absolute path of each extracted element in the eXist answer list — more matches (**M**) or fewer matches (**m**);
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 ordering of the XPath sequence in the absolute path of the CRE — nearer to beginning (**B**) or nearer to end (**E**).

There are 16 possible CRE heuristic combinations, since the third heuristic is complementary to the other two and is always applied at the end. We have found that for the INEX 2003 test set, the best results are obtained when using the **MpE** heuristic combination [8]. With **MpE**, less specific and more general elements are ranked higher than more specific and less general elements.

However, we have also observed that different CRE heuristic combinations may be more suitable for different choices of evaluation metrics, where retrieving more specific and less general elements early in the ranking (such as with using the **PME** heuristic) produces better results. We implement and compare these two heuristics in different runs for the ad-hoc track in INEX 2004.

The following sections provide a detailed description of our runs for each (CO and VCAS) sub-track.

3.2 CO sub-track

For the CO sub-track we consider the following runs:

- **Zettair** – using the full-text information retrieval system as a baseline run;

CO run	%Ovp	Different quantisation functions (Original assessments)									
		strict		s3_e321		s3_e32		e3_s321		e3_s32	
		MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
Zettair	0	0.049	0.073	0.008	0.097	0.020	0.088	0.088	0.206	0.071	0.132
Hybrid_MpE	82.2	0.124	0.103	0.041	0.194	0.072	0.174	0.178	0.218	0.155	0.182
Hybrid_MpE_NO	0	0.051	0.076	0.008	0.100	0.020	0.091	0.089	0.209	0.073	0.138
Hybrid_PME	82.1	0.081	0.100	0.038	0.206	0.052	0.182	0.089	0.141	0.083	0.121
Hybrid_PME_NO	0	0.047	0.088	0.023	0.197	0.027	0.165	0.031	0.123	0.034	0.109

Table 1: Performance results of INEX 2004 CO runs when using different quantisation functions and across 25 CO topics. The case of Original relevance assessments is used. For each run, an overlap indicator shows the percentage of overlapping elements in the answer list. Values for the best runs (for each measure and under each function) are shown in bold.

- **Hybrid_MpE** – using the hybrid system with MpE heuristic combination in the CRE module;
- **Hybrid_MpE_NO** – using the hybrid system, with MpE heuristic combination, and no overlap among the elements in the final answer list;
- **Hybrid_PME** – using the hybrid system with PME heuristic combination in the CRE module;
- **Hybrid_PME_NO** – using the hybrid system, with PME heuristic combination, and no overlap among the elements in the final answer list;
- **Hybrid_VCAS_PME** – using the hybrid system with PME heuristic combination in the CRE module. As with the previous run, the structural constraints remain, while the target element is allowed to represent any element;
- **Hybrid_CAS** – using the initial hybrid system (without the CRE module), where the structural constraints and the target element in the **Title** part of each VCAS topic are strictly matched.

Our goals are threefold. First, we aim to explore which heuristic combination yields best performance for the hybrid system under different retrieval scenarios. Second, we aim to investigate the impact of overlapping result elements on system performance. Thus the two cases of non-overlap runs, **Hybrid_MpE_NO** and **Hybrid_PME_NO**, implement different non-overlap strategies: the former allows less specific and more general elements to remain in the list and removes all the other (contained) elements, whereas the latter retains more specific and less general elements, and removes all the other elements that contain them. Finally, by comparing the hybrid runs with the baseline run, we aim to better understand the issues surrounding the CO retrieval task.

3.3 VCAS sub-track

For the VCAS sub-track we consider the following runs:

- **Zettair** – using the full-text information retrieval system as a baseline run;
- **Hybrid_CO_MpE** – using the hybrid system with MpE heuristic combination in the CRE module. The structural constraints and the target element in the **Title** part of each VCAS topic are removed, leaving query terms only.
- **Hybrid_CO_PME** – using the hybrid system with PME heuristic combination in the CRE module. As with the previous run, each VCAS topic is treated as being a CO topic;
- **Hybrid_VCAS_MpE** – using the hybrid system with MpE heuristic combination in the CRE module. The target element in the **Title** part of each VCAS topic is

not explicitly specified (that is, it is allowed to have any granularity), while the structural constraints are strictly matched;

As with the CO runs, we aim to achieve several goals through these VCAS runs. First, we aim to investigate which retrieval choice (CO, VCAS or CAS) results in a more effective VCAS retrieval. Second, for the hybrid runs using the CRE module and a particular retrieval choice, we aim to identify the best choice of heuristic. Finally, by comparing the hybrid runs with the baseline run, we want to empirically check whether we can justify using a plain full-text search engine in the VCAS retrieval task.

4. EXPERIMENTS AND RESULTS

For each of the retrieval runs, the resulting answer list for a CO/VCAS topic comprises up to 1500 articles or elements within articles. To measure the overall performance of each run, two standard information retrieval measures are used: *Mean Average Precision* (MAP), which measures the ability of a system to return relevant elements, and *Precision at 10* (P@10), which measures the number of relevant elements within the first 10 elements returned by a system.

In INEX 2004, an evaluation metric with different quantisation functions is used to evaluate the retrieval effectiveness of XML systems [5]. Thus, the exhaustivity and specificity values for *relevant* elements may vary depending on the choice of quantisation function. For example, if the strict quantisation function (**e3_s3**) is used, MAP will measure the ability of a system to return *highly relevant* (E3S3) elements, whereas if the **e3_s321** or **s3_e321** functions are used, MAP will measure the ability of a system to return *highly exhaustive* (E3S3, E3S2, E3S1) or *highly specific* (E3S3, E2S3, E1S3) elements. In the following we describe results obtained from evaluating the retrieval effectiveness of our runs for each CO and VCAS sub-track.

CO run	%Ovp	Strict quantisation function (General assessments)					
		All topics		Broad topics		Narrow topics	
		MAP	P@10	MAP	P@10	MAP	P@10
Zettair	0	0.154	0.073	0.364	0.211	0.036	0.024
Hybrid_MpE	82.2	0.126	0.050	0.240	0.056	0.062	0.048
Hybrid_MpE_NO	0	0.152	0.073	0.359	0.211	0.036	0.024

Table 2: Performance results of three INEX 2004 CO runs when using the strict quantisation function and different CO topic categories. The case of General relevance assessments is used. For each run, an overlap indicator shows the percentage of overlapping elements in the answer list. Values for the best runs (for each measure and under each topic category) are shown in bold.

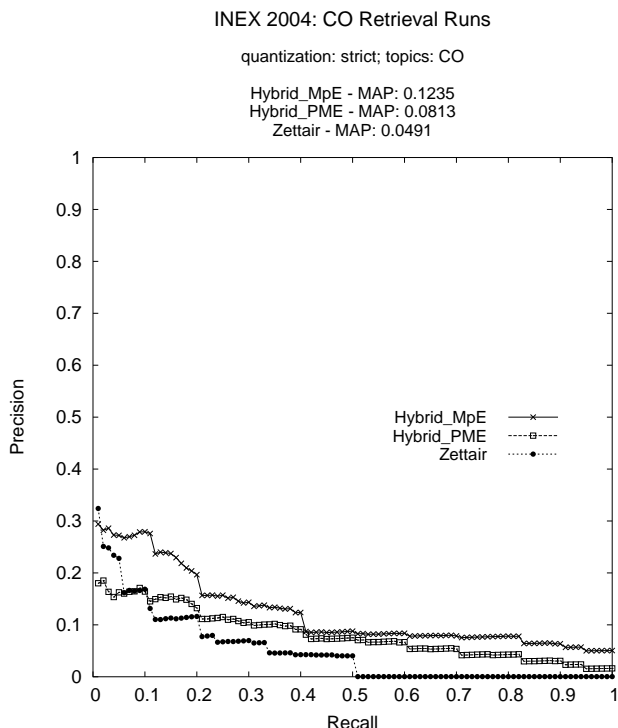


Figure 6: Evaluation of three INEX 2004 CO retrieval runs using strict quantisation function and the case of Original relevance assessments.

4.1 CO sub-track

Table 1 shows evaluation results for the CO retrieval runs when the case of Original relevance assessments is considered. Different quantisation functions are used to evaluate the retrieval effectiveness, and values for the best runs (for each measure under each function) are shown in bold. Several observations can be drawn from these results.

First, for overlap runs using the hybrid system, the MpE heuristic yields better performance than the PME heuristic, except for the *highly specific* quantisation functions (*s3_e321* and *s3_e32*), where the number of relevant elements in the first 10 returned elements is on average higher when using the PME heuristic.

Second, the non-overlap hybrid runs perform worse than the corresponding overlap hybrid runs. This is very likely to be a

result of the varying CO recall base, which, as previously discussed in Section 2.1, is as apparent in INEX 2004 as it was in INEX 2003 [8]. We revisit the latter comparison in the next section, where a non-varying recall base is considered for evaluation (the case of General relevance assessments).

Last, the hybrid runs perform better on average than the baseline run, except for the *highly exhaustive* quantisation functions (*e3_s321* and *e3_s32*), where the baseline run is competitive, and, with the P@10 measure, performs even better than both the overlap and the non-overlap Hybrid_PME runs. These results show that when highly exhaustive elements are the target of retrieval, a full-text search engine could still be used to satisfy the information need almost as well.

Figure 6 shows recall/precision curves for the two overlap hybrid runs (Hybrid_MpE and Hybrid_PME) and the baseline run (Zettair). The runs are evaluated by using the strict quantisation function and the case of Original relevance assessments. For low recall (0.1 and less), Zettair outperforms Hybrid_PME, although its performance gradually decreases and reaches zero for 0.5 (and higher) recall. Overall, Hybrid_MpE performs best and is substantially better than Hybrid_PME.

General CO retrieval scenario

In the following analysis, we use the strict quantisation function and the case of General relevance assessments to compare the performance of the two Hybrid_MpE runs (overlap and non-overlap) with Zettair. When a run is evaluated with the strict quantisation function, the case of General relevance assessments reflects a non-overlapping recall base, since an article is allowed to only contain General, non-overlapping (highly relevant) elements (see Section 2.1 for definition of General elements). Moreover, our previous analysis has distinguished two different categories of CO topics. Thus, in this General retrieval scenario, the performance of the above runs are also compared across three topic categories: the *All topics* category, with all the 25 CO topics, and the *Broad* and the *Narrow* categories, with 9 and 16 CO topics, respectively.

Table 2 shows the evaluation results for each run. Two observations are clear in the cases of *All* and *Broad* topic categories: first, with both MAP and P@10 measures Zettair performs best, although with P@10 the non-overlap hybrid run (MpE_NO) performs the same as Zettair; and second, unlike for the case of varying recall base (the case of Original

VCAS run	%Ovp	Different quantisation functions (Original assessments)									
		strict		s3_e321		s3_e32		e3_s321		e3_s32	
		MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
Zettair	0	0.052	0.119	0.012	0.146	0.021	0.146	0.063	0.296	0.033	0.154
Hybrid_CO_MpE	78.3	0.101	0.104	0.037	0.200	0.056	0.158	0.110	0.262	0.084	0.162
Hybrid_CO_PME	78.2	0.034	0.096	0.029	0.189	0.036	0.135	0.068	0.204	0.051	0.123
Hybrid_VCAS_MpE	67.8	0.103	0.154	0.027	0.235	0.047	0.192	0.107	0.323	0.078	0.227
Hybrid_VCAS_PME	67.8	0.045	0.142	0.021	0.227	0.029	0.187	0.072	0.258	0.059	0.196
Hybrid_CAS	5.4	0.032	0.142	0.018	0.212	0.026	0.173	0.030	0.200	0.034	0.189

Table 3: Performance results of INEX 2004 VCAS runs when using different quantisation functions and across 22 VCAS topics. The case of Original relevance assessments is used. For each run, an overlap indicator shows the percentage of overlapping elements in the answer list. Values for the best runs (for each measure and under each function) are shown in bold.

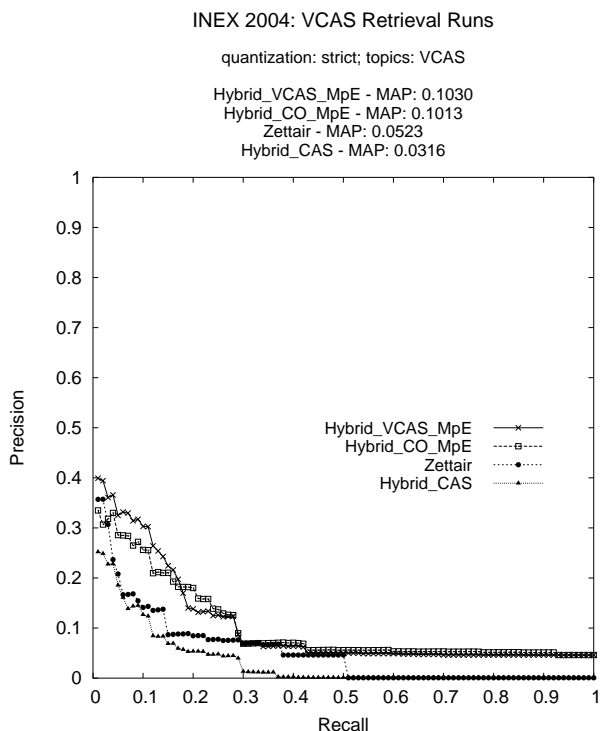


Figure 7: Evaluation of four INEX 2004 VCAS retrieval runs using strict quantisation function and the case of Original relevance assessments.

nal relevance assessments), the non-overlap hybrid run substantially outperforms the overlap hybrid run. In the case of *Narrow* topics, the overlap hybrid run performs best, whereas the performance of the other two runs is the same.

4.2 VCAS sub-track

Table 3 shows evaluation results for the VCAS retrieval runs and the case of Original relevance assessments. Different quantisation functions are used to evaluate the retrieval effectiveness, and values for the best runs (for each measure under each function) are shown in bold. Several observations can be drawn from the results of Table 3.

First, the strict hybrid run (Hybrid_CAS) (where structural constraints and the target element of a VCAS topic are strictly matched) performs worse than the other hybrid runs. This is not surprising, since the relevance assessments for VCAS topics have been done in the same way as those for CO topics. Moreover, when using strict quantisation function (with both MAP and P@10) Hybrid_VCAS runs (the choice of strict structural constraints and no explicit target element) perform better than Hybrid_CO runs (the choice where plain CO queries are used).

Second, as with CO topics the MpE heuristic in hybrid runs yields better performance than the PME heuristic.

Last, the hybrid runs perform better on average than the baseline run, except when using strict quantisation function (with MAP), where Zettair performs better than the strict hybrid run and both the hybrid-PME runs.

Figure 7 shows recall/precision curves for the three hybrid runs that use different retrieval choices (CO, VCAS and CAS) and the baseline run (Zettair). The runs are evaluated using the strict quantisation function and the case of Original relevance assessments. The VCAS run performs best, particularly for low recall (0.2 and less), however its performance is almost identical with that of the CO run for 0.3 (and higher) recall. When highly relevant elements are target of retrieval, Zettair clearly outperforms the strict (CAS) hybrid run.

General VCAS retrieval scenario

In this scenario we use the strict quantisation function and the case of General relevance assessments to compare the performance of three hybrid VCAS runs (with retrieval choices CO, VCAS and CAS) with Zettair. The three VCAS topic categories are also used in this analysis: the *All* category, with all the 22 VCAS topics, and the *Broad* and *Narrow* categories, with 6 and 16 VCAS topics, respectively.

Table 4 shows the evaluation results for each run. One observation is very clear: for each VCAS topic category (with both MAP and P@10 measures), Zettair by far outperforms all the other runs. This is a very interesting observation, since the unit of retrieval in Zettair is a full article, and queries used are plain content-only queries. For each VCAS topic category (with P@10 measure), the strict hybrid run also outperforms the other two hybrid runs. Of these, the

VCAS run	%Ovp	Strict quantisation function (General assessments)					
		All topics		Broad topics		Narrow topics	
		MAP	P@10	MAP	P@10	MAP	P@10
Zettair	0	0.192	0.119	0.625	0.367	0.029	0.045
Hybrid_CO_MpE	78.3	0.128	0.035	0.417	0.100	0.020	0.015
Hybrid_VCAS_MpE	67.8	0.128	0.046	0.412	0.100	0.021	0.030
Hybrid_CAS	5.4	0.061	0.085	0.162	0.233	0.023	0.040

Table 4: Performance results of four INEX 2004 VCAS runs when using strict quantisation function and different CO topic categories. The case of General relevance assessments is used. For each run, an overlap indicator shows the percentage of overlapping elements in the answer list. Values for the best runs (for each measure and under each topic category) are shown in bold.

VCAS run again performs better (overall) than the CO run.

5. CONCLUSIONS

In this paper we have reported on our participation in the ad-hoc track of INEX 2004. We have designed and submitted different runs for each CO and VCAS sub-track to investigate different aspects of the XML retrieval task.

The two different cases of INEX 2004 relevance assessments, which were identified as a result of our analysis, model different user behaviours; we have shown that the preferred retrieval aspects vary on the user model used. Moreover, distinguishing between existing topic categories can, in some assessment cases, influence the choice of these aspects.

For the CO sub-track, we have shown that where users prefer less specific and non-overlapping answers, a full-text search engine alone can satisfy user information needs. Our hybrid system, which is also capable of retrieving non-overlapping compound answers, is another effective alternative. However, our results also show that a system should also distinguish between different categories of CO retrieval topics. For a particular topic category, an XML system capable of retrieving more focused — and possibly overlapping — answers is a better choice.

For the VCAS sub-track, in the same retrieval scenario where users prefer less specific and non-overlapping answers to their queries, the same choice of using a full-text search engine, which ignores all the structural constraints and target elements, is very effective. Distinguishing between different topic categories in this case does not appear to make any difference on performance.

We have also used Zettair and the hybrid system in the INEX 2004 Heterogeneous track. However, since the relevance assessments for the heterogeneous XML collections are not yet available, we do not report their performance results in this paper.

The performance values for our INEX 2004 runs, generated with MAP and P@10, are much lower comparing to the same values for information retrieval systems retrieving whole documents. It is our hope that this work will aid better understanding of the different aspects of the XML retrieval task, and ultimately lead to more effective XML retrieval.

6. ADDITIONAL AUTHORS

S.M.M. Tahaghoghi, RMIT University, Melbourne, Australia.
E-mail: saied@cs.rmit.edu.au.

7. REFERENCES

- [1] Y. Chiamarella, P. Mulhem, and F. Fourel. A Model for Multimedia Information Retrieval. Technical report, FERMI ESPRIT BRA 8134, University of Glasgow, April 1996.
- [2] N. Fuhr, M. Lalmas, and S. Malik, editors. *INitiative for the Evaluation of XML Retrieval (INEX)*. *Proceedings of the Second INEX Workshop*. Dagstuhl, Germany, December 15–17, 2003, March 2004.
- [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 Fuhr et al. [2], pages 81–88.
- [4] J. Kamps, M. de Rijke, and B. Sigurbjoernsson. Length Normalization in XML Retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 80–87, 2004.
- [5] G. Kazai. Report on the INEX2003 Metrics working group. In Fuhr et al. [2], pages 184–190.
- [6] 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*, pages 72–79, 2004.
- [7] J. Pehcevski, J. A. Thom, and A.-M. Vercoustre. Enhancing Content-And-Structure Information Retrieval using a Native XML Database. In *Proceedings of The First Twente Data Management Workshop (TDM'04) on XML Databases and Information Retrieval*, pages 24–31, 2004.
- [8] J. Pehcevski, J. A. Thom, and A.-M. Vercoustre. Hybrid XML Retrieval: Combining Information Retrieval and a Native XML Database. *Journal of Information Retrieval: Special Issue on INEX (accepted for publication)*, 2004.
- [9] J. Pehcevski, J. A. Thom, and A.-M. Vercoustre. RMIT INEX Experiments: XML Retrieval using Lucy/eXist. In Fuhr et al. [2], pages 134–141.

A Voting Method for XML retrieval

Gilles Hubert

⁽¹⁾ IRIT/SIG-EVI, 118 route de Narbonne, 31062 Toulouse cedex 4

⁽²⁾ ERT34, Institut Universitaire de Formation des Maîtres, 56 av. de l'URSS, 3400 Toulouse

hubert@irit.fr

ABSTRACT

This paper describes the retrieval approach proposed by the SIG/EVI group of the IRIT research center in INEX'2004 evaluation. The approach uses a voting method coupled with some processes to answer content only and content and structure queries. This approach is based on previous works we led in the context of automatic text categorization.

Keywords

Information Retrieval, XML retrieval, voting method, automatic text categorization

1. INTRODUCTION

The development of systems to perform searches in collections constituted of XML (eXtensible Markup Language) documents [3] has become a need since the use of XML is growing. Consequently, a growing number of systems intend to provide means to retrieve relevant components among XML documents. XML retrieval systems need to take into account content and structural aspects.

Regarding the variety of proposed XML retrieval systems it is interesting to evaluate their effectiveness. For that, the Initiative for the Evaluation of XML retrieval (INEX) provides a testbed and scoring methods allowing participants to evaluate and compare their results.

Underlying approaches of systems participating to INEX can be classified in two categories [5] : model-oriented approaches and system-oriented approaches. Model-oriented approaches gather notably approaches based on language models [11][8][1] or other probabilistic models [14] which obtained good results in 2003. System-oriented approaches extend textual document retrieval system adding XML-specific processing. Various systems in this category [10][6][13][15] obtained good results in 2003.

In this paper, we present an IR approach initially applied to automatic categorization of structured documents according to concept hierarchies and its evolution brought for XML retrieval notably within the context of INEX.

Section 2 is a short presentation of the INEX initiative 2004 edition. Section 3 presents the initial context in which the method was initiated and its first application within INEX in 2003. The evolutions made to this approach for INEX 2004 are described in section 4. Section 5 presents the submitted runs and the obtained

results. In section 6 we conclude analyzing the experiment and considering future works.

2. THE INEX INITIATIVE

2.1 Collection

The INEX documents correspond to approximately 12,000 articles of the IEEE Computer Society's publications from 1995 to 2002 marked up in XML. All the documents respect the same DTD. The collection gathers over eight millions XML elements of varying length and granularity (ex. title, paragraph or article).

2.2 Queries

INEX introduces two types of queries:

- CO (Content Only) queries describe the expected content of the XML elements to retrieve.
- CAS (Content and Structure) queries combine content and explicit references to the XML structure using a variant of Xpath [4]. CAS topics contain indications about the structure of expected XML elements and about the location of expected content.

Both CO and CAS topics are made up of four parts: topic title, topic description, narrative and keywords.

Within the ad-hoc retrieval task, two types of tasks are defined: (1) the CO task, using CO queries, (2) the VCAS task, using CAS queries, for which the structural constraints are considered as vague conditions.

3. A VOTING METHOD IN INFORMATION RETRIEVAL

The approach we proposed is derived from a process we first defined for textual document categorisation [7][2]. Document categorisation intends to link documents with pre-defined categories. Our approach focuses on categories organised as taxonomy. The original aspect of our approach is that it involves a voting principle instead of a classical similarity computing. The association of a text to categories is based on the Vector Voting method [12]. The voting process evaluates the importance of the association between a given text and a given category.

This method is similar to the HVV method (Hyperlink Vector Voting) used within the Web context to compute the relevance of a Web page regarding the web sites referring to it [9]. In our context, the initial strategy considers that the more the category terms appear in the text, the more the link between the text and this category is strong. Thus, this method relies on terms

describing each category and their automatic extraction from the document to be categorised. The result is a list of categories annotating each document.

For INEX'2003, this categorisation process has been applied. Every XML component has been processed as a complete document. Every topic has been considered as a category of a flat taxonomy. The result was a list of topics corresponding to each XML component. It was then reversed and reordered to fit the INEX format of results.

Results obtained for the submitted runs [16] have led us to improve the process to suit a retrieval process. The axes of this evolution have been as follows:

- inverse the voting process to estimate the relevance of each XML component according to each topic,
- modify the voting function to take into account the great variations of element sizes and to take into account topic treatment rather than category treatment,
- integrate the aggregation aspect of an XML element (i.e. elements composed of relevant elements),
- integrate structural constraint processing for CAS topics.

4. EVOLUTION OF THE VOTING METHOD WITHIN INEX

4.1 INEX collection pre-processing

From the INEX collection point of view, the documents are considered as sets of text chunks identified by xpaths. For each XML component, concepts are extracted automatically and saved with the xpath identifying the XML component in which they appear and the number of occurrences in the component. Concept extraction involves notably stop word removal. Optionally, some processes can be applied to concepts such as stemming using Porter's algorithm. For INEX'2004 experiments all XML tags except text formatting tags (bold, italic, underline) have been taken into account.

From the topic point of view, although our method can use all the parts constituting CO and CAS topics, we used only the title part for the INEX'2004 experiments as requested. For both topic types, stop words are removed and optionally terms can be stemmed using Porter's algorithm.

4.2 Voting function

The voting function must take into account the importance in the XML element of each term describing the topic and the importance of each term in the topic representation. We have studied different voting functions and the one providing the best results is described as follows:

$$Vote(E, T) = \sum_{\forall t \in T} F(t, E) \cdot \frac{F(t, T)}{S(T)}$$

where

T is the topic

E is an XML element

$F(t, E)$ This factor measures the importance of the term t in the XML element E. $F(t, E)$ corresponds the

number of occurrences of the term t in the element E.

$$\frac{F(t, T)}{S(T)}$$

This factor measures the importance of the term t in the topic representation T. $F(t, T)$ corresponds to the number of occurrences of the term t in the topic T and $S(T)$ corresponds to the size (number of terms) of T.

The voting function combines two factors: the presence of a term in the element and the importance of this term in the topic.

4.3 Scoring function

The voting function is coupled with a third factor representing the importance of the topic presence within the XML element.

The final function (scoring function) that computes the score of an XML element regarding a given topic is the following:

$$Score(E, T) = Vote(E, T) \cdot f\left(\frac{NT(T, E)}{S(T)}\right)$$

where

$$\frac{NT(T, E)}{S(T)}$$

This factor measures the presence rate of terms representing the topic in the text (importance of the topic). $S(T)$ corresponds to the number of terms in the topic representation T and $NT(T, E)$ corresponds to the number of terms of the topic T that appear in the XML element E.

Applying a function f to the third factor (i.e. the presence rate of terms representing the topic in the text) aims at varying the influence of this factor on the scoring function. We tried different functions f , for example the initial function was the exponential (i.e. $f\left(\frac{NT(T, E)}{S(T)}\right) = e^{\frac{NT(T, E)}{S(T)}}$).

4.4 Additional processes for both CO and CAS topics

The scoring function is completed with the notion of *coverage*. The aim of the coverage is to ensure that only documents in which the topic is represented enough will be selected for this topic. The coverage is a threshold corresponding to the percentage of terms from a topic that appears in a text. For example, 50% of coverage implies that at least half of the terms describing a topic have to appear in the text of a document to select it.

$$\text{If } \frac{NT(T, E)}{S(T)} \geq CT \text{ then}$$

$$Score(E, T) = Vote(E, T) \cdot f\left(\frac{NT(T, E)}{S(T)}\right)$$

else

$$Score(E, T) = 0$$

The hierarchical structure of XML has to be taken into account. The hypothesis on which is based our system is that an element containing a component selected as relevant is also relevant. Our system 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.

$\forall E_a$ ancestor of E and $d(E_a, E) \cdot \alpha < 1$

$$Score(E_a, T) = Score(E_a, T) \cdot (1 - d(E_a, E) \cdot \alpha) \cdot Score(E, T)$$

where

α is a constant coefficient 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/body/p/s/ss1 the distance between ss1 and body is equal to 3 i.e. $d(\text{body}, \text{ss1})=3$)

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 hypothesis chosen for INEX is quite different notably due to relevance dimensions: exhaustivity and specificity. Considering exhaustivity, a composed element is considered at least as relevant as the most relevant of its components. Considering specificity, the relevance of an element composed of several relevant components is less or equal to the relevance of the most relevant component. It would be interesting to evaluate the impact of this difference of relevance propagation on the retrieval results of our system.

In addition, in INEX, terms constituting a topic title can have either the prefix + or -. The sign + is used to emphasize a concept and - denotes an unwanted concept. The + and - signs do not have strict semantics but just indicate preferences wished by the topic's author. An element containing a term prefixed by - in the topic title can be judged relevant to the information need. In the same way, an element judged relevant to the information need even if it does not contain the term prefixed by + in the topic title.

To take into account the possibility of having prefixed terms, a coefficient is associated to each term. A coefficient is fixed for each case: term not prefixed, term with the prefix + and term with the prefix -.

$$Vote(E, T) = \sum_{\forall t \in T} sc(t, T) \cdot F(t, E) \cdot \frac{F(t, T)}{S(T)}$$

where

$sc(t, T) = \text{value 1}$ if t has the prefix - in the topic

$sc(t, T) = \text{value 2}$ if t has no prefix in the topic

$sc(t, T) = \text{value 3}$ if t has the prefix + in the topic

4.5 Specific processes for CAS topics

On one hand, we take into account different types of constraints on content. Structural constraints on xpath of elements which are expected to contain keywords (e.g. about(/p,'+authorization + "access control" +security') and constraints on the year of the article (e.g. //yr <='2000') are taken into account. These kinds of structural constraints on content gathered all the constraints appearing in the CAS topics of INEX'2004. The voting method applied to CO topics has been extended to take into account such constraints as follows:

$$Vote(E, T) = \sum_{\forall t \in T} (1 + \beta) \cdot F(t, E) \cdot \frac{F(t, T)}{S(T)}$$

where

if E matches a structural constraint defined on t then

$$\beta > 0$$

else

$$\beta = 0$$

On the other hand, an additional step identifies the structural constraints on target elements indicated in CAS topics. All the structural constraints defined on target elements of topics are taken into account and stored to be processed in a post-voting step to enrich the results issued from the voting step.

For VCAS evaluation, the target constraint specified in the topic does not have to be strictly verified. The constraint is rather regarded as a hint for expected results without eliminating the elements which do not satisfy the target constraint.

To take into account these principles, the score associated to the elements of the results that match the expected xpaths are increased. A factor is applied to the score of matching elements as follows:

If R matches X then

$$Score(E, T) = \gamma \cdot Vote(E, T) \cdot f\left(\frac{NT(T, E)}{S(T)}\right)$$

where $\gamma > 1.0$

5. EXPERIMENTS

5.1 Experiment setup

Our experiments aim at evaluating the efficiency of the evolution given to the voting function and the coefficient adjustments resulting from training performed on the INEX'2003 assessment testbed. The training phase only concerns system processes applied to both CO and CAS topics.

Three runs based on the voting method were submitted to INEX'2004. Two runs were performed on CO topics and one run was performed on CAS topics.

The runs on CO topics differ from the function f used in the voting method. The run labelled VTCO2004TC35xp400sC-515 uses the voting function:

$$Score(E, T) = Vote(E, T) \cdot \phi^{\left(\frac{NT(T, E)}{S(T)}\right)}$$

where $\phi=400$.

The run labelled VTCO2004TC35p4sC-515 uses the voting function:

$$Score(E, T) = Vote(E, T) \cdot \left(\frac{NT(T, D)}{S(T)}\right)^\lambda$$

where $\lambda=4$.

The run on CAS topics labelled VTCAS2004TC35xp200sC-515PP1 uses the voting function:

$$Score(E, T) = Vote(E, T) \cdot \phi^{\left(\frac{NT(T, E)}{S(T)}\right)}$$

where $\phi=200$.

The coefficient taking into account structural predicates associated to searched concepts was fixed to 1.0 (i.e. the vote of an element regarding a given concept is doubled when the element matches the structural constraint associated to the concept). The coefficient taking into account structural predicates for expected results was fixed to 2.0 (i.e. the score of an element matching the structural predicate is doubled). The values of these two coefficients were fixed arbitrarily.

For all submitted runs the other parameters of the scoring function were the same. 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).

Coefficients applied to take into account the signs '+' and '-' used to emphasise a concept or to denote an unwanted one were fixed to:

- +5.0 for concepts marked with '+' (the vote of these concepts increases the score of the elements in which they appear),
- -5.0 for concepts marked with '-' (the vote of these concepts reduces the score of the elements in which they appear),
- 1.0 for unmarked concepts.

The values of the parameters are those which gave the best results during a training phase done with INEX'2003 CO topics.

5.2 Results

The following table shows the preliminary results of the three runs based on the voting method:

Run	Aggregate score	Rank
VTCO2004TC35xp400sC-515	0.0783	13/70
VTCO2004TC35p4sC-515	0.0775	15/70
VTCAS2004TC35xp200sC-515	0.0784	5/51

Table 1: Results of the 3 runs performed using the voting method

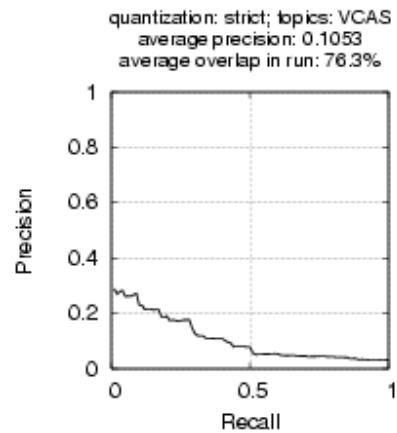
The results of the two runs for CO topics are detailed in the following table:

Quantisation	VTCO2004TC35xp400sC-515		VTCO2004TC35p4sC-515	
	Average precision	Rank	Average precision	Rank
strict	0.0778	18/70	0.0759	19/70
generalised	0.0683	14/70	0.0682	15/70
so	0.0559	16/70	0.0564	15/70
s3_e321	0.0395	22/70	0.0400	21/70
s3_e32	0.0508	17/70	0.0508	17/70
e3_s321	0.1456	10/70	0.1424	11/70
e3_s32	0.1106	11/70	0.1083	13/70

Table 2: Detailed results of the 2 runs for CO topics

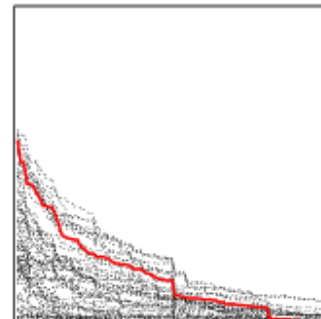
For CO topics, the run which has obtained the best results is the run labelled VTCO2004TC35xp400sC-515. The best measures have been obtained with e3s321 quantisation. Average precision is equal to 0.1456, placing the run at the 10th rank. The run labelled VTCO2004TC35p4sC-515 has obtained values slightly lower for most of the quantisations. Only the best results obtained for CO topics are presented in the following graphs that is to say run VTCO2004TC35xp400sC-515 for e3s321 quantisation.

INEX 2004: VTCAS2004TC35xp200sC-515PP1



X 2004: VTCO2004TC35xp400sC-51

quantization: e3_s321; topics: CO
average precision: 0.1456
rank: 10 (69 official submissions)



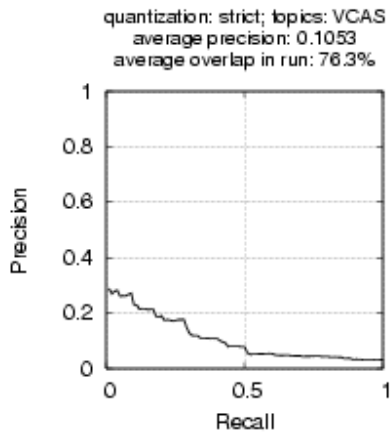
For CAS topics, the run VTCAS2004TC35xp200sC-515PP1 has been ranked at the 5th place. The results of the run are detailed in the following table:

Quantisation	VTCAS2004TC35xp200sC-515PP1	
	Average precision	Rank
strict	0.1053	5/51
generalised	0.0720	6/51
so	0.0554	9/51
e3_e321	0.0462	12/51
e3_e32	0.0644	10/51
e3_s321	0.1162	5/51
e3_s32	0.0892	5/51

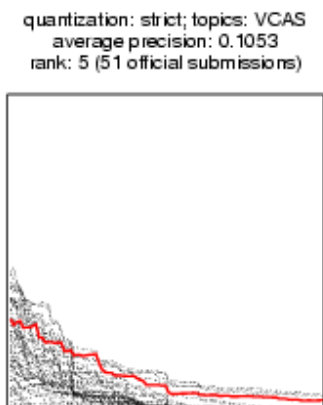
Table 3: Detailed results of the run for CAS topics

The best measures have been obtained for quantisations strict, e3s321 and e3s32 for which the run is ranked 5. The following figures present the results corresponding to the strict quantisation and e3s321 quantisation.

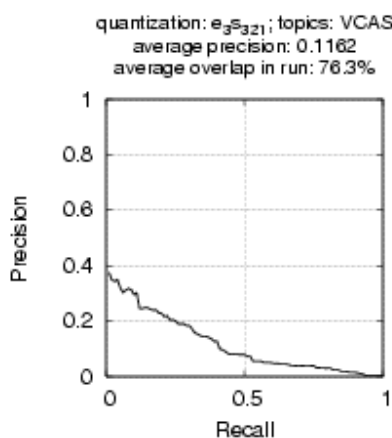
INEX 2004: VTCAS2004TC35xp200sC-515PP1



2004: VTCAS2004TC35xp200sC-515

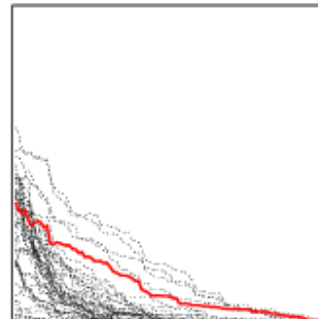


INEX 2004: VTCAS2004TC35xp200sC-515PP1



2004: VTCAS2004TC35xp200sC-515

quantization: e3s321; topics: VCAS
average precision: 0.1162
rank: 5 (51 official submissions)



6. DISCUSSION AND FUTURE WORKS

Regarding the experiments that were performed and the obtained results we can notice that:

- the chosen functions and parameters for the scoring method tend to support exhaustivity rather than specificity. Indeed, the importance of the factor measuring the representation of the topic (i.e. $NT(T,E)/S(T)$) dominates in the scoring function and this factor is related to the exhaustivity relevance. It would be interesting to modify the scoring function to increase the number of elements judged as relevant regarding specificity.
- The measures obtained using INEX'2003 CO topics were globally better. This suggests that our scoring method is more efficient on certain queries. It would be interesting to identify a class (or classes) of queries for which the function works better, a class (classes) of queries for which the function is less efficient and to understand why. The function could evolve to extend its efficiency to other kinds of queries or different functions could be applied regarding different query classes.
- The values of coefficients applied for structural constraint matching have been fixed arbitrarily. Additional experiments on INEX'2004 CAS topics will help us to adjust the values of these coefficients.
- Evaluate the profit of adding a relevance feedback process to our method. On one hand, feedback from first ranked elements of the assessments can be performed. This is the process chosen this year in the relevance feedback track. On the other hand, we plan to integrate a feedback process using first ranked elements of a first search using our system.

7. REFERENCES

- [1] Abolhassani, M., Fuhr, N., Applying the Divergence from Randomness Approach for Content-Only Search in XML Documents. 26th European Conference on IR Research (ECIR), LNCS 2997, p. 409-419, Apr 2004.
- [2] Augé, J., Englmeier, K., Hubert, G., Mothe, J., Catégorisation automatique de textes basée sur des hiérarchies de concepts, 19^{ème} Journées de Bases de Données Avancées (BDA), Lyon, p. 69-87, Oct 2003.

- [3] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, Y., Extensible, Markup Language (XML) 1.0. (Third Edition), W3C Recommendation., <http://www.w3.org/TR/REC-xml/>, Feb. 2004.
- [4] Clark, J., DeRose, S., XML Path Language (XPath), W3C Recommendation, <http://www.w3.org/TR/xpath.html>, Nov. 1999.
- [5] Fuhr, N., Maalik, S., Lalmas, M., Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2003, Proceedings of the Second INEX Workshop. Dagstuhl, Germany, March 2004.
- [6] Geva, S., Leo-Spork, M., XPath Inverted File for Information Retrieval, INEX 2003 Workshop Proceedings, pp. 110-117, 2003.
- [7] IRAIA: Getting Orientation in Complex Information Spaces as an Emergent Behaviour of Autonomous Information Agents, European Information Societies Technology, IST-1999-10602, March 2000-2002.
- [8] Kamps, J., de Rijke, M., Sigurbjörnsson, B., Length normalization in XML retrieval. Proceedings of the 27th International Conference on Research and Development in Information Retrieval (SIGIR), pages 80-87. New York NY, USA, 2004
- [9] Li, Y., Toward a qualitative search engine, IEEE Internet Computing, vol. 2, n°4, p. 24-29, 1998.
- [10] List J., Mihajlovic V., de Vries A. P., Ramirez G., Hiemstra D., The TIJAH XML-IR system at INEX 2003, INEX 2003 Workshop Proceedings, pp. 102-109, 2003.
- [11] Ogilvie, P., Callan J., Using Language Models for Flat Text Queries in XML Retrieval, Proceedings of the Second INEX Workshop. Dagstuhl, Germany, March 2004.
- [12] Pauer, B., Holger, P., Statfinder, Document Package Statfinder, Vers. 1.8, may 2000.
- [13] Pehcevski, J., Thom J., Vercoustre, A.M., Enhancing Content-And-Structure Information Retrieval using a Native XML Database", Proceedings of The First Twente Data Management Workshop on XML Databases and Information Retrieval (TDM'04), Enschede, The Netherlands, June 2004
- [14] Piwowski B., Vu H.-T., Gallinari P., Bayesian Networks and INEX'03, Proceedings of the Second INEX Workshop. Dagstuhl, Germany, March 2004.
- [15] Trotman, A., O'Keefe, R. A., Identifying and Ranking Relevant Document Elements(2003), INEX 2003 Workshop Proceedings, pp, 2003.
- [16] Sauvagnat, K., Hubert, G., Boughanem, M., Mothe, J., IRIT at INEX 2003, Proceedings of the Second INEX Workshop. Dagstuhl, Germany, pp 142-148, 2003

The University of Amsterdam at INEX 2004

Börkur Sigurbjörnsson Jaap Kamps* Maarten de Rijke

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
borkur,kamps,mdr@science.uva.nl

ABSTRACT

This paper describes the INEX 2004 participation of the Informatics Institute of the University of Amsterdam. We completely revamped our XML retrieval system, now implemented as a mixture language model on top of a standard search engine. To speed up structural reasoning, we indexed the collection's structure in a separate database. We address three research questions. First, we investigate the effectiveness of blind feedback based on top-ranking XML-elements. Second, we analyze the impact of removing overlapping elements in the result set. Third, for the content-and-structure topics, we want to compare the relative effectiveness of approaches that interpret the topic strict, or ignore the structural hints altogether. Experimental evidence is based on both of the INEX 2004 ad hoc tasks, content-only and content-and-structure, evaluated against a range of metrics.

1. INTRODUCTION

We follow an Information Retrieval (IR) approach to the Content-Only (CO) and Vague-Content-And-Structure (VCAS) ad hoc tasks at INEX. In our participation at INEX 2004 we build on top of our element-based approach at INEX 2003 [10], and extend our language modeling approach to XML retrieval.

Specifically, we addressed the following technological issues, mainly to obtain a statistically more transparent approach. For our INEX 2003 experiments we combined article and element scores outside our language model, meaning that we created a run based on an article index and one based on an element index, which were then combined using well-known run combination techniques [6]. This year we implemented a proper mixture language model for this combination. At INEX 2003 we estimated the language model for the collection by looking at statistics from our overlapping element index. For our experiments at INEX 2004 we estimate this collection model differently, by looking at statistics from our article index. The main changes in our blind feedback approach, compared to last year, is that this year we perform query expansion based on an

*Currently at Archives and Information Studies, Faculty of Humanities, University of Amsterdam.

element run, whereas last year we performed the expansion based on an article run. All our runs were created using the ILPS extension to the Lucene search engine [7, 3].

Our main research questions for both tasks were twofold. First, we wanted to investigate the effect of blind feedback on XML element retrieval. Second, we wanted to cast light on the problem of overlapping results; in particular, we investigate the effect of removing overlapping results top-down from a retrieval run. A third, additional research question only concerns the VCAS task: we investigate the difference between applying a content-only approach and a strict content-and-structure approach.

The remainder of this paper is organized as follows. In Section 2 we describe our experimental setup, and in Section 3 we provide details on the official runs we submitted to INEX 2004. Section 4 presents the results of our experiments, and in Section 5 we discuss our findings in the broader INEX context, and draw some initial conclusions.

2. EXPERIMENTAL SETUP

2.1 Index

Our approach to XML retrieval is IR-based. We create our runs using two types of inverted indexes, one for XML articles only and another for all XML elements. Furthermore, we maintain a separate index of the collection structure.

2.1.1 Article index

For the article index, the indexing unit is a complete XML document containing all the terms appearing at any nesting level within the `<article>` tag. Hence, this is a traditional inverted index as used for standard document retrieval.

2.1.2 Element index

For the element index, the indexing unit can be any XML element (including `<article>`). For each element, all text nested inside it is indexed. Hence, the indexing units overlap (see Figure 1). Text appearing in a particular nested XML element is not only indexed as part of that element, but also as part of all its ancestor elements. The article index can be viewed as a restricted version of the element index, where only elements with tag-name `<article>` are indexed.

Both the article and the element index were word-based: we applied lower-casing and stop-words were removed using the stop-word list that comes with the English version on the Snowball stemmer [12], but other than that words were indexed as they occur in the text, and no stemming was applied.

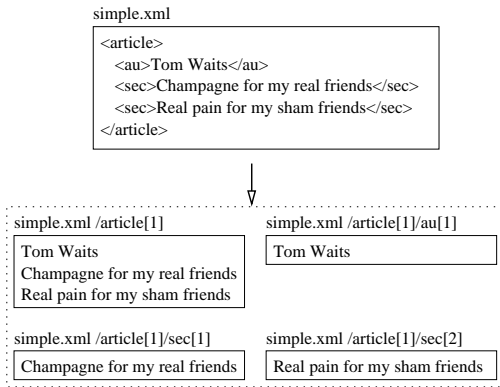


Figure 1: Simplified figure of how XML documents are split up into overlapping indexing units.

2.1.3 Structure index

The structure of the collection is indexed using a relational database. To index the XML trees we use pre-order and post-order information of the nodes in the XML trees [1].

2.2 Query processing

For both the CO and the VCAS task we only use the `<title>` part of the topics. We remove words and phrases bounded by a minus-sign from the queries; other than that, we do not use the plus-signs, or phrase marking of the queries.

For the CAS topics we have a NEXI tokenizer which can decompose the query into a set of about functions [11]. If there is a disjunction in a location-path, we break it up into a disjunction of about functions. That is,

```
about(./(abs|kwd), xml)
```

becomes

```
about(./abs,xml) or about(./kwd,xml).
```

If there are multiple about functions with the same scope we merge them into a single one. That is,

```
about(., broadband) or about(., dial-up)
```

becomes

```
about(., broadband dial-up).
```

For some of the VCAS runs we ignore the structural constraints and use only a collection of content query terms. That is, from the query

```
//article[about(., sorting)]//sec[about(., heap sort)]
```

we collect the query terms

```
sorting heap sort.
```

We will refer to these as the *full content queries*.

2.3 Retrieval model

All our runs use a multinomial language model with Jelinek-Mercer smoothing [2]. We estimate a language model for each of the elements. The elements are then ranked according to their prior probability of being relevant and the likelihood of the query, given the

estimated language model for the element:

$$P(e|q) \propto P(e) \cdot P(q|e). \quad (1)$$

We assume that query terms are independent, and thus we rank our elements according to:

$$P(e|q) \propto P(e) \cdot \prod_{i=1}^k P(t_i|e), \quad (2)$$

where q is a query made out of the terms t_1, \dots, t_k . To account for data sparseness we estimate the element language model by taking a linear interpolation of three language models: one for the element itself, one for the article that contains the element, and a third one for the collection. That is, $P(t_i|e)$ is calculated as

$$\lambda_e \cdot P_{mle}(t_i|e) + \lambda_d \cdot P_{mle}(t_i|d) + (1 - \lambda_e - \lambda_d) \cdot P_{mle}(t_i), \quad (3)$$

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). We estimate the language models, $P_{mle}(\cdot|e)$ and $P_{mle}(\cdot)$, using maximum likelihood estimation. For the element model we use statistics from the element index; for the document model we use statistics from the article index; and for the collection model we use document frequencies from the article index.

The language modeling framework allows us to easily model non-content features. One of the non-content that proved to be useful during our experiments for INEX 2003 is document length. Specifically, we assign a prior probability to an element e relative to its length in the following manner:

$$P(e) = \frac{|e|}{\sum_e |e|}, \quad (4)$$

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

2.4 Query Expansion

We have been experimenting with blind feedback in all editions of INEX so far, focusing on query expansion for the content-only task exclusively. Initially, we experimented with Rocchio-style reweighting to select up to 10 terms from the top 10 documents [9]. In INEX 2002 we observed that query expansion with Rocchio on the article index gave intuitively useful expanded queries, leading to the kind of improvements that familiar from article retrieval [5]. However, expanding queries based on the top 10 retrieved XML elements seemed not to work due to the short and overlapping elements in the top 10 results. Hence, we decided to expand queries on the article index, and then run the expanded queries against the element index. This did, indeed, give us a boost for the 2002 topics, but, alas, substantially lowered our score for the 2003 topics [11].

Our analysis of the failure of article-index based feedback in INEX 2003 was that the terms were useful, but highly unlikely to occur in the proper element. An example is getting prominent author names from the bibliography, which are relevant and useful retrieval cues but generally do not appear in a paragraph (maybe in the author field, or the bibliography).¹

We decided to go back to the idea of doing blind feedback directly on the XML element index. This has the advantage of conser-

¹We have been planning to incorporate context (i.e., tags in which term occurs) into our model, but this would require some CAS features for the CO runs that are non-trivial to implement.

Run-id	λ_e	λ_d	units	terms	% overlap
UAms-CO-T	0.1	0.3	–	–	71.96
UAms-CO-T-FBack	0.1	0.3	15	5	81.85
UAms-CO-T-FBack-NoOverl	0.1	0.3	15	5	0.00

Table 1: Overview of our official content-only runs for INEX 2004. All runs are *automatic* runs, that only use the T (title) topic field.

vatism, the initially retrieved top 10 elements will keep their high ranking, but the problem of overlap in the initial result set remains. In pre-submission experiments, the language modeling approach to feedback [8] proved more robust, and improved performance on the 2003 topics.

3. RUNS

In this section we describe the official runs submitted by the University of Amsterdam for INEX 2004.

All our runs use the language modeling framework described in the previous section. For all runs we use a two level smoothing procedure: we smooth against both the article and the collection. Our collection model uses the document frequencies from the article index. For computing the likelihood of a term given an element, see Equation 3, we use the following parameter settings for all runs: $\lambda_e = 0.1$ and $\lambda_d = 0.3$. All runs also use the same length prior settings in Equation 4.

3.1 Content-Only task

Table 1 provides an overview of our CO runs. We now describe the specifics of each of the CO runs.

UAms-CO-T

This run uses the mixture language model approach and parameter settings as described above.

UAms-CO-T-FBack

This run uses the same model and parameters as the previous run. Additionally, this run uses blind feedback to expand the queries. An element run was used as a basis for our feedback. We considered the top 15 elements to be relevant and chose the 5 best query terms as described in [8].

UAms-CO-T-FBack-NoOverl

This run uses the same model, parameters and feedback approach as the previous run. Additionally, overlapping results are filtered away. The filtering is done in a top-down manner. That is, the result list is processed from the most relevant to the least relevant element. A result is removed from the result list if it overlaps with an element that has been processed previously.

3.2 Vague Content-And-Structure task

We now describe our VCAS runs; again, we provide a table with an overview; cf. Table 2.

UAms-CAS-T-FBack

This run uses the full-content version of the queries. The run is identical to UAms-CO-T-FBack, except for the topics, of course.

Run-id	λ_e	λ_d	units	terms	% overlap
UAms-CAS-T-FBack	0.1	0.3	15	5	77.76
UAms-CAS-T-FBack-NoOverl	0.1	0.3	15	5	0.00
UAms-CAS-T-XPath	–	–	–	–	18.77

Table 2: Overview of our official vague content-and-structure runs for INEX 2004. All runs are *automatic* runs, that only use the T (title) topic field.

UAms-CAS-T-FBack-NoOverl

This run uses the full-content version of the queries. The run is identical to UAms-CO-T-FBack-NoOverl, except for the topics, of course.

UAms-CAS-T-XPath

This run is created using our system for the INEX 2003 Strict Content and Structure task. It uses both content and structural constraints. Target constraints are interpreted as strict. We refer to [11] for a detailed description of the retrieval approach used. The run is identical to the run referred to as “Full propagation run” in that paper.

4. RESULTS

In this section we will try to analyze the results of our retrieval efforts. Result analysis for XML retrieval remains a difficult task: there are still many open questions regarding how to evaluate XML element retrieval. We will show our results for all the suggested measures and try to interpret the flow of numbers.

4.1 Content-Only task

Table 3 shows the results for our CO runs, using all different metrics. We see that the run which uses blind feedback outperforms the normal run on all metrics except for the measures where high exhaustivity is rewarded. Hence, at first glance, it seems thus that blind feedback does more for the specificity of the results than for the exhaustiveness of the results. This seems somewhat counterintuitive and we will discuss it further below. The run where overlap was removed does not score well for any metric.

Figure 2 shows precision-recall curves for our CO runs for three measures: strict, generalized measure, specificity oriented (so). Query expansion gives improvements on all recall levels. The normal and non-overlapping runs have similar precision at zero, but the non-overlapping run quickly drops. The non-overlapping run simply fails to retrieve many of the relevant elements. This comes as no surprise since the relevant elements, themselves, are frequently overlapping.

Measure	Run		
	CO-T	CO-T-FBack	CO-T-FBack-NoOverl
aggregate	0.1030	<i>0.1174</i>	0.0270
strict	0.1013	<i>0.1100</i>	0.0332
generalized	0.0929	<i>0.1225</i>	0.0198
so	0.0717	<i>0.1060</i>	0.0149
s3_e321	0.0528	<i>0.0877</i>	0.0148
s3_e32	0.0668	<i>0.0891</i>	0.0168
e3_s321	<i>0.1840</i>	0.1699	0.0507
e3_s32	<i>0.1515</i>	0.1368	0.0387

Table 3: Average scores for our CO runs, with this best scoring run in italics.

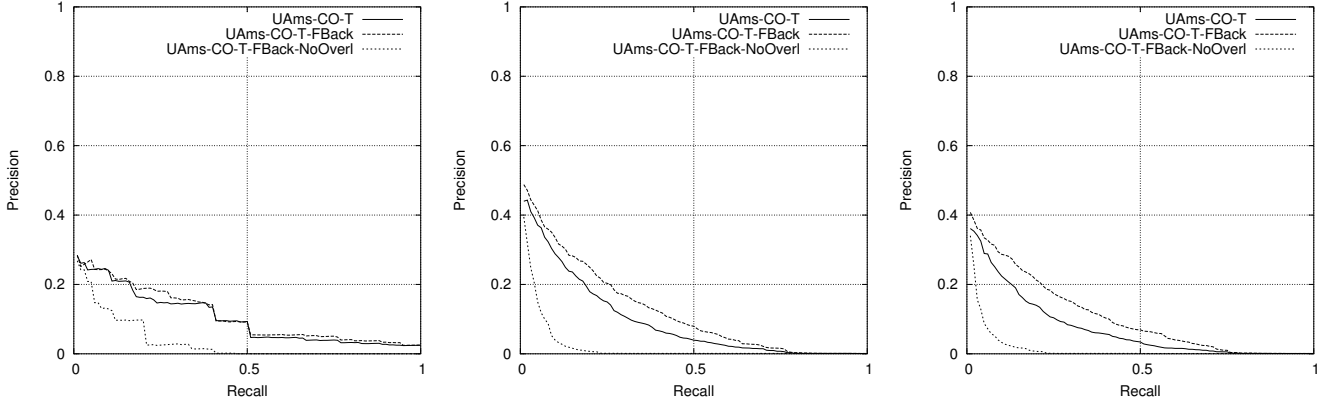


Figure 2: Precision-recall curves for our CO runs. (Left): strict measure. (Center): generalized measure. (Right): specificity oriented (so) measure.

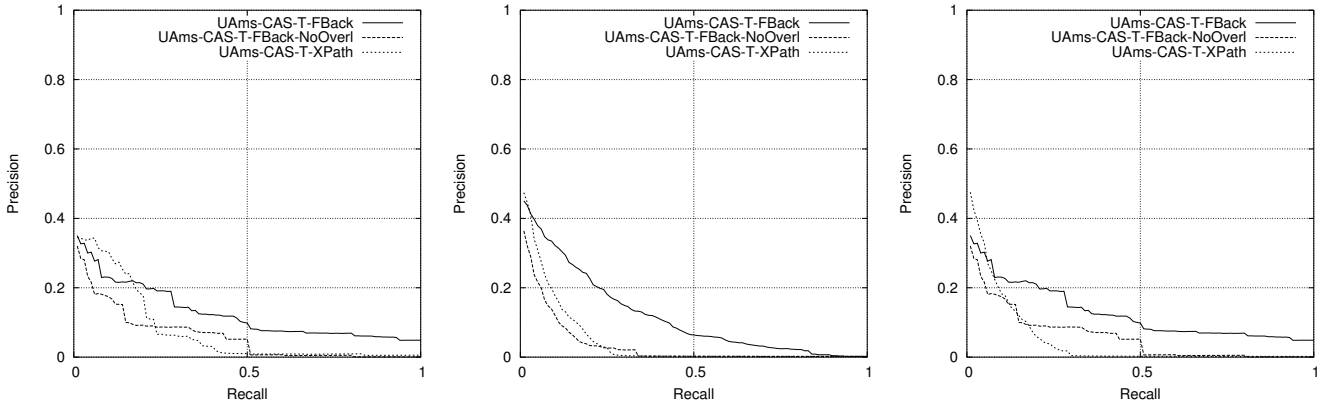


Figure 3: Precision-recall curves for our VCAS runs. (Left): strict measure. (Center): generalized measure. (Right): specificity oriented (so) measure.

4.2 Vague Content-And-Structure task

Table 4 shows the results for our VCAS runs, using all the different metrics. We see that the CO-style run clearly outperforms the XPath-style run with respect to all metrics. Again, the run without overlap scores the least of the three.

Figure 3 shows precision-recall curves for our VCAS runs, again for three measures: strict, generalized measure, and specificity oriented (so). The XPath-style run, tailored for a strict interpretation

Measure	Run (CAS-T-...)		
	...FBack	...FBack-NoOverl	...XPath
aggregate	<i>0.1065</i>	0.0397	0.0619
strict	<i>0.1260</i>	0.0582	0.0735
generalized	<i>0.1167</i>	0.0330	0.0451
so	<i>0.0912</i>	0.0282	0.0472
s3_e321	<i>0.0770</i>	0.0318	0.0537
s3_e32	<i>0.0817</i>	0.0365	0.0781
e3_s321	<i>0.1508</i>	0.0495	0.0581
e3_s32	<i>0.1020</i>	0.0404	0.0774

Table 4: Average scores for our VCAS runs, with the best scoring run in *italics*.

of content-and-structure topics, seems to function as a precision device. The run outperforms the CO-style run at lower recall levels. The low scores on higher recall level can immediately be explained by the fact that the target element is respected in the XPath-style run, but not in the relevance judgments.

5. DISCUSSION AND CONCLUSIONS

In this paper, we documented our experiments at the INEX 2004 ad hoc retrieval track. We addressed three main research questions. First, we investigated the effectiveness of element-based query expansion, and found that it improved retrieval effectiveness on all but the exhaustiveness-oriented measures. We will discuss this case below. Second, we investigated the impact of (non-)overlap on the runs, and found that returning overlapping results results in superior scores on all measures. Our non-overlapping runs were, indeed, completely non-overlapping. Perhaps this is an unrealistically strong requirement, for it proves difficult to predict the choices of the assessors, and many relevant elements will be removed from the ranking. On a more positive note, the XPath-style run for SCAS had only 19% overlap, and got the best score at low recall levels. Third, our results for the VCAS task showed clear superiority of content-oriented-based approaches over a strict interpretation of the content-and-structure topics. From the vantage point of a retrieval system, our experiments highlighted the great

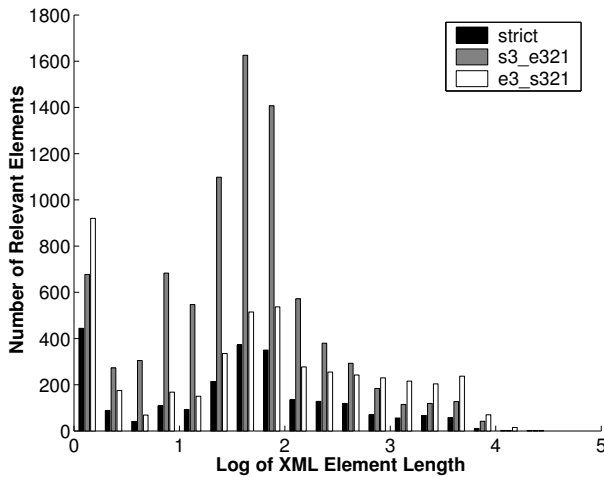


Figure 4: Length of relevant elements for strict, specificity-oriented, and exhaustiveness oriented measures in INEX 2004.

similarity between the CO and VCAS tasks. The most notable difference, perhaps, is the fact that the XPath-style run can function as a precision device.

Previously, we have shown that a more radical length bias is essential to achieve good results [4]. Those experiments were performed using both the title and description fields of the topics. In the language modeling framework, as shown in Section 2.3, the final score of an element is the product of the prior probability of an element and the likelihood of the query given an element. However, the length of a query does have an effect on the number calculated for the query-likelihood. As a result, the normal length bias has a bigger impact on the shorter queries. Initial pre-submission experiments for the title-only topics showed the normal length-prior settings in Equation 4 in Section 2.3 to be sufficient. We did use blind feedback to expand queries with up to 5 terms. This will result again in longer queries, and perhaps may suggest that these are similar to the longer TD-topics. This is true in part, but there is an important difference between the TD-topics and (expanded) T-topics: all keywords from the title are content-bearing words specific for the query, as are supposedly the expanded terms. This may also be a factor that lessens the need for the extreme length-priors shown to be crucial for TD-topics [4].

We now return to the finding that query expansion does not help on the exhaustiveness-oriented measures, i.e., `e3_s32` and `e3_s321`. One would expect that strict expansion of the query with useful terms (witnessing the other measures), leads to improvement of recall, and therefore would help exhaustiveness rather than specificity. In contrast, we see improvements on all measures *but* the exhaustiveness-oriented ones. This is clearly counterintuitive. Our best explanation to date has to do with the changing recall base of the measures. In Figure 4 we plot the (log of) element length against the number of relevant elements, where relevancy is determined by one of three measures: strict, specificity, and exhaustiveness. As can be seen from the plot, the strict and specificity measure return different counts but a very similar distribution over length. The exhaustiveness measure, in contrast, has a preference for much larger elements. This is not unexpected: if we stress the exhaustiveness dimension, we would generally expect to find larger chunks of text containing more information. As a result, our nor-

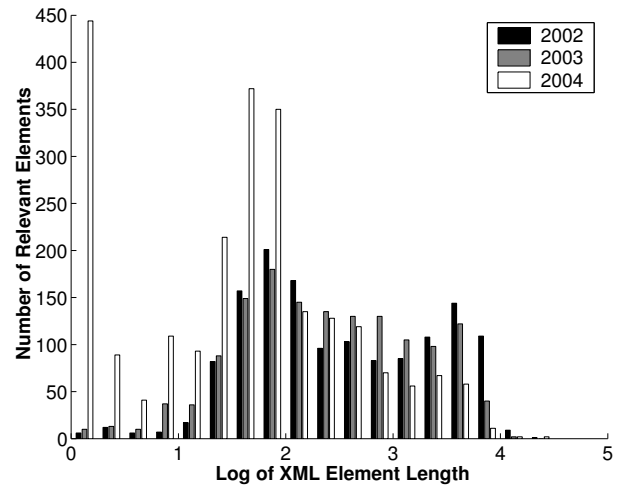


Figure 5: Length of relevant elements in INEX 2002–2004, measured using the strict measure.

mal length prior is clearly insufficient to satisfy the exhaustiveness measure. The normal length prior creates more bias for the shorter unexpanded queries. Thus, for runs with a larger length bias may still show improvements for the expanded queries.

Figure 5 presents the distribution of the length of relevant XML elements over the three years of INEX CO, where relevancy is measured using the strict measure. While one has to be careful in making performance and test set comparisons across years, the following observations seem legit.

First, over the three years there is an declining preference for the larger elements such as full articles. In the first edition of INEX, i.e., in 2002, assessors frequently judged the larger elements relevant. In 2003, there was less of a preference for large elements, and in 2004 trend seems to persist: an even smaller fraction of the longer elements were judged relevant. With exception of the (almost) empty elements, the distribution of elements is qualitatively not very different from earlier years. Second, there is an amazing number of very small elements, ranging from empty to just one or a few words, that is judged as relevant. This raises a number of questions regarding the INEX relevance assessment stage. We find it implausible that an element with no or just one or two words can completely satisfy the information need of the topic (i.e., be judged as highly exhaustive and highly specific).

6. ACKNOWLEDGMENTS

Jaap Kamps was supported by the Netherlands Organization for Scientific Research (NWO), under project number 612.066.302. Maarten de Rijke was supported by grants from NWO, under project numbers 365-20-005, 612.069.006, 612.000.106, 220-80-001, 612.-000.207, 612.066.302, 264-70-050, and 017.001.190.

7. REFERENCES

- [1] T. Grust. Accelerating XPath Location Steps. In *Proc. SIGMOD*, pages 109–120. ACM Press, 2002.
- [2] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, 2001.
- [3] ILPS. The ILPS extension of the Lucene search engine,

2004. <http://ilps.science.uva.nl/Resources/>.

- [4] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length normalization in XML retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (SIGIR 2004)*, pages 80–87, 2004.
- [5] J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. The importance of morphological normalization for XML retrieval. In *Proceedings of the First Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, pages 41–48. ERCIM Publications, 2003.
- [6] J. Kamps and M. De Rijke. The effectiveness of combining information retrieval strategies for European languages. In *Proceedings 19th Annual ACM Symposium on Applied Computing*, pages 1073–1077, 2004.
- [7] Lucene. The Lucene search engine, 2004. <http://jakarta.apache.org/lucene/>.
- [8] J. Ponte. Language models for relevance feedback. In W.B. Croft, editor, *Advances in Information Retrieval*, chapter 3, pages 73–96. Kluwer Academic Publishers, Boston, 2000.
- [9] J. J. Rocchio, Jr. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice-Hall, Englewood Cliffs NJ, 1971.
- [10] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-Based Approach to XML Retrieval. In *INEX 2003 Workshop Proceedings*, pages 19–26, 2004.
- [11] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Processing content-oriented XPath queries. In *Proceedings of the Thirteenth Conference on Information and Knowledge Management (CIKM 2004)*, pages 371–380. ACM Press, 2004.
- [12] Snowball. The Snowball string processing language, 2004. <http://snowball.tartarus.org/>.

GPX - Gardens Point XML Information Retrieval at INEX 2004

Shlomo Geva

Centre for Information Technology Innovation
Faculty of Information Technology
Queensland University of Technology
Queensland 4001 Australia

s.geva@qut.edu.au

Abstract *Traditional information retrieval (IR) systems respond to user queries with ranked lists of relevant documents. The separation of content and structure in XML documents allows individual XML elements to be selected in isolation. Thus, users expect XML-IR systems to return highly relevant results that are more precise than entire documents. In this paper we describe the implementation of a search engine for XML document collections. The system is keyword based and is built upon an XML inverted file system. We describe the approach that was adopted to meet the requirements of Content Only (CO) and Vague Content and Structure (VCAS) queries in INEX 2004.*

Keywords Information Retrieval, XML, Search Engine, Inverted Files

1.0 Introduction

The widespread use of Extensible Markup Language (XML) documents in digital libraries has lead to development of information retrieval (IR) methods specifically designed for XML collections. Most traditional IR systems are limited to whole document retrieval; however, since XML documents separate content and structure, XML-IR systems are able to retrieve the relevant portions of documents. This means that users interacting with XML-IR system will potentially receive highly relevant and highly precise material. However, it also means that XML-IR systems are more complex than their traditional counterparts, and many challenges remain unsolved. These issues were specifically addressed at INEX 2002 and INEX 2003, with marked improvement in performance of most systems.

Since all systems base retrieval on keywords, one would expect that most system would be able to identify the same set of

documents in response to a query on several keywords. The key difference between systems is therefore in the ranking of these documents. Often it is possible to identify many thousands of elements with a given set of keywords. The trick is to rank the elements and select the top 1500 elements. There are four important issues to be resolved with respect to IR in an XML collection:

1. Accurate and efficient selection of elements that satisfy the containment constraints.
2. Accurate and efficient selection of elements that satisfy the structural constraints.
3. The assignment of scores to matching elements.
4. The assignment of scores to antecedents of selected elements

Steps 1 to 3 are common to ordinary text collection IR systems, except that the unit of retrieval has finer granularity (XML element). Step 4 is an additional step that is required in the context of XML oriented IR. Rather than identify relevant documents, the XML IR system is required to select and score elements at different levels of granularity.

This paper presents a system that attempts to provide a solution to the selection and ranking question, while at the same time provide an effective and efficient search engine that is based on an inverted file scheme.

First we discuss the internal storage of the XML collection and present a database structure that is aimed at increasing the efficiency of the system. Then we discuss a reformulation of the queries as a set of sub-queries, and describe the identification of leaf level elements that contain keywords. We describe the retrieval and scoring process of individual elements. We then discuss the ranking scheme that is used to propagate scores to antecedent

elements at coarser granularity. Finally we present benchmark test results using the INEX 2003 XML collection, and the 2004 official results assessments.

2.0 Query Interpretation

INEX provides a test collection of over 12,000 IEEE journal articles, a set of queries and a set of evaluation metrics. Two types of queries are used in INEX 2004: CO and VCAS. Content Only (CO) queries ignore document structure and only contain content stipulations. In contrast Vague Content and Structure (VCAS) queries explicitly express both content and structural requirements. Both CO and CAS queries are expected to return appropriately sized elements – not just whole documents, and all queries are loosely interpreted with respect to structural and containment constraints – the overriding goal is to satisfy the user’s information need rather than the strict query formulation. Figures 1 and 2 are examples of both query types.

```
<inex_topic topic_id="XX" query_type="CO">
<title>
  "multi layer perceptron" "radial basis
  functions" comparison
</title>
<description>
  The relationship and comparisons between
  radial basis functions and multi layer
  perceptrons
</description>
</inex_topic>
```

Figure 1: CO Query

```
<inex_topic topic_id="XX"
query_type="CAS">
<title>
  //article[about(.,information
  retrieval)]//sec[about(.,compression)]
</title>
<description>
  Find sections about compression in
  articles about information retrieval.
</description>
</inex_topic>
```

Figure 2: CAS Query

Both the *description* and *title* elements express the user’s information needs. The description expresses users’ need in a natural language (e.g. English). The title expresses users’ information need in either a list of keywords/phrases (CO) or

as a formal XPath-like language (CAS) called Narrowed Extended XPath I (NEXI) [4].

The syntax of NEXI is similar to XPath, however, it NEXI only uses XPath’s descendant axis step, and extends XPath by incorporating an ‘about’ clause to provide an IR-like query. NEXI’s syntax is `//A[about(//B,C)]` where **A** is the context path, **B** is the relative path and **C** is the content requirement. Conceptually each ‘about’ clause in a NEXI query represents an individual information request. So conceptually the query

```
//A[about(//B,C)]//X[about(//Y,Z)]
```

contains two requests:

```
//A[about(//B,C)]
```

and

```
//A//X[about(//Y,Z)].
```

However, in NEXI only elements matching the leaf (i.e. rightmost) ‘about’ clause, the second request here, are flagged as of direct interest to the user. We refer to these requests and elements as ‘return requests’ and ‘return elements’. Elements that match the other ‘about’, clauses, the first request here, are used to support the return elements in ranking. We refer to these requests and elements as ‘support requests’ and ‘support elements’. It should be noted that under VCAS rules, the vague interpretation of queries allows the return of elements whose XPath signature does not strictly conform to the query specification. The structural constraints are regarded as retrieval hints, much in the same way that keywords are regarded as retrieval hints.

2.2 Processing NEXI Queries

Once NEXI queries are input into the system they are converted into an intermediate language called the RS query language. The RS query language converts NEXI queries to a set of information requests:

Instruction | Retrieve_Filter | Search_Filter | Content

Instruction: Either ‘R’ or ‘S’, corresponding to ‘return’ or ‘support’ component.

Retrieve_Filter: A logical XPath expression that describes which elements should be retrieved by the system. Often this correlates to context path

of a NEXI query, so, in the query `//A[about(//B,C)]` the retrieve filter is `//A`.

Search_Filter: A logical XPath expression that describes which elements should be searched by the system. Often this correlates to relative path of a NEXI query, so, in the query `//A[about(//B,C)]` the search filter is `//A//B`.

Figure 3 is an example of the queries introduced in Figure 1 and Figure, 2 when converted to RS queries.

<p>RS Query 1: <code>R//*/**/*</code> relationship, comparisons, radial basis functions, multi layer perceptions</p> <p>RS Query 2: <code>R//article//sec//article//seccompression</code> <code>S//article//article</code> information retrieval</p>

Figure 3: Examples of RS Queries

3.0 XML File Inversion

In our scheme each term in an XML document is identified by 3 elements. File path, absolute XPath context, and term position within the XPath context.

The file path identifies documents in the collection; for instance:

`C:/INEX/ex/2001/x0321.xml`

The absolute XPath expression identifies a leaf XML element within the document, relative to the file's root element:

`/article[1]/bdy[1]/sec[5]/p[3]`

Finally, term position identifies the ordinal position of the term within the XPath context.

One additional modification that we adopted allowed us to support queries on XML tag attributes. This is not a strictly content search feature, but rather structure oriented search feature. For instance, it allows us to query on the 2nd named author of an article by imposing the additional query constraint of looking for that qualification in the attribute element of the XML author element. The representation of attribute values is similar to normal text with a minor modification to the XPath context representation – the attribute name is appended to the absolute XPath expression. For instance:

`article[1]/bdy[1]/sec[6]/p[6]/ref[1]/@rid[1]`

Here the character '@' is used to flag the fact that "rid" is not an XML tag, but rather an attribute of the preceding tag <ref>. An inverted list for a given term, omitting the File path and the Term position, is depicted in figure 4.

In principle at least, a single table can hold the entire cross reference list (our inverted file). Suitable indexing of terms can support fast retrieval of term inverted lists. However, it is evident that there is extreme redundancy in the specification of partial absolute XPath expressions (substrings). There is also extreme redundancy in full absolute XPath expressions where multiple terms in the same document share the same leaf context (e.g. all terms in a paragraph). Furthermore, many XPath leaf contexts exist in almost every document (e.g. `/article[1]/fm[1]/abs[1]`).

We have chosen to work with certain imposed constraints. Specifically, we aimed at implementing the system on a PC and base it on the Microsoft Access database engine. This is a widely available off-the-shelf system and would allow the system to be used on virtually any PC running under any variant of the standard Microsoft Windows operating system. This choice implied a strict constraint on the size of the database – the total size of an Access database is limited to 2Gbyte. This constraint implied that a flat list structure was infeasible and we had to normalise the inverted list table to reduce redundancy.

Context
XPath
<code>article[1]/bdy[1]/sec[6]/p[6]/ref[1]</code>
<code>article[1]/bdy[1]/sec[6]/p[6]/ref[1]/@rid[1]</code>
<code>article[1]/bdy[1]/sec[6]/p[6]/ref[1]/@type[1]</code>
<code>article[1]/bm[1]/bib[1]/bibl[1]/bb[13]/pp[1]</code>
<code>article[1]/bm[1]/bib[1]/bibl[1]/bb[14]/pdt[1]/day[1]</code>
<code>article[1]/bm[1]/bib[1]/bibl[1]/bb[14]/pp[1]</code>
<code>article[1]/bm[1]/bib[1]/bibl[1]/bb[15]</code>
<code>article[1]/bm[1]/bib[1]/bibl[1]/bb[15]/@id[1]</code>
<code>article[1]/bm[1]/bib[1]/bibl[1]/bb[15]/ti[1]</code>
<code>article[1]/bm[1]/bib[1]/bibl[1]/bb[15]/obi[1]</code>

Figure 4: XPath inverted list

The structure of the database used to store the inverted lists is depicted in Figure 5. It consists

of 4 tables. The *Terms* table is the starting point of a query on a given term. Two columns in this table are indexed - The *Term* column and the *Term_Stem* column. The *Term_Stem* column holds the Porter stem of the original term. The *List_Position* is a foreign key from the *Terms* table into the *List* Table. It identifies the starting position in the inverted list for the corresponding term. The *List_Length* is the number of list entries corresponding to that term. The *List* table is (transparently) sorted by Term so that the inverted list for any given term is contiguous. As an aside, the maintenance of a sorted list in a dynamic database poses some problems, but these are not as serious as might seem at first, and although we have solved the problem it is outside the scope of this paper and is not discussed any further.

A search proceeds as follows. Given a search term we obtain a starting position within the List table. We then retrieve the specified number of entries by reading sequentially. The inverted list thus obtained is *Joined* (SQL) with the *Document* and *Context* tables to obtain the complete de-normalised inverted list for the term. The retrieval by *Term_Stem* is similar. First we obtain the Porter stem of the search term.

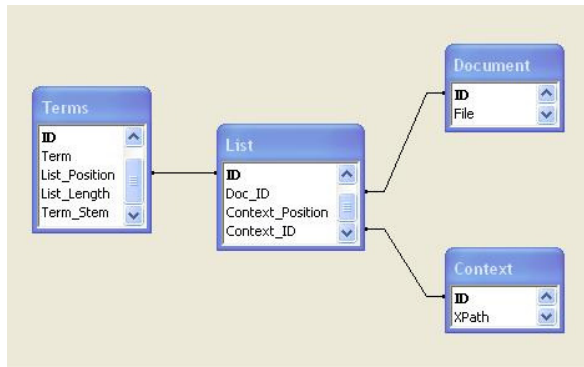


Figure 5: Schema for XML Inverted File.

Then we search the list by *Term_Stem* – usually getting duplicate matches. All the lists for the duplicate hits on the *Terms* table are then concatenated. Phrases and other proximity constraints can be easily evaluated by using the *Context_Position* of individual terms in the *List* table. With this normalization the database size was reduced to 1.6GByte and within the Microsoft Access limits.

4.0 Ranking Scheme

Elements are ranked according to a relevance judgement 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 judged relevant if it contains at least one query term. A branch node is judged relevant if it contains a relevant child element. Once an element (either leaf or bunch) is judged relevant its relevancy judgement score is calculated. A heuristically derived formula (Equation 1) is used to calculate the relevance judgment 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 penalises elements with frequently occurring query terms (frequent in the collection), and it rewards elements with evenly distributed query term frequencies within the elements.

Equation 1: Calculation of a Leaf Element's Relevance Judgement Score

$$L = N^{n-1} \sum_{i=1}^n \frac{t_i}{f_i}$$

Here n is the number of unique query terms contained within the leaf element, N is a small integer (we used $N=5$). The term N^{n-1} scales up the score of elements having multiple distinct query terms. The system is not sensitive to the value of N – we experimented with $N=3$ to 10 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 judgment 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, these 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 judgement 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 Judgement Score

$$R = D(n) \sum_{i=1}^n L_i$$

Where:

n = the number of children elements

$D(n) = 0.49$ if $n = 1$

0.99 Otherwise

L_i = the i^{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, structural constraints are only loosely interpreted. So elements are collected regardless of compliance with the structural stipulations of the topic. So in the example above, ancestors or descendants of **Y** may be returned, depending on their score and final rank.

5.0 Assessment against 2003 data

We conducted experiments against the INEX 2003 query set and evaluation metrics. The results from 2003 INEX queries were submitted into the official INEX evaluation program that calculated the recall/precision graphs. We tested the system against the SCAS and against the CO

data. It should be noted that in the case of SCAS, only results that strictly satisfy the structural constraints are permissible. Therefore, the test against the SCAS set was only useful in assessing the utility of the search engine in identifying return elements, but not in testing the selection of antecedent elements. The CO data however allowed us to test this approach.

The results from the INEX 2003 data were encouraging. We were able to obtain results that were better than the best run in both the CO and the SCAS track under the strict metric, and very close to the best results under the generalised metric. Figures 6 to 9 depict the retrieval results of our unofficial test runs against the 2003 data.

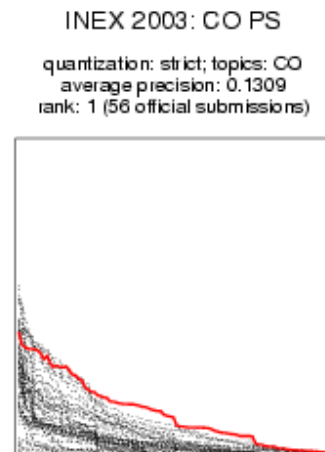


Figure 6: CO run evaluation

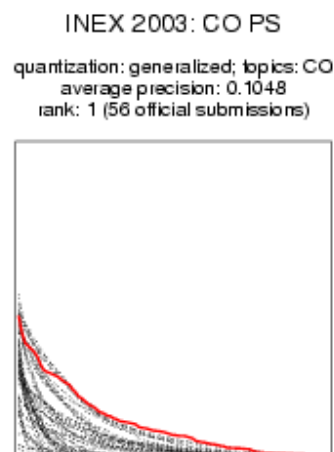


Figure 7: CO run evaluation

INEX 2003: SCAS PS

quantization: strict; topics: SCAS
average precision: 0.2790
rank: 3 (38 official submissions)

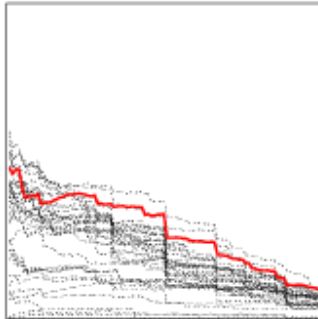


Figure 8: SCAS run evaluation

INEX 2003: SCAS PS

quantization: generalized; topics: SCAS
average precision: 0.2419
rank: 4 (38 official submissions)

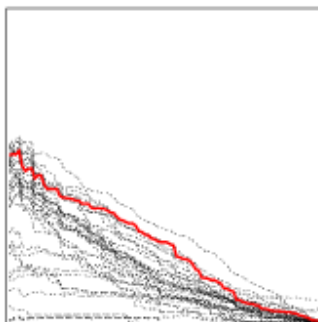


Figure 9: SCAS run evaluation

6.0 Assessment against 2004 data

The system was implemented in C# and run under Windows XP. We tested the system in both CO and VCAS tasks. The entire sets of queries from each of the tasks were evaluated in about 30 minutes on a Pentium M 1.6 MHz processor with 1GB RAM. We have submitted 3 official runs in each of the tracks. We have used the same 3 run strategies in both tasks as follows. The basic run was as described in section 4. A second run eliminated from the search any keyword with a frequency greater than 50,000 in the INEX collection. This almost halved the time it took to evaluate all the queries. Our results from tests on the 2003 data indicated that this would have little impact on the final result and indeed this is confirmed by the 2004 results

(see figures 10 and 11). The third run was used to test a slightly different ranking approach. We have changed the decay factor in Equation 2 as follows:

$$D(n) = \begin{cases} 0.25 & \text{if } n = 1 \\ 0.49 & \text{otherwise} \end{cases}$$

This means that more specific elements are preferred. It takes the accumulation of several children before a parent element accumulates a score higher than its children scores. Our experiments against 2003 data revealed that this strategy worked best against metrics that preferred specificity over exhaustivity. We have not used stemming, but used plural/singular expansion of search terms.

The results of the INEX 2004 benchmark are depicted in figures 10 to 12 (downloaded from LIP6 web site of official INEX 2004 metrics.)

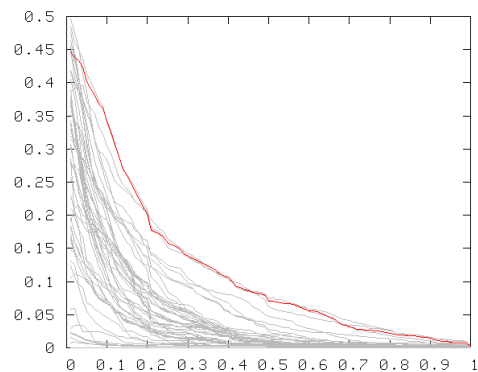


Figure 10: VCAS evaluation, stopping words with frequency greater than 50,000 in the collection (rank: 1)

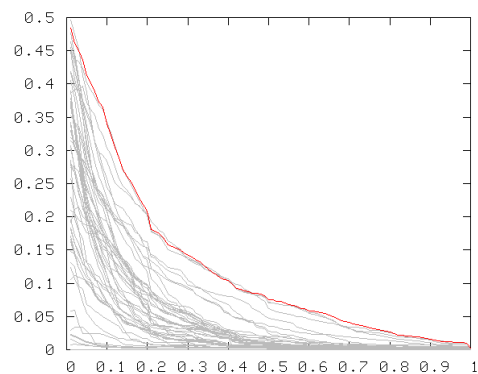


Figure 11: VCAS run evaluation, standard configuration (rank: 2)

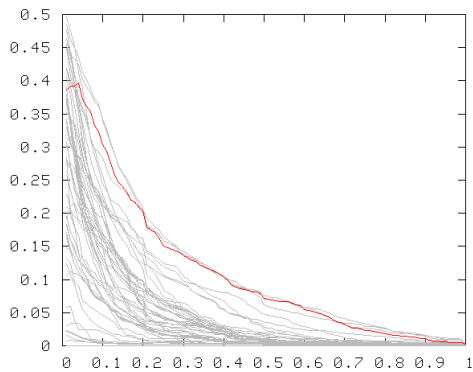


Figure 12: VCAS run evaluation, stopping words with frequency greater than 50,000, Decay factor 0.49 / 0.25 (rank: 3)

It can be seen that it did not make a great difference to the overall ranking whether stopping was used, or whether we varied the decay factor. This is a comforting result because it indicates that the approach is not sensitive to the “magic numbers” that we used in Equations 1 and 2. The best result was obtained when stopping keywords at frequency above 50K and using the standard ranking strategy (equations 1 and 2).

The results for the CO track are depicted in figures 13 to 15, and we observe the same pattern.

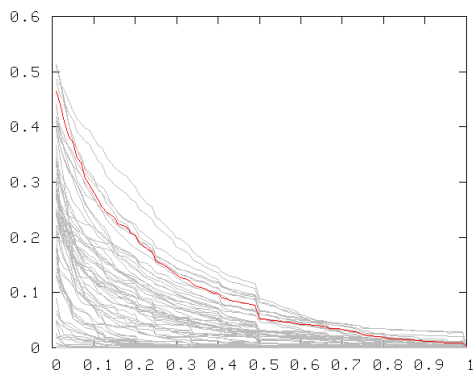


Figure 13: CO run evaluation standard configuration (rank: 7)

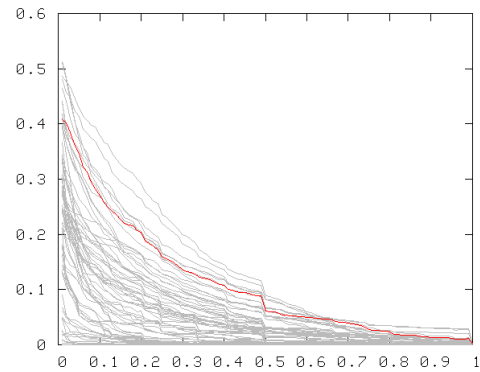


Figure 14: CO run evaluation, stopping words with frequency greater than 50,000 in the INEX collection (rank: 6)

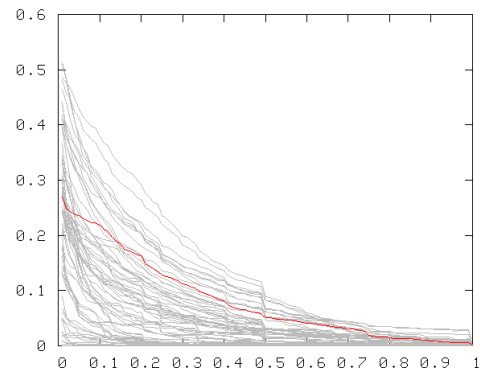


Figure 15: CO run evaluation, stopping words with frequency greater than 50,000, Decay factor 0.49 or 0.25 (rank: 12)

7.0 Unofficial submissions

Several other unofficial submissions were used to test additional techniques. We have tested the utility of blind feedback and of evaluating VCAS topics as if they were CO topics.

7.1 Blind Feedback

Blind feedback was performed as follows. First a retrieval run was performed. Then the top 10 result elements were extracted. All the words in these elements were collected and stop words eliminated (words occurring more than 50,000 times in the collection). Words occurring in less than 20% of leaf elements in the results were then eliminated. The remaining words were then sorted by frequency count and the 5 most frequent terms (at most) added to the topic. Then a second retrieval run was performed. As usual,

the idea is to identify words that occur frequently in highly ranked elements. The results of this experiment are depicted in figure 16.

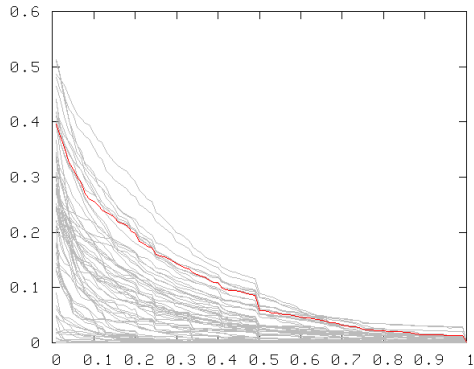


Figure 16: CO topics evaluated with blind feedback using the top 10 elements (unofficial rank: 8)

The results are similar to those obtained without blind feedback. We experimented with various parameters on 2003 data, but were unable to discover an advantageous configuration.

7.2 Evaluating VCAS as CO

CAS queries contain both structural and containment constraints. We have tested the significance of the structural constraints by transforming the VCAS queries into CO queries. This is easily done by collecting all the keywords from all the about clauses to form a CO title elements for the topic. One would expect degradation in performance, but the question is to what extent performance will degrade. Figure 17 depicts the performance of a VCAS as CO evaluation.

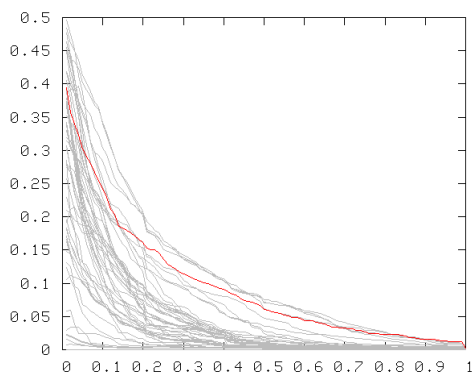


Figure 17: VCAS evaluated as CO by removing structural constraints (unofficial rank 5th)

The performance of the search engine with this approach is surprisingly good. The average

precision is 0.09, which places the submission in 5th position overall. It seems to suggest that at least with the current INEX collection there is relatively little advantage to specifying structural constraints.

8.0 Conclusion and Future Outlook

This paper presents an XML IR system responds to user queries with relevant and appropriately sized results in a timely manner. The approach is based on simple inverted lists indexing and on simple heuristics in ranking. Despite its simplicity our system produces results that are comparable with the best alternatives at INEX.

Future work will concentrate on improving the ranking strategy, improving the blind feedback implementation, the use of ontology for query expansion, and on natural language processing.

The system had been implemented as a massively distributed IR system and we are looking forward to an opportunity to evaluate it against a Terabyte size collection.

References

- [1] N. Fuhr and S. Malik. Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2003. In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17, 2003, pages 1-11. 2004.
- [2] B. Sigurbjörnsson, J. Kamps, M. de Rijke, *An Element-based Approach to XML Retrieval*, In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17, 2003, pages 19-26, 2004.
- [3] Trotman, A. and O'Keefe, "The Simplest Query Language That Could Possibly Work", In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17, 2003, pages 167-174, 2004.
- [4] A. Trotman and B. Sigurbjörnsson, *Narrowed Extended XPath I (NEXI)*, <http://www.cs.otago.ac.nz/postgrads/andrew/2004-4.pdf>, 2004.
- [5] R. J. Van Rijsbergen, R. J., *Information Retrieval*, Butterworths, Second Edition, 1979.

Hierarchical Language Models for XML Component 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

Experiments using hierarchical language models for XML component retrieval are presented in this paper. The role of context is investigated through incorporation of the parent’s model. We find that context can improve the effectiveness of finding relevant components slightly. Additionally, biasing the results toward long components through the use of component priors improves exhaustivity but harms specificity, so care must be taken to find an appropriate trade-off.

1. INTRODUCTION

Language modeling approaches have been applied successfully to retrieval of XML components in previous INEX evaluations [7][12][5][9][10]. In [9] and [10], the authors presented a hierarchical language model for retrieval of XML components. These works proposed that each document component be modeled by a language model estimated using evidence in the node and its children nodes. The work here extends the model to include the parent node’s model in estimation, which allows for some context and is called shrinkage.

New experiments using this model are presented that examine the role of shrinkage introduced in this work and the use of the prior probabilities popularized by [5] for the evaluation of Content-Only queries. Our experiments show that shrinkage provides a modest boost in performance. Prior probabilities can have a strong effect in biasing results, particularly in improving exhaustivity (finding all relevant text) while at the same time harming specificity (finding the best component within the hierarchy). A prior based on the square of the length of text contained in a component and its children was found to be most effective.

Section 2 presents the model of documents and Section 3.1 describes how document components are ranked. Experimental methodology and results are presented in Sections 4

and 5. Related work is discussed in Section 6 and Section 7 concludes the paper.

2. MODELING DOCUMENTS WITH HIERARCHICAL STRUCTURE

Hierarchically structured documents may be represented as a tree, where nodes in the tree correspond to components of the document. From the content of the document, a generative distribution may be estimated for each node in the tree. The distribution at a node may be estimated using evidence from the text of the node, the children’s distributions, or the parent’s distribution, and so on.

Representing hierarchically structured documents in this manner is simple and flexible. In this approach we combine the evidence from the various components in the document using the document structure as guidance. This model uses linear interpolation to combine the evidence from the component, its children components, and its parent component. The model below is similar to previous work by the authors [9][10], but is extended to allow for the inclusion of a component’s context within the document.

More formally, the hierarchical structure of the document is represented by a tree, each vertex $v \in \mathcal{V}$ in the tree corresponding to a document component. Directed edges in the tree are represented as a list of vertex pairs $(v_i, v_j) \in \mathcal{E}$ when v_i is the parent of v_j . *Parent*, *children* and *descendants* functions may be defined as:

$$\begin{aligned} \text{parent}(v_j) &= v_i : (v_i, v_j) \in \mathcal{E} \\ \text{children}(v_i) &= \{v_j : (v_i, v_j) \in \mathcal{E}\} \\ \text{descendants}(v_i) &= \left\{ \begin{array}{l} v_j : v_j \in \text{children}(v_i) \text{ or} \\ \exists v_k \in \text{children}(v_i) \\ \text{s.t. } v_j \in \text{descendants}(v_k) \end{array} \right\} \end{aligned}$$

As stated above, the generative model for a component may be estimated using a linear interpolation of the model estimated directly from the component, its children’s models, and its parent model. Estimation of the generative models for the components of a document is a three step process. First, a smoothed generative model is θ_{v_i} estimated from the observations of the component in the document v_i that

does not include evidence of children components:

$$P(w|\theta_{v_i}) = (1 - \lambda_{v_i}^u) P(w|MLE(v_i)) + \lambda_{v_i}^u P(w|\theta_{type(v_i)}) \quad (1)$$

This model estimates a distribution directly from observed text within the document component. The $\theta_{type(v_i)}$ model is a collection level background model for smoothing these estimates. The background model is sometimes referred to as a “universal” model, hence the u in λ^u . The *type* function may be used to specify document component specific models, as the language in titles titles may be different from other text, or it may simply return one language model for all components, which would provide larger amounts of text for the estimation of the corpus model.

The next step is to estimate the intermediate θ'_{v_i} model, from the bottom up to the top of the tree:

$$P(w|\theta'_{v_i}) = \lambda_{v_i}^c P(w|\theta_{v_i}) + \sum_{v_j \in children(v_i)} \lambda_{v_j}^c P(w|\theta'_{v_j}), \quad (2)$$

$$1 = \lambda_{v_i}^c + \sum_{v_j \in children(v_i)} \lambda_{v_j}^c$$

This model incorporates the evidence from the children nodes. If the λ^c parameters are set proportional to the length of the text in the node, as in

$$\lambda_{v_i}^c = \frac{|v_i|}{|v_i| + \sum_{v_k \in descendants(v_i)} |v_k|} \quad (3)$$

$$\lambda_{v_j}^c = \frac{|v_j| + \sum_{v_k \in descendants(v_j)} |v_k|}{|v_i| + \sum_{v_k \in descendants(v_i)} |v_k|}$$

where $|v_i|$ is the length in tokens of the text in node v_i not including tokens in its children, θ'_{v_i} is equivalent to a flat text model estimated from the text in node v_i interpolated with a background model. However, this choice of parameters is not required, and the weight placed on a child node may be dependent on the node’s type. For example, the text in title nodes may be more representative of queries than the body of a document, so a higher weight on the title model may improve retrieval performance.

After the estimation of θ'_{v_i} , the θ''_{v_i} models used for ranking the components are estimated from the root of the tree down:

$$P(w|\theta''_{v_i}) = (1 - \lambda_{parent(v_i)}^p) P(w|\theta'_{v_i}) + \lambda_{parent(v_i)}^p P(w|\theta''_{parent(v_i)}) \quad (4)$$

Incorporating the parent model in this way allows the context of the component within the document to influence the language model. Incorporating a parent’s language model in this way is referred to as shrinkage. In [8], McCallum and Nigam introduced shrinkage to information retrieval in the context of text classification. Classes were modeled hierarchically in this work. Class language models were estimated from the text in the documents assigned to the class, and all ancestor language models in the class hierarchy. The hierarchical model for classes used in [8] is very similar to the

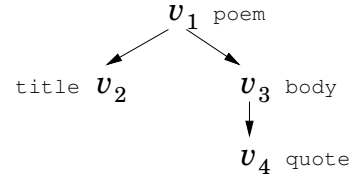


Figure 1: The tree representing document structure for “Little Jack Horner”.

model of documents presented in this proposal. The difference in model estimation this proposal is the application of shrinkage to document models, rather than class models.

The choice of linear interpolation parameters λ may depend upon the task and corpus. Ideally, the choice of these parameters would be set to maximize some measure of retrieval performance through automated learning.

A set of rankable items, document components that may be returned by the system, \mathcal{R} must also be defined for the document. For the hierarchical model presented here, \mathcal{R} may be any subset of \mathcal{V} .

2.1 Example

The estimation process described above may be clarified through describing the process for an example document. The example document is a well known children’s poem encoded in XML:

```

<poem id='p1'>
  <title> Little Jack Horner </title>
  <body>
    Little Jack Horner
    Sat in the corner,
    Eating of Christmas pie;
    He put in his thumb
    And pulled out a plumb,
    And cried, <quote> What a
    good boy am I! </quote>
  </body>
</poem>
  
```

There are four components of this document, corresponding to the poem, title, body, and quote tags. Let us now assign labels to these components v_1 to the poem, v_2 to the title component, v_3 to the body component, and v_4 to the quote. The structure of the document may be drawn, as in Figure 1 or be described as a set of vertices and edges:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

$$\mathcal{V} = \{v_1, v_2, v_3, v_4\}$$

$$\mathcal{E} = \{(v_1, v_2), (v_1, v_3), (v_3, v_4)\}$$

Not all components of the document may be rankable items. A set of rankable items must be defined. In our example, perhaps only the poem and the body of the poem are considered rankable items: $\mathcal{R} = \{v_1, v_3\}$.

The estimation process is illustrated in Figure 2. First, smoothed θ_{v_i} models are estimated for each vertex using

the text occurring in the document component corresponding to the vertex. Note that “What a good boy am I!” is not used for the estimation of θ_{v_3} , it is only used in the estimation for the model of v_3 ’s child node v_4 :

$$P(w|\theta_{v_i}) = (1 - \lambda_{v_i}^u) P(w|MLE(v_i)) + \lambda_{v_i}^u P(w|\theta_{type(v_i)}) \quad (5)$$

Next, the θ'_{v_i} models are estimated by combination of the θ_{v_i} model and the θ' models of v_i ’s children. For example, θ'_{v_3} is an interpolation of θ_{v_3} and θ'_{v_4} . Similarly, θ'_{v_1} is an interpolation of θ_{v_1} , θ'_{v_2} , and θ'_{v_3} :

$$P(w|\theta'_{v_1}) = \lambda_{v_1}^{c'} P(w|\theta_{v_1}) + \lambda_{v_2}^c P(w|\theta'_{v_2}) + \lambda_{v_3}^c P(w|\theta'_{v_3})$$

$$P(w|\theta'_{v_2}) = P(w|\theta_{v_2}) \quad (6)$$

$$P(w|\theta'_{v_3}) = \lambda_{v_3}^{c'} P(w|\theta_{v_3}) + \lambda_{v_4}^c P(w|\theta'_{v_4})$$

$$P(w|\theta'_{v_4}) = P(w|\theta_{v_4})$$

Finally, the θ''_{v_i} models used in ranking are estimated by interpolating the θ'_{v_i} model with the $\theta''_{parent(v_i)}$ model. In the example, θ''_{v_1} is simply taken as θ'_{v_1} as v_1 has no parent. The other vertices do have parents, and θ''_{v_3} is an interpolation of θ'_{v_3} and θ''_{v_1} :

$$P(w|\theta''_{v_1}) = P(w|\theta'_{v_1})$$

$$P(w|\theta''_{v_3}) = (1 - \lambda_{v_1}^p) P(w|\theta'_{v_3}) + \lambda_{v_1}^p P(w|\theta''_{v_1}) \quad (7)$$

We can expand the equations for these rankable items to use only θ models as follows:

$$P(w|\theta''_{v_1}) = \lambda_{v_1}^{c'} P(w|\theta_{v_1}) + \lambda_{v_2}^c P(w|\theta_{v_2}) + \lambda_{v_3}^c \left(\lambda_{v_3}^{c'} P(w|\theta_{v_3}) + \lambda_{v_4}^c P(w|\theta_{v_4}) \right)$$

$$P(w|\theta''_{v_3}) = (1 - \lambda_{v_1}^p) \left(\lambda_{v_3}^{c'} P(w|\theta_{v_3}) + \lambda_{v_4}^c P(w|\theta_{v_4}) \right) + \lambda_{v_1}^p P(w|\theta''_{v_1})$$

$$= (1 - \lambda_{v_1}^p + \lambda_{v_1}^p \lambda_{v_3}^{c'}) \left(\lambda_{v_3}^{c'} P(w|\theta_{v_3}) + \lambda_{v_4}^c P(w|\theta_{v_4}) \right) + \lambda_{v_1}^p \left(\lambda_{v_1}^{c'} P(w|\theta_{v_1}) + \lambda_{v_2}^c P(w|\theta_{v_2}) \right) \quad (8)$$

3. RANKING ITEMS FOR QUERIES

This section describes how rankable items are ordered for queries. Ordering of items is based on the query-likelihood model, where items are ranked by the probability of generating the query. It is also desirable to provide support for structured queries as well, which will be discussed.

3.1 Queries

Rankable items across documents for flat text queries may simply be ordered by $P(Q|\theta''_{v_i})$, where

$$P(Q|\theta''_{v_i}) = \prod_{w \in Q} P(w|\theta''_{v_i})^{tf(w,Q)} \quad (9)$$

This is the natural adaptation of query-likelihood [11][4][16][13][15] to the model.

There are many cases where it is desirable to place constraints on where the query terms appear in the document structure of a representation. This can be done by constraining which θ_{v_i} distribution generates the terms. For example, consider the NEXI [14] query

```
//poem[about(./title, Horner)]
```

which requests poem components where the title component is about ‘Horner’. The NEXI query language was developed as a simple adaptation of XPath to information retrieval. All example queries in this proposal will be expressed in NEXI. For our example document with a single representation, instead of measuring $P(\text{‘Horner’}|\theta''_{v_1})$, corresponding to the probability the poem component generated the query term ‘Horner’, $P(\text{‘Horner’}|\theta''_{v_2})$ is used. $P(\text{‘Horner’}|\theta''_{v_2})$ measures the probability that the title component generated ‘Horner’.

There are cases where a structural constraint on the query may be applicable to multiple components. Consider the query:

```
//document[about(./paragraph, plum)]
```

Most documents contain many paragraphs. Which distribution is chosen to generate ‘plum’? Many reasonable options are applicable. One approach may create a new distribution estimated as a combination of all of the θ''_{v_i} distributions corresponding to paragraph components. Another approach may take the θ''_{v_i} distribution corresponding to paragraph nodes that maximizes the probability of generating ‘plum’, which is the approach taken here.

Constraining the generating distribution in this manner is a strict interpretation of the constraints expressed in the query (as in previous SCAS tasks). The ranking items as described above requires that only poems be returned as results and that they contain titles. Note that through the use of smoothing, ‘Horner’ is not required to be present in the title. However, if the structural constraints are intended as hints to relevance (as in the VCAS task), then this approach can only return a subset of relevant items. Loose interpretation of structural constraints is something that remains a challenge and will be investigated as a part of future work.

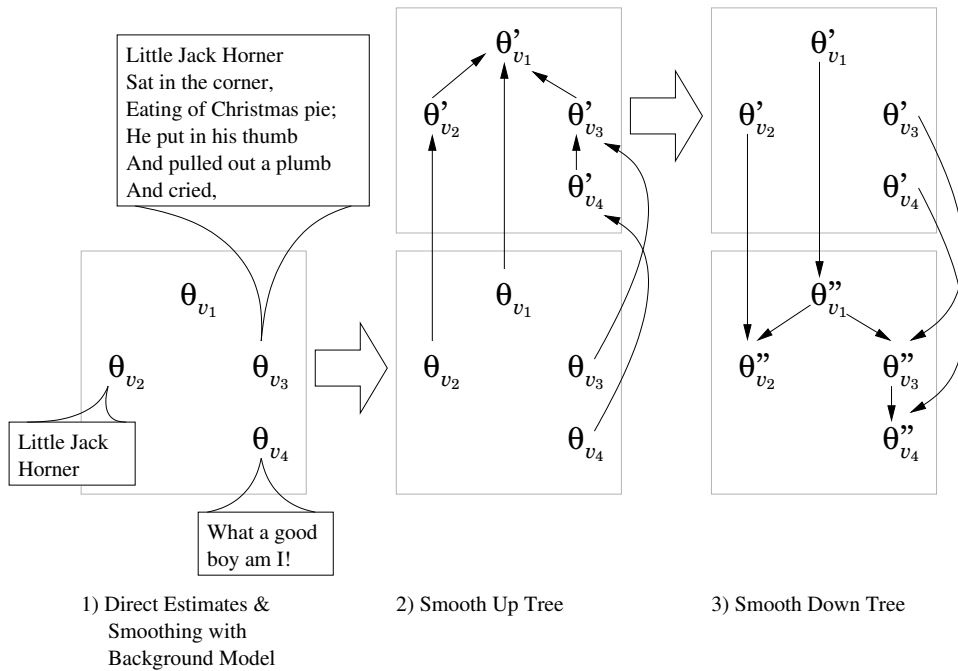


Figure 2: The estimation process for “Little Jack Horner”.

3.2 Priors

Query independent information about relevant documents is not uncommon and may be useful for ranking purposes. For example, there may be a tendency for longer documents to be more likely to be relevant than short documents. This information may be leveraged through the use of priors, which are a belief independent of the query that the document may be relevant. They are incorporated to ranking within the generative framework using Bayes rule:

$$\begin{aligned}
 & P(v_i \text{ is Rel} | Q, g(v_i) = a) \\
 & \propto P(Q | v_i \text{ is Rel}, g(v_i) = a) \\
 & P(v_i \text{ is Rel} | g(v_i) = a) \\
 & \approx P(Q | \theta''_{v_i}) P(v_i \text{ is Rel} | g(v_i) = a)
 \end{aligned}
 \tag{10}$$

where $g(v_i)$ is a function of the rankable item such as the length of v_i and $P(Q | \theta''_{v_i})$ is assumed representative of $P(Q | v_i \text{ is Rel}, g(v_i) = a)$. Theoretically, the prior probability $P(v_i \text{ is Rel} | g(v_i) = a)$ can be estimated from training data. However, in practice the prior is not a true prior probability estimate as it is often used to correct a bias in the ranking function at the same time as incorporating the prior belief that v_i is relevant. This makes the choice of how $P(v_i \text{ is Rel} | g(v_i) = a)$ is estimated somewhat of a fine art, rather than a theoretically driven process.

4. METHODOLOGY

All experiments use a local adaptation of the Lemur [1] toolkit for XML retrieval. Two databases were built - one using the Krovetz stemmer [6], and one without stemming. A stopword list of around 400 words was used for both

databases. Our prior INEX paper [10] describes most adaptations to index structures and basic retrieval algorithms used presently.

Since then, some query nodes for structured retrieval have been added. We presently only support *AND* clauses, *about* clauses, and path constraints. Numeric constraints are ignored by the retrieval system. *OR* clauses are converted to *AND* clauses, temporarily sidestepping the issue of how the *OR* probabilities are computed. *NOT* clauses are dropped from the query, as are terms with a ‘-’ in front of them. Different clauses of the queries are given equal weight. Phrase constraints are dropped, but the words are kept. A ‘+’ in front of a query term is ignored. Basically, all queries are converted to contain only *AND* clauses with path constraints and about clauses. For example, query 66 is converted from

```
//article[./fm//yr < 2000]
//sec[about(., 'search engines')]
```

to

```
//article//sec[about(., search engines)]
```

The graph structures used were taken directly from the XML structure of the document. All components were considered rankable items. The weight placed on the collection model λ^u is 0.2 and when using shrinkage, λ^p is set to 0.1. A single background model estimated from all text in the collection was used. Estimation of θ' models use $\lambda^{c'}$ and λ^c set according to Equation 3. These parameters were chosen by experimentation on the INEX 2003 topics.

The prior probabilities used in experiments are all based on the aggregated length of a component may take the following form:

- *linear* – $P(v_i \text{ is Rel} | \text{length}(v_i) = x) \propto x$
- *square* – $P(v_i \text{ is Rel} | \text{length}(v_i) = x) \propto x^2$
- *cubic* – $P(v_i \text{ is Rel} | \text{length}(v_i) = x) \propto x^3$

where

$$\text{length}(v_i) = |v_i| + \sum_{v_k \in \text{descendants}(v_i)} |v_k|. \quad (11)$$

5. EXPERIMENTS

This section describes some experiments with variations of the system. The discussion in this section centers on the content-only topics. Figure 3 examines the effects of prior probabilities on the strict measure and the specificity oriented measure for content-only topics. The runs in this figure used the Krovetz stemmer and a shrinkage parameter $\lambda^p = 0.1$. The use of more extreme length priors generally resulted in noticeable improvements to the strict measure but at a sacrifice to the specificity oriented measure. Results for more configurations and measures are presented in Table 5. The trends in Figure 3 are confirmed in the table. The more extreme the prior, the higher the exhaustivity and the lower the specificity. Using a more extreme prior also reduced overlap in the result lists, as these runs had distinct biases toward long components. For the rest of the discussion in this section, a linear prior was chosen as a good trade-off between improved exhaustivity and harmed specificity.

Next, some experiments using different shrinkage parameters are explored. For these runs, only the system using the Krovetz stemmer and a linear prior was explored. Figure 4 demonstrates that small values of λ^p boost precision modestly for both strict and specificity oriented measures. A non-zero shrinkage parameter did seem to help, and using too large a parameter hurt precision at low recall. The bottom half of Table 5 contains more evaluation measures and parameter settings. Small settings for the shrinkage parameter can improve performance across all measures, but these results may not be significant.

This section concludes with a brief discussion of the content-and-structure runs. Our original submission had a bug which allowed a component to be returned in the result list multiple times (with different supporting components). This bug has been fixed, and a comparison of the official runs and the bug-fixes are in Table 5. The runs using query structure took a strict interpretation of constraints and as such, it is not surprising the did poorly for the VCAS task. Our best performing run did not use any structure in the query. All non-structural constraints were removed from the query so that keywords present in the about clauses remained. This was then treated as a flat text query and run using the configuration for CO topics. The use of prior probabilities was investigated, and it was found that the trade-off between exhaustivity and specificity observed for CO held for VCAS as well. However, there does seem to be a preference for

shorter components in the VCAS task, as the square prior hurt performance across the board, while the linear prior improved performance.

6. RELATED WORK

Much of the current work in XML component retrieval can be found these proceedings and in [2][3], so only highly related works will be discussed here.

The Tjah system [7] also uses generative models for retrieval of XML components. They do not explicitly model the hierarchical relationship between components when estimating language models. Instead, they estimate a language model for each component using the text present in it and its children. This is equivalent to our model when $\lambda^p = 0$ and $\lambda^{c'}$ and λ^c are set according to Equation 3. They also incorporate prior probabilities using a log-normal and a linear component length prior. To provide context in the scoring of a component, they average the component score with the document score. For structured queries, constraints on structure are processed similarly. However, for *OR* clauses, the maximum of the scores is taken, while the minimum is taken for *AND* clauses. A system configuration for vague evaluation of structured queries is realized using query rewrites.

Kamps, Sigurbjörnsson, and de Rijke [5] [12] also work within the language modeling framework. Like [7], they do not explicitly model hierarchical relationship between document components when estimating the language model for a component. Rather than estimating the background model using a maximum likelihood estimator, they use an estimate based on element frequencies. They present experiments using a linear, square, and cubic component length prior, and also experiment with a cut-off filtering out short components. As with [7], their model is comparable to our model when $\lambda^p = 0$ and $\lambda^{c'}$ and λ^c are set according to Equation 3. For processing structured queries, [12] describes an approach that combines query rewrites with strict interpretation of the query.

7. CONCLUSIONS

This paper described experiments using hierarchical language models for modeling and ranking of XML document components. It extended previous work to incorporate context through the use of shrinkage, which helps modestly for flat text queries. A very small choice for the shrinkage parameter was found to be best for retrieval. This paper also presented experiments using length based priors. A prior probability proportional to the length of the component was found to be most effective across a number of measures.

For vague content and structure queries, where structure is intended only as a hint to the retrieval system, we found that ignoring structure in the query was better than taking a strict interpretation of the structural constraints. This is much like using a flat text query. As with the content only queries, a linear length prior was found to improve performance, but the vague content and structure queries may have a preference for shorter components than the content only queries on average.

Future experiments will examine use of the structural con-

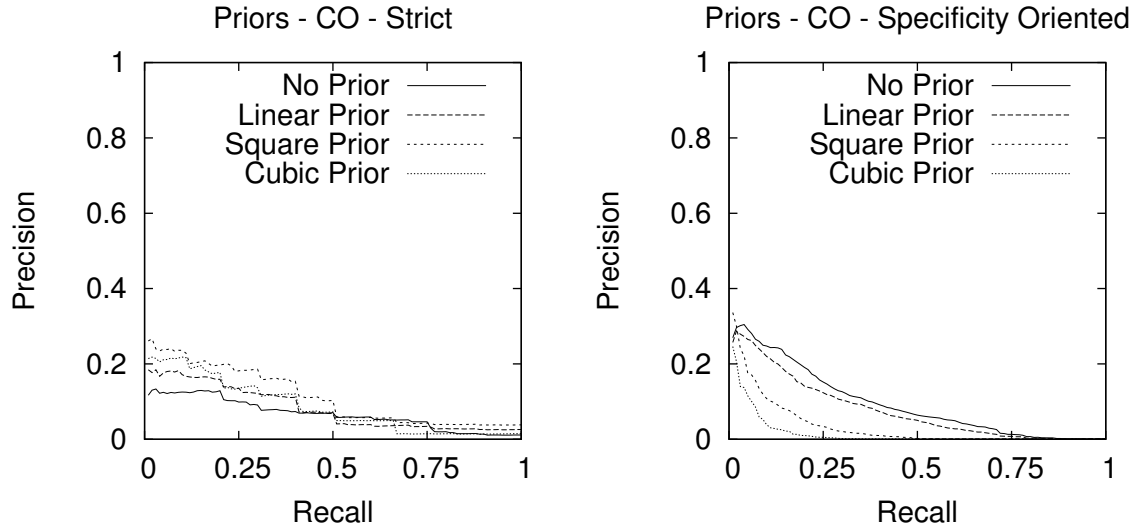


Figure 3: More extreme length priors can greatly improve performance under strict evaluation, but at a sacrifice to specificity oriented evaluation.

Table 1: Run performance for Content-Only topics.

Official	λ^p	Stemmer	Prior	Strict	Generalized	SO	s3 e321	s3 e32	e3 s321	e3 s32	Overlap	Aggregate
YES ¹	0.0	-	-	0.0640	0.0728	0.0655	0.0541	0.0494	0.0806	0.0693	66.7	0.0651
NO	0.0	-	linear	0.0896	0.0770	0.0650	0.0564	0.0602	0.1234	0.0950	72.6	0.0809
NO	0.0	-	square	0.1224	0.0448	0.0318	0.0236	0.0461	0.1864	0.1618	47.1	0.0881
NO	0.0	-	cubic	0.0902	0.0226	0.0167	0.0130	0.0266	0.1367	0.1342	41.7	0.0629
YES ²	0.1	-	-	0.0675	0.0908	0.0897	0.0771	0.0685	0.0769	0.0729	74.8	0.0774
NO	0.1	-	linear	0.0688	0.0829	0.0721	0.0640	0.0634	0.1055	0.0839	73.4	0.0772
NO	0.1	-	square	0.1268	0.0480	0.0344	0.0262	0.0493	0.1885	0.1639	46.3	0.0910
NO	0.1	-	cubic	0.0927	0.0230	0.0173	0.0139	0.0277	0.1380	0.1364	40.5	0.0641
YES ³	0.1	Krovetz	-	0.0667	0.0947	0.0941	0.0835	0.0728	0.0776	0.0753	73.0	0.0807
NO	0.1	Krovetz	linear	0.0817	0.0882	0.0770	0.0663	0.0655	0.1191	0.0997	72.6	0.0853
NO	0.1	Krovetz	square	0.1129	0.0445	0.0330	0.0252	0.0434	0.1721	0.1716	46.2	0.0861
NO	0.1	Krovetz	cubic	0.0859	0.0200	0.0151	0.0117	0.0231	0.1324	0.1492	40.4	0.0625
NO	0.000	Krovetz	linear	0.0745	0.0771	0.0651	0.0561	0.0537	0.1173	0.0910	72.4	0.0764
NO	0.025	Krovetz	linear	0.0874	0.0867	0.0748	0.0632	0.0636	0.1315	0.1066	72.1	0.0877
NO	0.050	Krovetz	linear	0.0849	0.0885	0.0765	0.0654	0.0651	0.1315	0.1050	72.3	0.0881
NO	0.075	Krovetz	linear	0.0830	0.0889	0.0775	0.0667	0.0660	0.1255	0.1010	72.5	0.0869
NO	0.100	Krovetz	linear	0.0817	0.0882	0.0770	0.0663	0.0655	0.1191	0.0997	72.6	0.0853
NO	0.125	Krovetz	linear	0.0682	0.0840	0.0734	0.0632	0.0599	0.1076	0.0905	72.8	0.0781
NO	0.150	Krovetz	linear	0.0591	0.0761	0.0670	0.0573	0.0516	0.0915	0.0769	73.0	0.0685

¹Lemur_CO_NoStem_Mix02 ²Lemur_CO_NoStem_Mix02_Shrink01 ³Lemur_CO_KStem_Mix02_Shrink01

Table 2: Run performance for Content-and-Structure topics.

Official	Structure	λ^p	Prior	Strict	Generalized	SO	s3 e321	s3 e32	e3 s321	e3 s32	Overlap	Aggregate
YES ¹	NO	0.1	-	0.0710	0.0746	0.0759	0.0834	0.0799	0.0700	0.0764	74.0	0.0759
NO	NO	0.1	Linear	0.0889	0.0847	0.0804	0.0819	0.0839	0.0949	0.0904	69.9	0.0864
NO	NO	0.1	Square	0.0585	0.0468	0.0377	0.0359	0.0419	0.0836	0.0576	48.3	0.0217
YES ²	YES	0.0	-	0.0468	0.0180	0.0166	0.0253	0.0367	0.0419	0.0424	2.4	0.0325
NO (fix)	YES	0.0	-	0.0616	0.0276	0.0309	0.0409	0.0604	0.0413	0.0590	4.6	0.0459
YES ³	YES	0.1	-	0.0466	0.0199	0.0177	0.0249	0.0369	0.0457	0.0455	2.4	0.0339
NO (fix)	YES	0.1	-	0.0621	0.0274	0.0302	0.0409	0.0606	0.0418	0.0591	4.8	0.0460

¹Lemur_CAS_as_CO_NoStem_Mix02_Shrink01 ²Lemur_CAS_NoStem_Mix02 ³Lemur_CAS_NoStem_Mix02_Shrink01

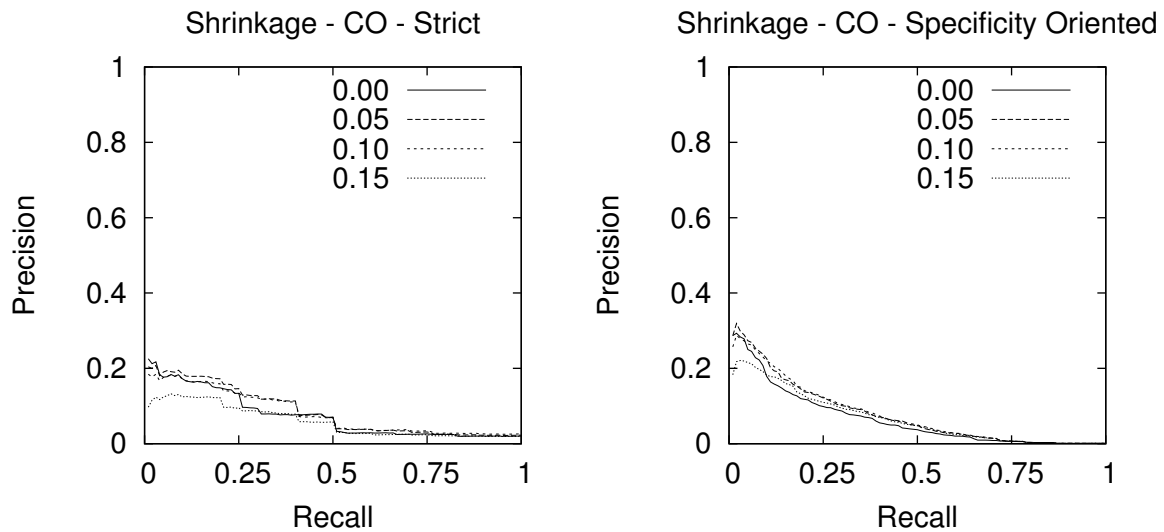


Figure 4: Very small shrinkage parameter values boost precision moderately at mid-recall ranges for strict evaluation and across the range for specificity oriented evaluation.

straints in the content and structure queries as hints for relevance within the framework. More experimentation with how the shrinkage parameter is set will be performed, as well as different approaches to setting the interpolation parameters for the combination of evidence from child nodes.

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.

9. REFERENCES

- [1] The lemur toolkit for language modeling and information retrieval. Technical report. <http://lemurproject.org/>.
- [2] N. Fuhr, N. Goevert, G. Kazai, and M. Lalmas, editors. *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*. ERCIM, 2003.
- [3] N. Fuhr, S. Maalik, and M. Lalmas, editors. *Proc. of the Second Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX)*, Dagstuhl, Germany, Dec. 2003.
- [4] D. Hiemstra. *Using language models for information retrieval*. PhD thesis, University of Twente, 2001.
- [5] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length normalization in xml retrieval. In *Proceedings of the Twenty-Seventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 80–87, 2004.
- [6] R. Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 191–202. ACM, 1993.
- [7] J. List, V. Mihajlovic, G. Ramirez, and D. Hiemstra. The tijah xml-ir system at inex 2003. In *INEX 2003 Workshop Proceedings*, pages 102–109, 2003.
- [8] A. McCallum and K. Nigam. Text classification by bootstrapping with keywords, em and shrinkage. In *Proceedings of the ACL 99 Workshop for Unsupervised Learning in Natural Language Processing*, pages 52–58, 1999.
- [9] P. Ogilvie and J. Callan. Language models and structured document retrieval. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, 2003.
- [10] P. Ogilvie and J. P. Callan. 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, Dec. 2003.
- [11] J. Ponte and W. Croft. A language modeling approach for information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281. ACM Press, 1998.

- [12] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Processing content-and-structure queries for xml retrieval. In *Proceedings of the First Twente Data Management Workshop*, pages 31–38, 2004.
- [13] F. Song and W. Croft. A general language model for information retrieval. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, 1999.
- [14] A. Trotman and B. Sigurbjörnsson. Narrow Extended XPath I. Technical report, 2004. Available at <http://inex.is.informatik.uni-duisburg.de:2004/>.
- [15] T. Westerveld, W. Kraaij, and D. Hiemstra. Retrieving web pages using content, links, URLs, and anchors. In *The Tenth Text REtrieval Conf. (TREC-10), NIST SP 500-250*, pages 663–672, 2002.
- [16] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 2(2), April 2004.

Ranked Retrieval of Structured Documents with the S-Term Vector Space Model

Felix Weigel

Klaus U. Schulz

Holger Meuss

Centre for Information and Language Processing
University of Munich (LMU), Germany
Oettingenstraße 67, D-80538 Munich
{weigel,schulz}@cis.uni-muenchen.de

European Southern Observatory (ESO)
Headquarter Garching, Germany
Karl-Schwarzschild-Straße 2, D-85748 Garching
hmeuss@eso.org

ABSTRACT

This paper shows how the s-term ranking model [13] is extended and combined with index structures and algorithms for structured document retrieval to enhance both the effectiveness of the model and the retrieval efficiency. We explain in detail how previous work on ranked and exact retrieval can be integrated and optimized, and which adaptations are necessary. Our approach is evaluated experimentally at the INEX workshop 2004 [4]. The results are encouraging and give rise to a number of future enhancements.

1. INTRODUCTION

The retrieval and ranking of structured text documents has by now become an IR discipline in its own right. As more and more documents are available in ever growing web repositories, digital libraries, and intranet knowledge bases, performance both in terms of the ranking effectiveness and the retrieval efficiency is paramount. When searching the relevant parts of millions of documents occupying hundreds of megabytes, naive solutions often turn out to be inadequate. Many sophisticated ranking models for structured documents have been proposed [2, 19, 13, 12, 15, 14], most of which are based on the traditional *tf-idf* model for “flat” text documents [11]. Two key issues in the adaption of *tf-idf* to structured data are (1) the *document boundary problem*, i.e. the question which units of the data are to be treated as coherent pieces of information, and (2) the *structural similarity problem*, which concerns methods for quantifying the distance of a document to a given query. While the first problem is intrinsic to structured documents whose hierarchical nature blurs the physical boundaries imposed by a file system, the second problem arises in “flat” retrieval as well. However, similarity measures for structured documents necessarily make assumptions concerning the documents to be ranked, hence both problems are tightly linked.

The second aspect of performance, efficiency, seems to be a greater concern in the field of exact (i.e., unranked) retrieval of structured documents. However, ongoing work [18, 17, 14, 10] investigates how to integrate effective ranking techniques with data structures and algorithms for the efficient exact retrieval. In this paper, we follow the approach adopted in [18] and integrate the *s-term* ranking model [13] with the *Content-Aware DataGuide (CADG)* index, which stores additional information needed for computing relevance scores. As a further efficiency enhancement, an efficient

structural join [1] is applied during relevance computation. To increase the effectiveness of the original s-term model, we extend it with several IR features such as order and distance constraints on search term occurrences and Boolean constraints on textual and structural parts of the query. We also report on a first evaluation of the extended model in experiments at the INEX workshop 2004 [4].

The paper proceeds as follows: Section 2 reviews the original s-term model in a nutshell. Section 3 explains the data structures and algorithms used in our implementation of the s-term model, as well as the extensions to the model. Section 4 presents and discusses the experimental results we obtained at INEX 2004. Section 5 briefly reviews some of the related work cited above. Section 6 concludes with a glance at our agenda for the s-term model.

2. S-TERM VECTOR SPACE MODEL

The *s-term vector space model* [13] combines the vector space model (VSM) [11] with Kilpeläinen’s tree matching approach [5] to retrieving structured documents. The key concepts in the VSM, *document*, *query* and *term*, are adapted so as to cover both the structural and the textual dimension of the data. To this end, documents, queries and – most notably – terms are modelled as labelled trees in [13]. Note that the resulting *structured terms (s-terms)* are in general subtrees of queries and documents, labelled with both element names and “flat” terms as known from traditional IR. As a special case, an s-term representing a “flat” term consists of a single labelled text node. Each s-term contained in a query may be weighted to reflect its contribution to the underlying information need. In [13] it is shown that with suitable weights for query terms, the s-term vector space model can simulate both the traditional VSM for “flat” documents and the tree matching approach.

Like in the original VSM, documents are described by document vectors whose weight components are computed from the distribution of s-terms in the collection, using the *tf-idf* method [11]. The definition of the *inverse document frequency idf* is related to the notion of documents to be retrieved in response to a query, and hence to the document boundary problem (see Section 1). The s-term vector space model addresses this issue by introducing the concept of a *logical document* (as opposed to the *physical documents* comprising the collection), which applies to any subtree of the document tree. In other words, any subtree of the docu-

ment tree can be returned as an answer to a query, provided its root has the same label as the query root. This way the document boundary problem is solved at query time.

Interestingly, the s-term vector space model supports ranking of partial matches to a query. Both violations to structural and textual query conditions are tolerated, although by default the structure is regarded as a weaker criterion than the keywords in the query. More precisely, query keywords occurring in a structural context which does not satisfy the query still contribute to the relevance of a document. Conversely, documents which lack a particular query keyword but contain matches to other parts of the query (i.e., other s-terms not containing the missing “flat” term) may still be considered relevant. Adjusting the weights of individual s-terms in the query, the user may tune the model more towards structure or keyword matching.

Most of the concepts related to the s-term vector space model are introduced on an informal basis in this section. For formal definitions, see [13].

2.1 Data model

Both *documents* and *queries* are conceived as unordered labelled trees in the s-term vector space model. Order conditions are not considered, and the original s-term model draws no distinction between the parent/child relation and the more general ancestor/descendant relation between document nodes (but see Section 3.1). Besides documents and queries, [13] defines *structured terms (s-terms)* and the frequency of their occurrences in documents. The notion of s-terms can be viewed as a generalization of the “flat” terms known from traditional IR, which are simply words (or word compounds). By contrast, a structured term is an unordered tree whose nodes are labelled with element names or “flat” terms, as illustrated in Figure 1.

A query q contains an s-term s if s is a full subtree of q .¹ For instance, the query in Figure 1 contains the six s-terms s_0 to s_5 corresponding to the non-shaded parts of the insets on the right. An s-term s is said to *occur* in a document d if there is an embedding f from s to d respecting the labels and ancestor/descendant relationships. The document node $f(\text{root}(s))$ (where $\text{root}(s)$ denotes the root of a tree or s-term s) is called an *occurrence* or *match* of s in a document. Note that the concept of occurrence in documents, relying on tree embedding, is somewhat weaker than that of containment in a query: an occurrence of an s-term in a document may have descendants which do not match any query node. (In particular, the same document node may be an occurrence of different s-terms.) Any logical document in which at least one s-term contained in q occurs is an *answer* to q . Note that the occurring term need not be the *root s-term* (the largest s-term contained in q , e.g., s_0 in Figure 1). Intuitively, an answer is any subtree of the document tree which (1) has the same label as the query root and (2) contains an occurrence of a full query subtree (but see the discussion on *target elements* in Section 3.4).

2.2 Similarity of queries and documents

¹By a *full subtree* of a tree T we mean a subtree of T that has one of the nodes of T as its root and contains all children and descendants of that node as children and descendants. Hence the number of full subtrees equals the number of nodes in T .

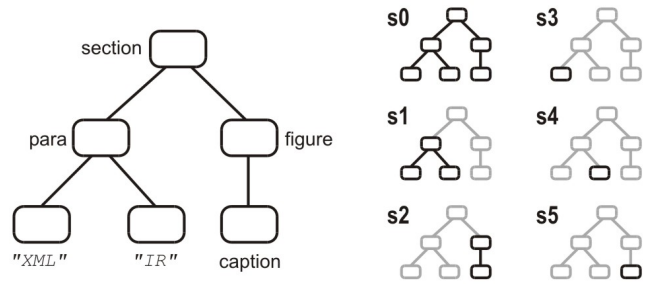


Figure 1: Query tree consisting of six s-terms.

Computation of term weights. The two *tf-idf* components *term frequency* and *inverse document frequency* and the resulting *weight* of a term in a document are defined as follows:

$$tf_{s,d} := \frac{freq_{s,d}}{maxfreq_d}$$

$$idf_s^t := \log \frac{|D^t|}{df_s^t} + 1$$

$$w_{s,d}^t := tf_{s,d} \cdot idf_s^t$$

Like in the VSM, the *term frequency* $tf_{s,d}$ indicates how often a given s-term s occurs in a given document d . The *raw (i.e., unnormalized) term frequency* $freq_{s,d}$ is normalized with the *maximal term frequency* $maxfreq_d$, i.e. the maximal number of occurrences of any s-term in d . As pointed out in [13], $maxfreq_d$ is bounded by the maximal number of occurrences of any element name or “flat” term in d .² In the second definition, D^t is the set of logical documents of type t , i.e. with the same label t as the query root. The *inverse document frequency* idf_s^t is defined in terms of the number $|D^t|$ of logical documents and the number df_s^t of logical documents in D^t containing the structured term s . Thus only logical documents of a particular type specified in the query (e.g., *section* in Figure 1) contribute to the inverse document frequency. Both $tf_{s,d}$ and idf_s^t together determine the weight $w_{s,d}^t$ of a given term s in a given logical document d of type t , as defined above.

Computation of relevance scores. Finally, each document d of type t is conceptually described by a document vector \underline{d} consisting of term weights $w_{s,d}^t$ for all terms s . The similarity ϱ of a document d w.r.t. to a query q is determined by the dot product $\varrho_{d,q} := \underline{d} \bullet \underline{q}$ where \underline{q} is the vector of all query term weights (0 for s-terms not contained in q). Note that there is no need to compute all document vectors and store them at indexing time. Instead only those term weights which contribute to the final result are calculated during query evaluation (see below).

3. IMPLEMENTATION

For our experimental evaluation of the s-term vector space model we integrated it with the retrieval system X² [9,

²At first sight it may seem biased to normalize complex terms with $maxfreq$ values determined mostly by “flat” terms with high frequencies. This obviously results in low tf values for complex terms. But the effect is compensated for by typically higher idf values for complex terms, which tend to occur less frequently in the collection than “flat” terms.

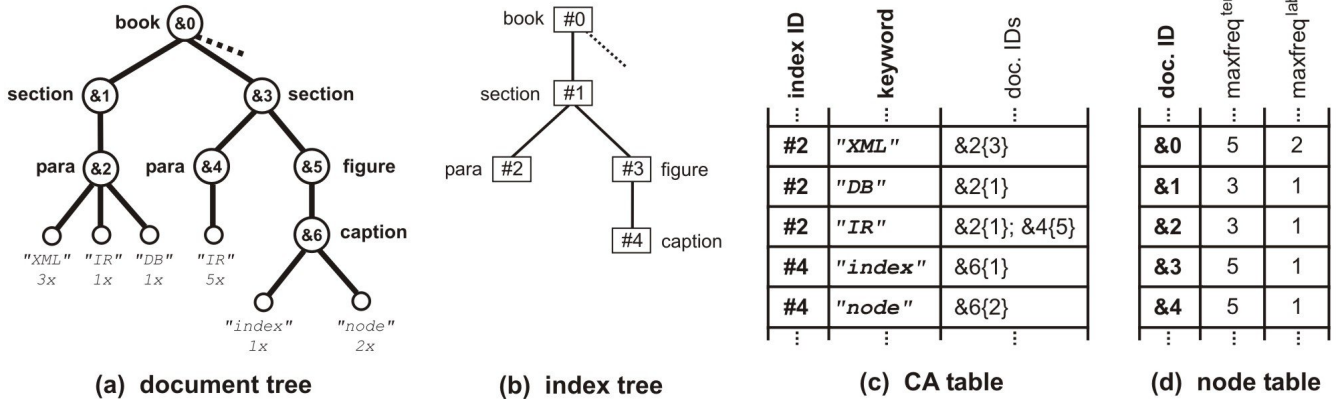


Figure 2: Indexing a sample document collection with the IR-CADG for s-term evaluation

6]. Pursuing a tree matching approach to structured document retrieval, X^2 makes use of a *Content-Aware DataGuide (CADG)* [17, 18] as path index for efficiently computing *Complete Answer Aggregates (CAA)* [8, 7], a data structure for structured query results. Section 3.1 describes the adaptations to the CADG and CAA which were necessary for s-term evaluation and summarizes the query formalism, as well as necessary modifications to the original ranking model. Based on these preliminaries, Section 3.2 then discusses the bottom-up tree matching algorithm replacing the original algorithm used by X^2 . Section 3.3 describes the computation of relevance scores for logical documents. Section 3.4 sketches further features of our implementation extending both the X^2 system and the original s-term model.

3.1 Preliminaries

Tree query language. In the query formalism used by X^2 , queries are trees consisting of labelled nodes. There are *structural* query nodes, which are matched by document nodes with a suitable label, and *textual* query nodes specifying keyword constraints. A path in the query tree consists of a sequence of structural nodes and (possibly) a single textual leaf containing “flat” search terms. Edges between query nodes may be either *rigid* or *soft*, corresponding to a parent/child or ancestor/descendant relation, respectively. Figure 1 depicts a query tree with four structural and two textual nodes, linked by rigid edges (solid lines). The original s-term model, which ignores the parent/child relation, is easily adapted to the distinction of soft and rigid edges.

Structural query nodes may be labelled with any number of alternative element names, or else a wildcard matching any element name. Textual query nodes may specify more than one search term in a logical disjunction or conjunction. A conjunction is either unordered or ordered. In the latter case, an optional minimal and maximal token distance may be specified for any two consecutive terms. Section 3.4 describes order and distance constraints in more detail. As a special case, the implementation of phrase search as required by the NEXI query language [16] is built upon this feature. To support another NEXI construct which was missing in X^2 , we integrated Boolean expressions over sibling query nodes. Allowing for partial s-term matches, this affects the way occurrences are counted, as explained in Section 3.4.

IR-CADG index. We slightly extended the original CADG index [17] used by X^2 to obtain an *Integrated Ranking CADG (IR-CADG)* for the s-term vector space model. Similar adaptations of the CADG index to other ranking models [19, 2, 15, 14] are described in previous work [18]. Figure 2 (b) and (c) depict the data structures of the IR-CADG for the sample document in (a), whose leaf nodes contain one or more occurrences of different “flat” terms. The index consists of an index tree (b), residing entirely in main memory, and a *Content/Annotation Table (CA Table)* stored in a relational database system (c). As can be seen in the figure, a single path in the index tree – e.g., the one leading to index node #2 in (b) – represents all identical label paths in the documents – in this case, the two *book/section/para* paths reaching the document nodes &2 and &4 in (a). (We say that these document nodes are *referenced by* #2.) Compared to the document tree, the index tree therefore has a path structure which is typically extremely small and far less redundant. This motivates its use as a compact structural summary for schema browsing and path matching [3, 9].

Keywords are indexed in the CA Table (c), which maps a given index node ID, representing a label path, and a “flat” term to the list of document nodes with that same label path which contain occurrences of that same term. Unlike the original CADG, the IR-CADG for the s-term vector space model also records how often the term occurs in the document node. This node-specific occurrence count, which we henceforth refer to as the *local term frequency* tf^{loc} , is stored with each document node ID in the CA Table (in Figure 2 (c), the integers in curly braces). Section 3.3 explains how raw term frequencies are computed from the local term frequencies in the CA Table. During index creation, for each document node being indexed the maximal number of occurrences of any element name ($maxfreq^{label}$) and any “flat” term ($maxfreq^{term}$) in its subtree is calculated. The values are needed for normalizing the raw term frequency (see Section 3.3). Since X^2 internally uses a *node table* for storing document-node specific information (such as byte offsets in XML source files etc.), we simply add columns for $maxfreq^{label}$ and $maxfreq^{term}$ to the node table.

Complete Answer Aggregates. A *Complete Answer Aggregate (CAA)* is a compact data structure for representing the complete set of answers (document subtrees) retrieved

in response to a structured query. (Its properties are analyzed in [8], along with an algorithm for computing CAAs in polynomial time.) As shown in Figure 3, a CAA A_q has the same overall structure (shaded edges) as the corresponding tree query q (see Figure 1), representing each structural or textual query node by a dedicated *slot* containing that query node’s matches. For instance, in Figure 3 the document node &6 is the only match to the query node labelled **caption** in Figure 1, whereas the “*IR*” node has two matches (&2 and &4). Besides document nodes, the CAA temporarily stores pointers to index nodes during query evaluation (omitted in the figure).

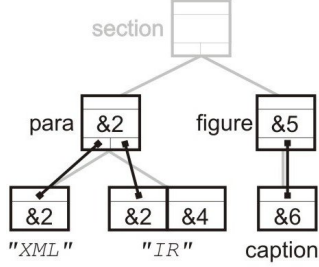


Figure 3: CAA for the query shown in Figure 1.

one match can be reached in each of the other slots by following these links. This is because a subtree of the document collection which does not comprise matches to all query nodes fails to satisfy the full set of query constraints, therefore being inadmissible in exact retrieval. By contrast, our implementation of the tree matching algorithm with *s*-term ranking (see Section 3.2) may produce partly filled CAAs like the one shown in Figure 3 which contains only document subtrees matching the smaller *s*-terms s_1 to s_5 , but no match to the root *s*-term s_0 .

In the following, the term *logical document root* refers to all document nodes in the set D^t . The index nodes referencing any such document node are called *logical index roots*.

3.2 Tree matching algorithm

The following tree matching algorithm computes a modified CAA as described in the previous subsection. It follows a bottom-up strategy in the sense that occurrences of smaller *s*-terms are obtained before those of larger *s*-terms. This is preferable to a top-down strategy since occurrences of a complex *s*-term invariably include occurrences of all subterms of that *s*-term in the document collection. From these subterm occurrences the occurrences of more complex *s*-terms can easily be derived in a step-wise process.

The core of the *s*-term tree matching algorithm is given in Listing 1. The procedure *evalSTerm* visits all nodes in the query tree in a depth-first traversal, recursively processing the subtrees of complex *s*-terms (lines 18 to 20). When a query leaf is reached, recursion halts and the occurrences of the corresponding leaf *s*-term in the documents are fetched from disk (*occurrence fetching*, line 24), along with their respective local term frequencies tf^{loc} (see the previous subsection). The CADG allows textual leaves representing “flat” terms and structural leaves representing element names to be treated alike, as postulated in [13]. All fetched document nodes are stored in the CAA slot corresponding to the

query leaf, together with pointers to (1) their referencing index node and (2) the lowest of their logical index roots (as defined above; there may be multiple logical document and index roots in case of a recursive document collection). The logical document and index roots are fetched once before the recursive matching procedure is called.

```

1 // evalSTerm: recursively evaluates an s-term
2 // → r: the root query node of the s-term to be evaluated
3 // → i: the IR-CADG to be used for evaluation
4 // ⇐ A: the CAA holding the query result
5 proc evalSTerm (r: QueryNode, i: IR-CADG, A: CAA)
6
7 // get hold of the parent query node and slot
8  $r_p :=$  the parent of r
9  $a_p :=$  the slot corresponding to  $s_p$  in A
10
11 // create a new slot for r in A
12  $a :=$  a new slot corresponding to r in A
13  $a.parent := a_p$ 
14
15 // recursively process inner s-terms
16  $C := r.children$ 
17 if  $C = \emptyset$  then
18   for all  $r_c \in C$  do
19     call evalSTerm ( $r_c, i, A$ )
20   end for
21
22 // recursion anchor: process leaf s-terms
23 else
24   call  $i.fetchOccurrences$  ( $r, a$ )
25 end if
26
27 // trigger path matching or joining
28 if r is the leftmost child of  $r_p$  then
29   call  $matchPath$  ( $r_p, a_p, r, a, i, A$ )
30 else
31   call  $joinPath$  ( $r_p, a_p, r, a, i, A$ )
32 end if
33
34 end proc

```

Listing 1: Recursive *s*-term evaluation

At this point the occurrences of the current *s*-term need to be propagated upwards along the query path in order to collect a set of candidate occurrences for the parent query node (*path matching*, line 29), or else to verify existing ones (*path joining*, line 31). In the following we assume that query nodes are processed in preorder. Different query plans (based e.g. on the selectivity of terms or labels associated with the query nodes) are ignored for the sake of simplicity.

Path matching. If the current query node r is the leftmost child of its parent r_p , no candidate occurrences for r_p have been collected yet. Obviously any occurrence of the parent *s*-term (the *s*-term rooted at r_p) physically includes occurrences of all child *s*-terms below r_p . These child occurrences, some of which are already known in the incarnation of r , hint at possible parent occurrences yet to be confirmed (so-called *occurrence candidates*). Candidates are all matches to the parent query node which are ancestors (in case of a soft edge, or parents for a rigid edge) of a child occurrence in

the same logical document. The label and level constraints involved are first checked in the index (remember each child occurrence comes with its referencing index node and lowest logical index root). Only for ancestor (parent) index nodes which are descendants of the lowest logical index root and match the parent query node, the corresponding ancestor (parent) document nodes are determined and passed back to the calling incarnation. If the ancestor (parent) document nodes must be fetched from disk, this may save many needless I/O operations for nodes which do not match the parent query node. Along with the referencing index nodes, the occurrence candidates for the parent query node are stored in the CAA before undergoing path joining and further upward propagation and finally producing the occurrences of the root s-term s_0 .

Path joining. When visiting child nodes other than the left-most one, a previous path matching step has already collected candidate occurrences for the parent query node r_p , some of which may not contain occurrences of the current child r' . Ruling out these false occurrence candidates is the aim of the path joining step (unless Boolean child constraints are enabled, see Section 3.4). The candidates for r_p to be kept are those which have at least one descendant (in case of a soft edge, or child for a rigid edge) among the occurrences of r' . Again, this condition is checked for the corresponding index nodes first, which may save the manipulation and comparison of huge sets of document nodes. Only if an index node referencing occurrence candidates of the parent s-term has a descendant (child) among the index nodes referencing occurrences of the current child s-term, the corresponding occurrence candidate sets are compared to rule out some of the parent candidates. When the last child query node has been processed, the remaining parent candidates are confirmed as occurrences of the parent s-term, ready to enter a new propagation phase until the query root is reached.

3.3 Relevance computation

When tree matching is over, the computed CAA contains the occurrences of all s-terms in the query, as mentioned in Section 3.1. Each occurrence e of a “flat” s-term s' comes with its local term frequency $tf_{s',e}^{loc}$ as defined in Section 3.1. For s-terms s'' involving node labels, $tf_{s'',e}^{loc}$ is fixed to 1 in the original model since any such s-term can occur only once in the same document node (the occurrence being defined as the document node itself, see Section 2.1). A modified definition of $tf_{s'',e}^{loc}$ for the use with Boolean child constraints is given in Section 3.4. Note that the root slot of the CAA contains only those logical document roots which are occurrences of the root s-term (i.e. the whole query tree). This is a subset of the set of logical document roots D^t fetched for tree matching. However, relevance scores are computed for all members of D^t since logical documents containing occurrences only of smaller s-terms are also part of the query result, even though their roots are missing in the CAA.

According to Section 2.2, to compute the weight $w_{s,d}^t$ quantifying the relevance of an s-term s w.r.t. a logical document d of type t , we need to calculate the following four

frequencies:

- $freq_{s,d}$: number of occurrences of s in d
- $maxfreq_d$: max. number of occurrences of any term in d
- $|D^t|$: number of logical documents of type t
- df_s : number of logical documents of type t containing occurrences of s

While $|D^t|$ is immediately available, determining the value of $maxfreq_d = \max(maxfreq_d^{term}, maxfreq_d^{label})$ requires access to the node table, but only for those logical document roots d' with $freq_{s,d'} > 0$. The raw term frequencies $freq$ are computed in a linear ancestor/descendant join of all logical document roots and all s-term occurrences from the CAA (see below). An array F of frequency vectors iteratively accumulates the $freq$ values for all terms and documents during the join. Let F_s denote the frequency vector for term s and $F_{s,d}$ its component for the logical document root d . For each pair (d, e) joining d with a descendant e of d containing an occurrence of the s-term s , the term frequency $F_{s,d}$ is incremented by $tf_{s,e}^{loc}$. After the join, F contains a raw term frequency $freq_{s,d}$ for each s-term s and each document d it occurs in. The document frequency df_s equals the length of the frequency vector F_s .

3.4 Further features

The following features extend both the X² system and the original s-term model as described in Sections 2 and 3.

Linear ancestor/descendant join. As explained in Section 3.3, the computation of term weights involves a join of logical document roots and s-term occurrences. We implemented a $|q|$ -way join, where $|q|$ is the number of s-terms in the query q , with an ancestor stack similar to the *stack-tree join* proposed in [1]. The stack serves to keep track of “active” members of the list of potential ancestors, whose subtree may contain some of the potential descendants to be checked next. This technique guarantees that the join is performed in time linear in the number of potential ancestor and descendant nodes involved, i.e. $\mathcal{O}(|D^t| + |A_q|)$ where $|A_q|$ is the number of occurrences stored in the aggregate A_q .

Target elements. In Section 2.1, the set of answers to a structured query q is defined as a subset of the set of logical documents induced by the label of $root(q)$. In some cases, however, it may be more useful to retrieve specific parts of logical documents while keeping the query-specific document concept intact. In our implementation, we extended the original s-term model by introducing the notion of a *target element*, which specifies which kind of document nodes are to be retrieved as answers to a given query q . This is done by marking a single node in the query tree for q as being the target element $target(q)$ of q . As a consequence, matches to the target element are either logical document roots or descendants of a logical document root. Note that the use of target elements typically makes the query result more specific; e.g., the user may be given individual paragraphs in response to a query which otherwise returns only articles.

As a special case, the query root $root(q)$ and the target element $target(q)$ may be identical, such that entire logical document are retrieved as described before. Otherwise the query is evaluated as follows. First the subtree q_{target} of q rooted at $target(q)$ is evaluated in isolation, i.e. with

$target(q)$ as root node specifying “logical subdocuments” to be retrieved in response to this part of the original query q . Ranking of the matches to $target(q)$ takes place as described in Section 3.3. In a second step, the remaining part q_{root} of q (i.e., the entire query tree except the subtree rooted at $target(q)$) is evaluated in the same way. During evaluation, each match e to $target(q)$ is associated with the set R_e of roots of logical documents containing e . As mentioned in the previous paragraph, $R_e \neq \emptyset$ for all such occurrences of the target element. Finally, the relevances score of any given match e to $target(q)$ computed in the first step is *biased* with the relevance scores of all elements in R_e , in order to capture both the relevance of e w.r.t. q_{target} and the relevance of the containing logical document(s) w.r.t. q_{root} . Many biasing methods are conceivable; currently we simply calculate the relevance of an occurrence e of $target(q)$ w.r.t. q as $\varrho_{e,q} := \varrho_{e,q_{target}} \cdot \sum_{e' \in R_e} \varrho_{e',q_{root}}$.

The effectiveness of this extension to the original s-term model depends largely on the choice of suitable target elements, an issue which we have not yet examined in detail.

Boolean constraints on child query nodes. Any inner query node may be constrained by an arbitrarily nested Boolean expression involving (1) atoms representing the query subtrees rooted at its child nodes and (2) the Boolean operators $\{\wedge, \vee, \neg\}$. Replacing the default conjunctive interpretation of child query nodes in tree matching, the Boolean constraints allow individual subtrees to be treated as optional, alternative, or negated. For instance, if the root of the query tree in Figure 1 is constrained by the expression $(s_1 \vee s_2)$, the s-term s_0 is matched by any subtree of the document tree in Figure 2 (a) where either s_1 or s_2 occurs (or both). The root slot of the CAA in Figure 3 then contains two occurrences of s_0 , namely &1 and &3.

The semantics of Boolean child constraints entails modifications of both the ranking model (see next paragraph) and the tree matching algorithm described in Section 3.2. During evaluation of an s-term s , the path join procedure $joinPath$ is only called when the child constraint β_s associated with s is a simple conjunction of child atoms (the default tree matching behaviour, see line 31 in Listing 1). Otherwise matches to all child subtrees below s are stored in the CAA by repeated calls to $matchPath$, as in line 29 of Listing 1. The child constraint β_s is evaluated after all children of s have been processed, i.e. immediately below line 20 in the incarnation for $r = s$, unless (1) β_s is a simple conjunction of child atoms, which is catered for by the calls to $joinPath$, or (2) β_s is a simple disjunction of child atoms, in which case all (even partial) occurrences stored in the slot a corresponding to r trivially match s . If none of these conditions holds, β_s is evaluated recursively for each occurrence in a . Subexpression involving a Boolean operator are interpreted recursively in the obvious way. As recursion anchor, an atom representing a child s-term s_c of s evaluates to \top iff the current occurrence of s in a is linked to one or more occurrences of s_c in the corresponding child slot of a .

As mentioned in Section 3.3, if an s-term s involving node labels occurs in a document node e , its local term frequency $tf_{s,e}^{loc}$ is fixed to 1 in the original s-term model. By contrast, when using Boolean child constraints we would like to quantify how close a – possibly partial – occurrence matches s . To this end, we redefine the local term frequency $tf_{s,e}^{loc}$ to be either 1 (see above) or equal to the number of atoms in β_s eval-

uating to \top for e , whatever value is greater. For instance, consider two occurrences e, e' of s and $\beta_s = (s_1 \vee s_2)$ for child terms s_1, s_2 of s . Assuming that only the subtree rooted at e contains occurrences of both s_1 and s_2 , $tf_{s,e}^{loc} > tf_{s,e'}^{loc}$ according to the new definition, which accounts for the fact that e satisfies more (non-Boolean) constraints specified by s . As a special case, all full occurrences of s (i.e., those which would satisfy a Boolean constraint of the form $(s_1 \wedge \dots \wedge s_j)$ where j is the number of children of s) have the same local term frequency. In this sense, our adaptation of the s-term model to Boolean child constraints generalizes the original model.

Order/distance constraints on keyword conjunctions.

We extended the query language of the X² system to support order and distance constraints on binary conjunctions of “flat” terms specified in a textual query node. For instance, the expression “XML [1,3] node” requires matching document nodes to contain an occurrence of the keyword “XML”, followed by an occurrence of the keyword “node”, such that both occurrences are separated by at most two tokens. Either value in the $[min, max]$ pair may be omitted; as special cases, “XML [,] node” specifies an ordered conjunction without distance constraints, and “XML node” is a shorthand for “XML [1,1] node” (i.e., a simple phrase search for directly adjacent tokens). Note that distance constraints in X² queries imply that the conjunction of terms is ordered. By contrast “XML , node” specifies an unordered conjunction of both terms (with arbitrary distance). Expressions of either type (i.e., “ $s_1 [min, max] s_2$ ” and “ s_1 , s_2 ”) may be chained together, implicitly forming a left-associative nested conjunction. The expression “XML [,3] node , IR [5,] rank”, e.g., selects document nodes containing (1) occurrences of “XML” and “node” in that order, separated by at most two tokens, and (2) an occurrence of “IR” anywhere in the textual content (possibly even between the former two occurrences), and (3) an occurrence of “rank” at least four tokens after the rightmost of the former three occurrences.

Since order and distance constraints apply only to term occurrences within the same document node, they are easily integrated with the index procedure $fetchOccurrences$. To this end, each occurrence of a “flat” term s in a document node e is associated with a list of *token position offsets* in the CA Table, omitted in Figure 2 (c), which indicate the number of tokens preceding the first occurrence of s in e as well as the number of tokens between any two consecutive occurrences of s in e . As an example, consider a leaf node whose textual content is “to be or not to be”. The respective lists of token position offsets of all keywords are: “to” $\langle 1, 4 \rangle$, “be” $\langle 2, 4 \rangle$, “or” $\langle 3 \rangle$, and “not” $\langle 4 \rangle$. (For the sake of the example, assume these terms are not treated as stop-words.) If a term occurs directly after a stop-word (which is not stored in the CA Table) or after a child node (in case of mixed content), the token position of that occurrence is incremented by one to avoid erroneous phrase matching.³

When matching a binary conjunction of the “flat” terms s_1 and s_2 , two CA Table entries are intersected to identify all nodes containing co-occurrences of both terms. Order and distance constraints are checked on the two lists of token position offsets associated with any document node in

³For instance, the expression “XML node” is matched neither by “XML and node” nor by “XML <i>root</i> node”. Obviously, phrase matching across mixed content may be desirable in cases such as “XML
 node”.

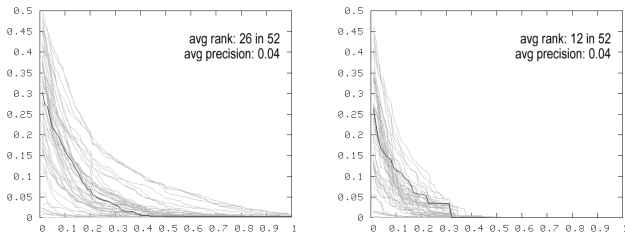


Figure 4: VCAS performance of the s-term model (left: avg. over all quantisations; right: best case).

the intersection. Those nodes satisfying the constraints for s_1 and s_2 keep only the list of token position offsets for the second term, reduced to the occurrences which justify the match. Thus in a nested expression involving a third term s_3 , the constraint check for the subsequent binary conjunction operates on (1) a list of token position offsets representing matches to the conjunction of s_1 and s_2 and (2) a list of token position offsets for s_3 . This corresponds to the aforementioned left-associative interpretation of chained term conjunctions.

4. EXPERIMENTS AND EVALUATION

We evaluated our implementation of the s-term vector space model at the third workshop of the *Initiative for the Evaluation of XML Retrieval (INEX)* [4] in 2004. Queries in both in the *Vague Content And Structure (VCAS)* and *Content Only (CO)* tracks were submitted to X² after automatic translation into the system’s query language. As general results, we observe that (1) the model performs reasonably well for structured queries (VCAS), occupying a position 26 among the 52 participants (position 12 in the best case), and (2) there is considerable room for optimizations which have not been considered yet. The plots in Figure 4 show the recall/precision graph⁴ for the s-term vector space model (black line) and all other participating approaches (shaded lines). The values in the left plot are averaged over all quantisation methods applied at INEX 2004, whereas the right plot shows only the quantisation for which the s-term model performs best (*RPng with overlap and strict quantisation*).

Fixed vs. flexible target element. Among the features presented in Section 3.4, the parameter with the greatest impact on ranking performance is the choice of the target element. In the course of this work we tested two simple strategies: either the target element is fixed to be identical with the query root (*fixed target element*), or the target element is determined in an XPath-style as the lowest structural query node outside predicates (*semi-flexible target element*). Figure 5 illustrates how performance degrades when only logical documents are returned as answers to VCAS queries (left column), compared to a semi-flexible choice of the target element (right column). While in the average over all quantisation methods the difference is nine positions, the impact is even higher (16 positions) for the *s3e32* quantisation which favours answers with high specificity (not shown in the figure). As could be expected, an-

⁴All plots are based on data provided by the Laboratoire d’Informatique de Paris 6 at inex.lip6.fr/2004/metrics/.

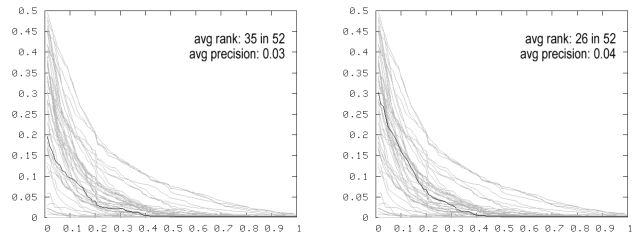


Figure 5: Fixed vs. flexible target element (VCAS).

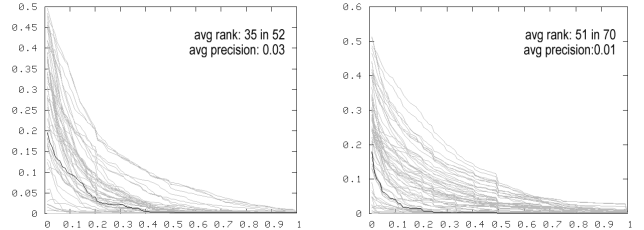


Figure 6: Structured (VCAS) vs. flat (CO) retrieval.

swers which are constantly at the article level are often fairly unspecific, including many irrelevant nodes.

Structured vs. flat retrieval. Finally, the plots in Figure 6 compare the performance of the s-term vector space model in the VCAS and CO tracks. Note that both plots are based on results for fixed target elements only, which explains the low overall precision of the s-term results. For CO queries, the target element was fixed to an article node containing the actual “flat” query. As can be seen in the right plot, the s-term model performs worse than almost 75% of all participants of the CO track, whereas in structured document retrieval (VCAS) it is closer to two thirds even without a flexible target element. This is not astounding given that the core concept of the model, the s-term, relies on the structure of the query. Obviously, when running “flat” text queries against structured documents the choice of the target element needs dedicated strategies since the query contains no structural hints as to which elements the user expects. Since the implementation of the s-term model as described in this work addresses this issue in a very naive way, we believe that there is considerable room for optimization here.

5. RELATED WORK

Previous work [18] describes in detail how to adapt the CADG index to four models for structured documents [19, 2, 15, 14]. In Section 3.1 of this work the CADG is modified along these lines in order to obtain an IR-CADG for the s-term model. In terms of the *Path/Term/Node (PTN) hierarchy* proposed in [18], which specifies how an index structure stores information for relevance ranking, the IR-CADG described in Section 3.1 contains only *Path/Term/Node*-specific information (local term frequencies in the CA Table).

As observed in [18], ranking models have different concepts of *idf* for structured documents. This issue is related to the document boundary problem mentioned in Section 1. In [18], two different types of *idf* are distinguished: *structured idf* counts only document satisfying structural and

textual query conditions, whereas *flat idf* is strictly term-specific. The s-term model, with its query-dependent notion of logical documents, features a structured *idf*. In [18] it is argued that under certain circumstances, this may help to match the user's information need more closely.

6. CONCLUSION AND FUTURE WORK

In this work, we extended the s-term vector space model [13] for ranked retrieval of structured documents, with a number of useful features such as Boolean constraints on tree queries, order and distance constraints on search terms (including phrase search), and the specification of target elements. We also described data structures and algorithms for the retrieval and ranking of structured documents using the s-term model in combination with the IR-CADG index [18, 17] and showed how the adaptation of state-of-the-art techniques for exact retrieval can complement our work, thus integrating effective ranking with efficient retrieval. Finally, we evaluated the ranking performance of our s-term implementation at the third INEX workshop 2004 [4]. The results show that while the model performs reasonably well for Vague Content And Structure (VCAS) queries in these first tests, there is also a fair potential for optimization, especially for Content Only (CO) queries. Future work on the model and the implementation may include:

- in the CO track, tests against the INEX collection where an unlabelled node instead of an article node is the fixed target element
- a truly *flexible target element* definition which dynamically determines document nodes to be retrieved as query results, even when no hints are given in the query (as in the CO case)
- more sophisticated biasing method for combining relevance scores of matches to logical document roots and target elements
- a substantial simplification of the s-term model in order to reduce the computational effort needed for relevance ranking

While the first three issues target the effectiveness of the model, the last point is motivated by the observation that during the computation of relevance scores possibly huge sets of s-term matches are joined with all logical document roots. We will examine how the number of nodes to be joined can be reduced without reducing the ranking effectiveness.

7. REFERENCES

- [1] S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, D. Srivastava, and Y. Wu. Structural Joins: A Primitive for Efficient XML Query Pattern Matching. In *Proc. 18th IEEE Int. Conf. on Data Engin.*, 2002.
- [2] N. Fuhr and K. Großjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In *Research and Development in Information Retrieval*, 2001.
- [3] R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proc. 23rd Int. Conf. on Very Large Data Bases*, 1997.
- [4] Initiative for the Evaluation of XML Retrieval (INEX), 2004. Organised by the DELOS Network of Excellence for Digital Libraries.
- [5] P. Kilpeläinen. *Tree Matching Problems with Applications to Structured Text Databases*. PhD thesis, University of Helsinki, 1992.
- [6] H. Meuss. *Logical Tree Matching with Complete Answer Aggregates for Retrieving Structured Documents*. PhD thesis, University of Munich, 2000.
- [7] H. Meuss, K. Schulz, and F. Bry. Towards Aggregated Answers for Semistructured Data. In *Proc. 8th Int. Conf. on Database Theory*, 2001.
- [8] H. Meuss and K. U. Schulz. Complete Answer Aggregates for Tree-like Databases: A Novel Approach to Combine Querying and Navigation. *ACM Transactions on Information Systems*, 19(2), 2001.
- [9] H. Meuss, K. U. Schulz, F. Weigel, S. Leonardi, and F. Bry. Visual Exploration and Retrieval of XML Document Collections with the Generic System X^2 . *Journal of Digital Libraries, Special Issue on Information Visualization Interfaces*, 2004.
- [10] R. Sacks-Davis, T. Arnold-Moore, and J. Zobel. Database Systems for Structured Documents. In *Proc. Int. Symposium on Advanced Database Technologies and Their Integration*, 1994.
- [11] G. Salton. *The SMART Retrieval System – Experiments in Automatic Document Processing*. Prentice Hall Inc., Englewood Cliffs, NJ., 1971.
- [12] T. Schlieder. Similarity Search in XML Data using Cost-Based Query Transformations. In *Proc. 4th Intern. Workshop on the Web and Databases*, 2001.
- [13] T. Schlieder and H. Meuss. Querying and Ranking XML Documents. *Special Topic Issue Journal American Society for Informations Systems on XML and Information Retrieval*, 53(6), 2002.
- [14] D. Shin, H. Jang, and H. Jin. BUS: An Effective Indexing and Retrieval Scheme in Structured Documents. In *Proc. 3rd ACM Int. Conf. on Digital Libraries*, 1998.
- [15] A. Theobald and G. Weikum. The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking. In *Proc. 8th International Conf. on Extending Database Technology*, 2002.
- [16] A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I. Available at inex.is.informatik.uni-duitburg.de:2004.
- [17] F. Weigel, H. Meuss, F. Bry, and K. U. Schulz. Content-Aware DataGuides: Interleaving IR and DB Indexing Techniques for Efficient Retrieval of Textual XML Data. In *Proc. 26th European Conf. on Information Retrieval*, 2004.
- [18] F. Weigel, H. Meuss, K. U. Schulz, and F. Bry. Content and Structure in Indexing and Ranking XML. In *Proc. 7th Int. Workshop on the Web and Databases*, 2004.
- [19] J. E. Wolff, H. Flörke, and A. B. Cremers. Searching and Browsing Collections of Structural Information. In *Proc. IEEE Forum on Research and Technology Advances in Digital Libraries*, 2000.

Component ranking and Automatic Query Refinement for XML Retrieval

Yosi Mass, Matan Mandelbrod
IBM Research Lab
Haifa 31905, Israel
{yosimass, matan}@il.ibm.com

Abstract

Queries over XML documents challenge search engines to return the most relevant XML components that satisfy the query concepts. In a previous work[6] we described an algorithm to retrieve the most relevant XML components that performed relatively well in INEX'03. In this paper we show an improvement to that algorithm by introducing a document pivot that compensates for missing terms statistics in small components. Using this new algorithm we achieved improvements of 30%-50% in the Mean Average Precision over the previous algorithm. We then describe a general mechanism to apply existing Automatic Query Refinement (AQR) methods on top of our XML retrieval algorithm and demonstrate a particular such method that achieved top results in INEX'04.

Keywords

XML Search, Information Retrieval, Vector Space Model, Automatic Query Refinement.

1 Introduction

While in traditional IR we are used to get back entire documents for queries, the challenge in XML retrieval is to return the most relevant components that satisfy the query concepts. The INEX initiative[4] sub classified this task into two sub tasks; Content only (CO) topics and Content and Structure (CAS) topics. In a CO task the user specifies queries in free text and the search engine is supposed to return the most relevant XML components that satisfy the query concepts. In a CAS task the user can limit query concepts to particular XML tags and to define the desired component to be returned using XPath[10] extended with an about() predicate.

In order to realize the problem in ranking XML components we first examine a typical class of IR engines that use *tf-idf* [8] to perform document ranking. Those engines maintain an inverted index in which they keep for each term among other things the number of documents in which it appear (*df*) and its number of occurrences in each document in the collection (*tf*). Then this statistics is used to estimate the relevance of a document to the query by measuring some distance between the two.

To be able to return a component instead of a full document search engines should modify their data structures to keep

statistics such as *tf-idf* at the component level instead of at the document level. This is not a straight forward extension since components in XML are nested and the problem is how to keep statistics at the component level such that it handles components nesting correctly.

In INEX'03 we described a method for component ranking by creating separate indices for the most informative component types in the collection as described in [6]. For example we created an index for full articles, an index for all sections, for all paragraphs etc. This approach solved the problem of statistics of nested components since in each index we have now components from same granularity so they are not nested.

While this approach solved the problem of nested components it introduced a deficiency that could distort index statistics. The problem is that the fine grained indices lack data that is outside their scope which is not indexed at all. For example the *articles* index contains 42,578,569 tokens while the *paragraphs* index contains only 31,988,622 tokens. This means that in the *paragraphs* index ~25% of the possible statistics is missing so for example a term with a low *df* based on the indexed tokens may actually be quite frequent outside the paragraphs so its actual *df* should be higher.

In this paper we describe a method to compensate for this deficiency using document pivot. Using this method we got a consistent improvement of 30%-50% in the mean average precision (MAP) for both INEX'03 and INEX'04 CO topics.

On top of this improvement we achieved further improvement by applying Automatic Query Refinement (AQR) on our XML component retrieval system. AQR was studied in [7] in the context of traditional IR engines. The idea there is to run the query in two rounds where highly ranked results from the first round are used to add new query terms and to reweigh the original query terms for the second round. We show how to adopt such AQR algorithms on top of our XML component ranking algorithm.

The paper is organized as follows – in section 2 we describe the document pivot concept and in section 3 we describe how to adopt AQR methods from traditional IR to XML retrieval systems. In section 4 we describe our

inverted index and our CO and VCAS runs. We conclude in section 5 with discussion of the approaches and with future directions.

2 Component ranking with Document Pivot

We start by briefing our component ranking approach from INEX'03 as described in [6] and then we show how it was improved using the document pivot concept.

As discussed above the problem in XML component ranking is how to keep statistics at the component level such that it handles components nesting correctly.

In [6] we solved that problem by creating different indices for the most informative component types. We created an index for articles, index for sections, index for sub sections and index for paragraphs. For simplicity we discuss now our approach for CO topics.

For a given CO topic we run the query in parallel on the set of indices and get back a sorted result set from each index with components of that index. So we get a sorted list of articles, a sorted list of section and so on.

We then described a method for comparing the different result sets so that we can merge the sets into a single sorted list of all component types. Why do we get different scores in each result set? Our scoring method is based on the vector space model where both the query Q and each document d are mapped to vectors in the terms space and the relevance of d to Q is measured as the cosine between them using the *tf-idf* statistics as described in Figure 1 below -

$$score(Q, d) = \frac{\sum_{t_i \in Q \cap d} w_Q(t_i) * w_d(t_i) * idf(t_i)}{\|Q\| * \|d\|}$$

$$w_Q(t) = \frac{\log(TF_Q(t))}{\log(AvgTF_Q)} \quad w_D(t) = \frac{\log(TF_d(t))}{\log(AvgTF_d)}$$

$$idf(t) = \log\left(\frac{\# DocumentsIntheCollection}{\# DocumentsContaining(t)}\right)$$

Figure 1 – Document scoring function

$TF_Q(t)$ is the number of occurrences of t in Q and $TF_d(t)$ is the number of occurrences of t in d .

$AvgTF_Q$ is the average number of occurrence of all query terms in Q and $AvgTF_d$ is the average number of occurrence of all terms in d .

$\|Q\|$ is the number of unique terms in Q and $\|d\|$ is the number of unique terms in d , both scaled by the average document length in the collection.

It can be seen that while scores of components in each index are comparable to each other, scores in different indices are at a different scale. For example the *articles*

index has 12,107 components so the *idf* of a relatively rare term is not very large compared to its *idf* in the *paragraphs* index which has 646,216 components. In addition the average document length in the *articles* index is 900 while the average document length in the *paragraphs* index is 37. Since $\|d\|$ and $\|Q\|$ are scaled by the average document length then the denominator of scores in the *paragraphs* index is much lower than in the *articles* index. Combining the *idf* difference and the length normalization difference shows why scores of components in the *paragraphs* index are relatively higher than scores of components in the *articles* index.

In order to compare the scores in different result sets we described in [6] a normalization formula that ensures absolute numbers that are index independent. This is achieved by each index computing $score(Q, Q)$ which is the score of the query itself as if it was a document in the collection. Since the score measures the cosine between vectors, then the max value is expected between two identical vectors. Each index therefore normalizes all its scores to its computed $score(Q, Q)$. The normalized results are then merged into a single ranked list consisting of components of all granularities.

While the approach of creating independent indices solved the problem of overlapping data it introduced another deficiency of missing data. The fine grained indices lack data that is outside their scope which is not indexed. For example the *articles* index contains 42,578,569 tokens while the *paragraphs* index contains only 31,988,622 tokens. The missing data in the fine grained indices can distort the *idf* statistics of the collection and therefore may affect the quality of the results.

To fix that problem we use this year a concept first mentioned in [9] which uses a document pivot (DocPivot) factor to scale the final component score by the score of its containing article. The final score of a component with original score S_c and with its full article score S_a is then

$$DocPivot * S_a + (1 - DocPivot) * S_c.$$

Assuming that the full *articles* index is the first index then the overall algorithm to return a result set for a given query Q is given in Figure 2 below. Step 3 is the new step introduced by the DocPivot.

For each index i

1. Compute the result set R_i of running Q on index i
2. Normalize scores in R_i to $[0,1]$ by normalizing to $\text{score}(Q,Q)$
3. 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

Figure 2 - Component ranking algorithm

We experimented with several values of DocPivot on the 2003 CO topics using the `inex_eval` with strict metric and got the graph marked as 2003 in Figure 3 below.

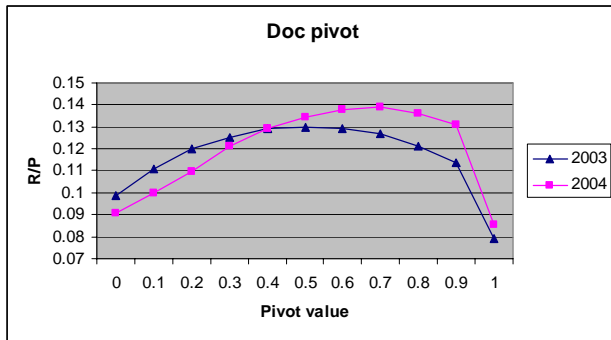


Figure 3 - Doc pivot on 2003/2004 data

The MAP with DocPivot = 0 is the result we achieved in our 2003 official submission. We can see that with DocPivot=0.5 we get improvements of 31% over the base 2003 run so we used that value for our 2004 runs.

Later when the 2004 assessments were available we tried those values on the 2004 CO topics using the aggregate metric and got the best MAP for DocPivot = 0.7 (the 2004 graph in Figure 3) which is 52% improvements over the base run with no DocPivot.

3 Automatic Query Refinement for XML

In this section we describe how to apply Automatic Query Refinement (AQR) on top of our XML ranking algorithm. AQR was studied in [7] in the context of traditional IR engines. The idea is to run the query in two rounds where highly ranked results from the first round are used to add new query terms and to reweigh the original query terms for the second round. We show now a method to adopt such AQR algorithms on top of our XML component ranking algorithm.

Assume we have an AQR algorithm that can be used to refine query results. Since we have separate indices for different component granularities we can run the AQR algorithm on each index separately. The modified XML component ranking algorithm is described in Figure-4.

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 $\text{score}(Q,Q)$
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

Figure 4 -Component ranking with AQR

We add step 2 which is the query refinement step and the rest of the algorithm continuous as in the simple case by normalizing and scaling scores in each index and finally merging the result sets.

We describe now a specific AQR algorithm that we used in INEX'04 and discuss some variants of its usage in our XML component ranking algorithm.

The AQR algorithm we used is described in [2]. The idea there is to add Lexical Affinity (LA) terms to the query where a Lexical Affinity is a pair of terms that appear close to each other in some relevant documents such that exactly one of the terms appears in the query.

The AQR is based on the information gain (IG) obtained by adding lexical affinities to the query. The IG of a lexical affinity L on a set of documents D with respect to a query Q denotes how much L separates the relevant documents in D from the non relevant documents for the query Q . IG is defined as:

$$IG_{Q,D}(L) = H_Q(D) - \left[\frac{|D^+|}{|D|} H_Q(D^+) + \frac{|D^-|}{|D|} H_Q(D^-) \right]$$

Figure 5 - Information Gain of a Lexical Affinity

where $H_Q(D)$ is the entropy (or degree of disorder) of a set of documents D , $D^+ \subseteq D$ is the set of documents containing L and $D^- \subseteq D$ is the set of documents not containing L . $H_Q(D)$ is defined as

$$H_Q(D) = -p_Q(D) \log(p_Q(D)) - (1-p_Q(D)) \log(1-p_Q(D))$$

Figure 6 - Entropy of a group

where $p_Q(D)$ is the probability of a document chosen randomly from D to be relevant to Q and is equal to $\frac{|R|}{|D|}$

where $R \subseteq D$ is the subset of relevant documents. Since we don't have the relevant documents R we use the scoring

function as an approximation for $P_Q(D^+)$ by taking sum of scores of documents in D^+ divided by $(100 \times \text{number of docs in } D^+)$. We do the same estimation for $P_Q(D)$

The AQR procedure works as follows: It gets the result set obtained by running the search engine on the query Q (algorithm step 1 in Figure-4) and additional 4 parameters (M, N, K, α) that are explained below. The AQR first constructs a list of candidate LAs that appear in the top M highly ranked documents from the result set. Then it takes D to be the set of the top N ($N \gg M$) highly rank documents and finds the K LAs with the highest IG on that set D .

Those LAs are used to re rank each document d in the result set by adding their *tf-idf* contribution to $score(Q, d)$ as given by Figure 1. Since they don't actually appear in the query Q we take their $TF_Q(L)$ to be the given parameter α .

We can have several variants for using the above AQR algorithm in the XML component ranking algorithm -

1. The AQR procedure can be applied on each index separately using same (M, N, K, α) parameters or index specific (M, N, K, α) parameters. In this variant different LAs are added to the query for each index.
2. We can apply the first part of the AQR using (M, N, K) on the full *articles* index to find the best LAs. Then apply the last part of the AQR that does the re ranking (with the parameter α) on each index. using the LAs that were extracted from the *articles* index.

The motivation for the 2nd variant is that most informative LAs can be obtained on the full *articles* index since it has the full collection data. In section 4 we describe the (M, N, K, α) parameters used in our runs.

4 Runs description

4.1 Index description

Similar to last year we have created six inverted indices for the most informative components which are {article, sec, ss1, ss2, {p+ip1}, abs}. We removed XML tags from all indices except from the {article} index where they were used for checking VCAS topic constraints. Content was stemmed using a Porter stemmer and components with content smaller than 15 tokens were not indexed in their corresponding index.

4.2 CO topics

Each CO topic has 4 parts : <title>, <description>, <narrative> and <keywords>. This year we could use only the <title> for formulating the query to our search engine. Due to the loosely interpretation of topics as appear in [5] we ignored '+' on terms and we ignored phrase boundaries and use the phrase's terms as regular terms. We still treated

'-' terms strictly namely components with '-' terms were never returned.

For example topic 166

```
<title>+"tree edit distance" + XML - image </title>
```

is executed as

```
tree edit distance XML -image
```

We submitted three runs where two of them were ranked 1st and 2nd among the official INEX CO runs. See table 1 below-

TASK:CO			
rank	Institute	avg	overlap(%)
1.	IBM Haifa Research Lab(CO-0.5-LAREFIENMENT)	0.1437	80.89
2.	IBM Haifa Research Lab(CO-0.5)	0.1340	81.46
3.	University of Waterloo(Waterloo-Baseline)	0.1267	76.32
4.	University of Amsterdam(UAms-CO-T-FBack)	0.1174	81.85
5.	University of Waterloo(Waterloo-Expanded)	0.1173	75.62
6.	Queensland University of Technology(CO_PS_Stop50K_099_049)	0.1073	75.89
7.	Queensland University of Technology(CO_PS_099_049)	0.1072	76.81
8.	IBM Haifa Research Lab(CO-0.5-Clustering)	0.1043	81.10
9.	University of Amsterdam(UAms-CO-T)	0.1030	71.96
10.	LIP6(simple)	0.0921	64.29

Table 1 - CO results

4.2.1 Doc Pivot run

In the run titled CO-0.5 we implemented the Component ranking algorithm as described in Figure-2 using DocPivot=0.5. This run was ranked 2nd in the aggregate metric.

4.2.2 AQR run

In the run titled CO-0.5-LAREFIENMENT we implemented our AQR algorithm from Figure-4 using $M=20, N=100, K=5$ and $\alpha=0.9$ on all indices. We have implemented the first algorithm variant where each index computes its own LAs to add. This run was ranked 1st using the CO aggregate metric. We leave for future work

experiments with more parameter settings with the two algorithm variants.

Some example LAs that were added to queries:

For topic 162:

```
Text and Index Compression Algorithms
```

We got LA pairs (compress, huffman), (compress, gigabyte), (index, loss)

For topic 169:

```
+"Query expansion" +"relevance feedback" +web
```

We got (query, search), (relevance, search), (query, user), (query, result)

4.3 VCAS topics

We applied an automatic translation from XPath[10] to XML Fragments[1] which is the query language used in our JuruXML search engine. XML Fragments are well-formed XML segments enhanced with

- '+/-' on XML tags and on content
- Phrases on content (" ")
- Parametric search on XML tag's value
- An empty tag (<>) that is used as parenthesis.

We can view any XML Fragment query as a tree¹ with the semantics that at each query node, '+' children must appear, '-' children should not appear and others are optional and only contribute to ranking. If a node doesn't have '+' children then at least one of its other (non '-') children must appear.

For example the query

```
<article>
  <abs>classification</abs>
  <sec>experiment compare</sec>
</article>
```

will return articles with *classification* under <abs> **or** with *experiment* or *compare* under <sec>. Note that the default semantics in XML Fragments is OR unless there are '+'s.

The same query with '+' on the tags -

```
<article>
  +<abs>classification</abs>
  +<sec>experiment compare</sec>
</article>
```

will return articles with *classification* under <abs> **and** with *experiment* or *compare* under <sec>.

Finally the query

```
<article>
```

¹ For a query given as several disjoint fragments we add a dummy root node to make the all query a valid XML data.

```
+<abs>classification</abs>
  <sec>experiment compare</sec>
</article>
```

will return only articles with *classification* under <abs>. Articles with *experiment* or *compare* under <sec> will be returned with higher ranking since the child <sec>experiment compare</sec> is optional.

The empty tag is used as a kind of parenthesis so the query

```
<title>
  <>+network +security</>
  <>+database +attributes</>
</title>
```

will return documents with *network* and *security* **or** with *database* and *attributes* under the <title> while the query

```
<title>
  +<>network security</>
  +<>database attributes</>
</title>
```

will return documents with *network* or *security* **and** with *database* or *attributes* under its <title>.

The automatic transformation from an XPath expression of the form

```
//path1[path1Predicates]//path2[path2Predicates]
```

to XML Fragments works as follows: It first creates a query node <path1> with two children: The first is a mandatory empty tag (+<>) surrounding path1Predicates and the second is the node <path2> prefixed with a '+'. The path1 and path2 Predicates are translated to nodes where 'about' predicates for the current node ('about(., "text")') are transformed to just *text* and about predicates for sibling nodes ('about(//path, "text")') are transformed to <path>text</path>. For example the INEX CAS topic 131

```
<title>//article[about(//au,"Jiawei Han")]//abs[about(., "data mining")]</title>
```

is translated to the following XML Fragments query -

```
+<article>
  +<>
    <au>"jiawei han"</au>
  </>
  +<abs>
    +<>
      "data mining"
    </>
  </abs>
</article>
```

Figure 7 - Automatic translation to XML Fragments

To support AND/OR between XPath predicates we use the empty tag where predicates that are ANDed are transformed to XML Fragments under '+<>' tag and predicates that are ORed are transformed to XML Fragments under <> with no prefix. For example topic 134 -

```
<title>//article[(about(., "phrase search") OR about(., "proximity search")) OR about(., "string matching")) AND (about(., tries) OR
```



```
about(., "suffix trees") OR about(., "PAT arrays")//sec[about(., algorithm)]</title>
```

is transformed to

```
<article>
  +< >
    +< >
      <>"phrase search"</>
      <>"proximity search"</>
      <>"string matching"</>
    </>
    +< >
      <>"tries"</>
      <>"suffix trees"</>
      <>"pat arrays"</>
    </>
  </>
  +<sec>
    +<>algorithm</>
  </sec>
</article>
```

Figure 8 - Translation of AND and OR

For the INEX VCAS topics and to support the INEX guideline for vagueness as appear in [5] we ignored the '+' on tags and similar to the CO case we ignored '+' on content and phrase boundaries. Ignoring '+' changes everything to OR semantics therefore the empty tags have no meaning and can be ignored. For example the above topic 131 is then equivalent to -

```
<article>
  <au>jiawei han</au>
  <abs>data mining</abs>
</article>
```

Figure 9 - The query as run by our system

We still keep the XML Fragments semantics that nodes with a single child must have that child so the above query will return only results which have *jiawei* or *han* under <au> or that have *data* or *mining* under the <abs>.

To decide which element to return we followed the XPath target element semantics that defines the last element in the XPath expression as the element to be returned up to the equivalent tags as defined in [5]. We run the VCAS topics using a minor modification of step 1 in the algorithm in Figure-2 above: The *articles* index in addition to creating its result set also check the query constraints and mark valid components to be returned. The other indices then return in their results set only components that were marked valid by the *articles* index.

Obeying the target element constraint resulted in a low 38% overlap and as a result with low MAP of 0.065 in the aggregate *inex_eval*. It seems like assessors ignored the target elements as for example in the above topic 131 full articles were assessed as most exhaustive and most specific for that topic.

4.4 NLP runs

We submitted one CO run and one VCAS run. For the CO run we used the topic's <description> part and just applied the algorithm from Figure-2 with DocPivot=0.5. This run got MAP of 0.1286 using the aggregate *inex_eval*. For the VCAS run we similarly used the topic's <description> with same DocPivot but ignored the XPath target element as if it was a CO topic. This run got MAP of 0.05.

5 Discussion

We have presented two extensions to our last year XML component ranking algorithm. The first extension introduces a document pivot that scales scores of components by the score of their containing article. This method achieved improvements of 31% over our base CO run in INEX'03 and 52% over our base CO run in INEX'04. We then described an algorithm to apply existing AQR algorithms on top of our XML component ranking algorithm and demonstrated an example such AQR method using Lexical Affinities with Maximal Information Gain. Our two runs that implemented those extensions were ranked 1st and 2nd in the CO track. The space of possible AQR parameter combinations and the variants for their usage in XML is quite large and we still have to explore the best combination that would give best results.

6 Acknowledgment

We would like to thank the INEX organizers for the assessment tool and for the *inex_eval* tool they have supplied.

7 References

- [1] A. Z. Broder, Y. Maarek, M. Mandelbrod and Y. Mass (2004), "Using XML to Query XML – From Theory to Practice". In proceedings of RIAO'04, Avignon France, Apr , 2004.
- [2] D. Carmel, E. Farchi, Y.Petruschka, A. Soffer, Automatic Query Refinement using Lexical Affinities with Maximal Information Gain. Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002.
- [3] D. Carmel, Y. Maarek, M. Mandelbrod, Y. Mass, A. Soffer, Searching XML Documents via XML Fragments, SIGIR 2003, Toronto, Canada, Aug. 2003
- [4] INEX, Initiative for the Evaluation of XML Retrieval, <http://inex.is.informatik.uni-duisburg.de>
- [5] INEX'04 Guidelines for Topic Development, <http://inex.is.informatik.uni-duisburg.de:2004/internal/pdf/INEX04TopicDevGuide.pdf>
- [6] 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
- [7] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems, Knowledge Engineering Review, 18(1):2003.
- [8] G. Salton, Automatic Text Processing – The Transformation, Analysis and Retrieval of Information by Computer, Addison Wesley Publishing Company, Reading, MA, 1989.
- [9] B. Sigurbjornsson, J. Kamps, M. Rijke, An element based approach to XML Retrieval, Proceedings of the Second Workshop of the Initiative for The Evaluation of XML Retrieval (INEX), 15-17 December 2003, Schloss Dagstuhl, Germany, pg 19-26.
- [10] XPath – XML Path Language (XPath) 2.0, <http://www.w3.org/TR/xpath2/>

TIJAH at INEX 2004

Modeling Phrases and Relevance Feedback

Vojkan Mihajlović¹

Georgina Ramírez²

Arjen P. de Vries²

Djoerd Hiemstra¹

Henk Ernst Blok¹

¹CTIT
P.O. Box 217
7500 AE Enschede
The Netherlands
{v.mihajlovic, d.hiemstra,
h.e.blok}@utwente.nl

²CWI
P.O. Box 94079
1090GB Amsterdam
The Netherlands
{georgina, arjen}@cwi.nl

ABSTRACT

This paper discusses our participation in INEX using the TIJAH XML-IR system. We have enriched the TIJAH system with several new features. An extensible conceptual level processing unit has been added to the system, following a standard layered database architecture. The algebra on the logical level and the implementation on the physical level have been extended to support phrase search and *structural* relevance feedback. The conceptual processing unit is capable of rewriting NEXI content-only and content-and-structure queries into the internal form, based on the retrieval model parameter specification, that are either pre-defined or based on a relevance feedback. Relevance feedback parameters are produced based on the data fusion of result element score values and sizes, and relevance assessments. The introduction of new operators supporting phrase search in score region algebra on logical level is discussed in the paper, as well as their implementation on physical level using the pre-post numbering scheme. The framework for structural relevance feedback is also explained in the paper. We conclude with a preliminary analysis of the system performance based on INEX 2004 runs.

1. INTRODUCTION

In our research for INEX 2004 we extended our TIJAH system to support more advanced IR techniques, namely phrase search and relevance feedback. The TIJAH system follows a layered database architecture, consisting of a conceptual, a logical and a physical level. Each level has been built upon a different data model and has its own operators. The top level is based on the NEXI query language. A NEXI query is first translated (at the conceptual level) into an internal query representation that resembles the NEXI query language closely, but enriched with some additional operators. The translation process is based on the retrieval model specification. The conceptual query is then transformed into a score region algebra (SRA) query plan [8] on the logical level of the TIJAH system. SRA views XML as a collection of regions and not as a tree-like structure, and operators in the SRA are based on the region containment relation and on region frequency counts for supporting vague containment conditions. The logical query plan is transformed into the physical plan (via Monet interpreter language - MIL) which is executed in the MonetDB database kernel [2].

The TIJAH system that we use for INEX 2004 is an extended version of the TIJAH system used in 2003 [7]. Each level of the TIJAH database system is extended to support phrase search and relevance feedback. Thus, the conceptual level is capable of handling phrases and supports relevance feedback specification. New operators are introduced into the score region algebra to support phrase modeling and relevance feedback specification and the physical level is enriched with new functions that implement phrase search. Furthermore, a fully automatic query rewriting unit is developed at the conceptual level capable of transforming original NEXI queries into proper conceptual queries based either on the retrieval model specification or on the relevance feedback data.

The retrieval model used for the NEXI *about* function is essentially the same as that used for INEX 2003 [7]. We calculate the relevance of a document component (i.e., XML element), following the idea of independence between the probability of relevance on exhaustivity and the probability of relevance on specificity. The relevance on exhaustivity is estimated using the language modeling approach to information retrieval [5]. The phrase model is kept orthogonal to the unigram language model for single terms, similarly to [10], and we used variants of the n-gram ($n > 1$) language model to see if and in what degree phrases can contribute to the TIJAH system performance. The relevance on specificity is assumed to be related to the component length (e.g., following a log-normal distribution).

This paper presents our approaches for two out of four tracks defined for INEX 2004, namely ad-hoc track and relevance feedback track. For the ad-hoc track, we developed approaches for both content-only (CO) and vague content-and-structure (VCAS) subtasks. Different models have been implemented in the TIJAH system for these subtasks. Moreover, the TIJAH system supports the specification of relevance feedback parameters and a simple model for relevance feedback on structure is implemented in our system.

The following section gives a global overview of the TIJAH system architecture. Section 3 describes the capabilities of a conceptual level of our system performing different NEXI query rewriting and expansions. Section 4 specifies an exten-

sion of score region algebra for phrase handling and explains how these expressions are mapped into efficient operations on physical level. Section 5 describes the incorporation of relevance feedback on structure in our system. The paper concludes with a discussion of the experiments performed with the TIJAH system for the two INEX ad-hoc search tasks (CO and CAS) and for INEX relevance-feedback task.

2. TIJAH SYSTEM ARCHITECTURE

The TIJAH XML-IR system follows a traditional three-level database architecture consisting of conceptual, logical and physical level. Although the concept has been well known in the database field for more than thirty years, we introduced some modifications in the architecture to bridge the gap between the traditional DBMS systems and IR systems.

2.1 Conceptual level

As a base for the conceptual level we used the Narrowed Extended XPath (NEXI) query language [11] as proposed by the INEX Initiative in 2003. NEXI query language supports only a subset of XPath syntax and extends XPath with a special *about* function that ranks XML elements by their estimated relevance to a textual query. As such, the invocation of the *about* function can be regarded as the instantiation of a retrieval model.

Throughout the paper we will use two NEXI examples, one taken from the INEX CAS topic 149:

```
//article[about(../(abs|kwd), "genetic algorithm")]
//bdy//sec[about(., simulated annealing)]
```

and the other from INEX CO topic 166:

```
+"tree edit distance" +XML -image
```

During the conceptual query processing a NEXI query language expression is encoded into an internal representation which is much like the original query, and all the manipulations are done on this internal representation. As a result of the processing on the conceptual level we obtain a conceptual query plan (see Figure 1).

2.2 Logical level

The difference on the logical level of traditional DBMSs and our system is in that we enhanced it with an algebra that takes into account the specific structure of the modeled data, i.e., XML in our case, to enable high level reasoning about the query specification and algebraic optimization. Since the algebra supports region score manipulation and ranked retrieval we named it score region algebra (SRA). As already stated in [7], considering XML as a sequence of tokens, modeling it in region algebra is straightforward. Furthermore, the exact retrieval model does not have to be completely specified, as the structural and semantic aware framework of score region algebra can be kept abstract regarding the exact retrieval model used. Finally, the properties of SRA operators can be used for query optimization as can be seen in [8].

The basic score region algebra operators that involve score manipulations are depicted in Table 1¹. We assume that the

¹We used a slightly different notation than in [7]. For more extensive coverage of score region algebra we refer to [8].

default value for score attribute is 1. Note that the probabilistic containment operators \sqsupset_p , \sqsupseteq_p , \blacktriangleright and \blacktriangleleft produce all regions from the first operand (R_1) as a result, i.e., the region start, end, type and name attribute values are copied from the left operand region set to the result region set, while the score attribute of the result set (p) gets its value based on the containment relation among regions in the left and regions in the right operand as well as their respective score values. The definitions of the probabilistic set-like operators (\sqcap_p and \sqcup_p) are similar to the definitions of basic set intersection and set union operators, i.e., the result region start, end, type and name are obtained the same way as for set intersection and union operators, except that the result score value for regions is defined based on the score values of regions in the left and right operand region set.

In the definition of score operators we introduced four complex scoring functions: f_{\sqsupset} , f_{\sqsupseteq} , f_{\blacktriangleright} and f_{\blacktriangleleft} , as well as two abstract operators: \otimes and \oplus , that define the retrieval model. For the \oplus operator we assume that there exist a default value for the score (denoted with d), and in case the region r_1 is not present in the region set R_2 the score is computed as $p = p_1 \oplus d$ and in case the region r_2 is not present in the region set R_1 the score is computed as $p = d \oplus p_2$.

The functions f_{\sqsupset} , f_{\sqsupseteq} , f_{\blacktriangleright} and f_{\blacktriangleleft} , applied to a region r_1 and a region set R_2 , result in the numeric value that takes into account the score values of all regions r_2 ($\in R_2$) and the numeric value that reflects the structural relation between the region r_1 and the region set R_2 . The abstract \otimes operator specifies how scores are combined in an **and** expression, while the \oplus operator defines the score combination in an **or** expression inside the NEXI predicate. The exact instantiation of these functions and operators is done on the physical level as can be seen in the next section.

2.3 Physical level

The SRA algebra is defined as a XML specific logical algebra and can be easily implemented with relational operators [7, 12, 3]. Since we used the MonetDB on the physical level the last step on logical level of the TIJAH system was a translation to Monet Interpreter Language (MIL). The MIL query plan is executed using MIL primitives that define the manipulation over Monet binary tables (BATs) [2].

Here we explain the implementation of the logical operators on the physical level based on a language modeling approach. The physical level is based on a pre-post numbering scheme [4] and the containment join operators (\bowtie_{\sqsupset} and \bowtie_{\sqsupseteq}) introduced in [7]. The context region in which we perform frequency counts is denoted by r , while the set R specifies the set of term regions.

In the specification of our retrieval model we first introduce three auxiliary functions at the physical level. These functions are used to compute the term frequency - $tf(r, R)$, the collection frequency - $cf(R)$ and the length prior - $lp(r)$. Variable λ represents the smoothing parameter for the inclusion of background statistics, μ is the mean value (i.e., of the logarithm distribution of the desired size for the element) and ρ is the variance (in our case set to 1) for the log-normal prior. These auxiliary functions can be implemented using two physical operators: a size operator $size(r)$ that returns

Table 1: Region algebra operators for score manipulation.

Operator	Operator definition
$\sigma_{t=type, n=name}(R)$	$\{r r \in R \wedge t = type \wedge n = name\}$
$R_1 \sqsupset_p R_2$	$\{r r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge p := p_1 \times f_{\sqsupset}(r_1, R_2)\}$
$R_1 \not\supset_p R_2$	$\{r r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge p := p_1 \times f_{\not\supset}(r_1, R_2)\}$
$R_1 \blacktriangleright R_2$	$\{r r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge p := p_1 \times f_{\blacktriangleright}(r_1, R_2)\}$
$R_1 \blacktriangleleft R_2$	$\{r r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge p := p_1 \times f_{\blacktriangleleft}(r_1, R_2)\}$
$R_1 \sqcap_p R_2$	$\{r r_1 \in R_1 \wedge r_2 \in R_2 \wedge (s_1, e_1, n_1, t_1) = (s_2, e_2, n_2, t_2) \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge p := p_1 \otimes p_2\}$
$R_1 \sqcup_p R_2$	$\{r r_1 \in R_1 \wedge r_2 \in R_2 \wedge ((s, e, n, t) := (s_1, e_1, n_1, t_1) \vee (s, e, n, t) := (s_2, e_2, n_2, t_2)) \wedge p := p_1 \oplus p_2\}$

the size of a selected region r , and a count operator $|R|$ that returns the number of regions in a region set R .

Function $tf(r, R)$ computes the term frequency for terms in R . The term frequency of a region set R (i.e., a set of term positions) is computed as:

$$tf(r, R) = \frac{|R \bowtie_{\square} r|}{size(r)}$$

Function $cf(R)$ computes the collection frequency of a *term* as follows:

$$cf(R) = \frac{|\sigma_{t=term}(W)|}{size(Root)},$$

where *term* is the name of a region $r \in R$, and *Root* represents the region that is not contained by any other region in the collection (i.e., the root region of the entire XML collection).

To define the length prior of the region r we used either the size of the element: $lp(r) = size(r)$, the standard element prior: $lp(r) = \log(size(r))$, or the log-normal distribution:

$$lp(r) = \frac{e^{-(\log(size(r)) - \mu)^2 / 2\rho^2)}}{size(r)\rho\sqrt{2\pi}}$$

Although in our framework arbitrary implementations of complex functions can be introduced on the physical level, for INEX 2004 we followed the language model [5] and the conclusion drawn from numerous experiments last year [7]. The complex scoring functions defined in our region algebra, $f_{\sqsupset}(r, R)$ and $f_{\not\supset}(r, R)$, implement the about function specified in NEXI, while $f_{\blacktriangleright}(r, R)$ and $f_{\blacktriangleleft}(r, R)$ specify the score propagation in nested regions.

$$\begin{aligned} f_{\sqsupset}(r, R) &= \lambda tf(r, R) + (1 - \lambda)cf(R) \\ f_{\not\supset}(r, R) &= 1 - tf(r, R) \\ f_{\blacktriangleright}(r, R) &= \frac{\sum_{r_i \in r \bowtie_{\square} R} (size(r_i) * p_i)}{\sum_{r_i \in r \bowtie_{\square} R} size(r_i)} \\ f_{\blacktriangleleft}(r, R) &= \sum_{r_i \in r \bowtie_{\square} R} p_i \end{aligned}$$

In this paper we take the simple approach for the score combination operators where \otimes is implemented as a product of two score values, while \oplus is the sum of scores (with the default value $d = 1$), as it shows good behavior for retrieval.

3. QUERY REWRITING

Upgrading the TIJAH system used previous year for INEX [7], we developed a fully automatic approach for translating NEXI queries first into internal conceptual representation, and later into logical algebra. The structure of the conceptual query translator is depicted in Figure 1. The conceptual level of the TIJAH consists of three processing

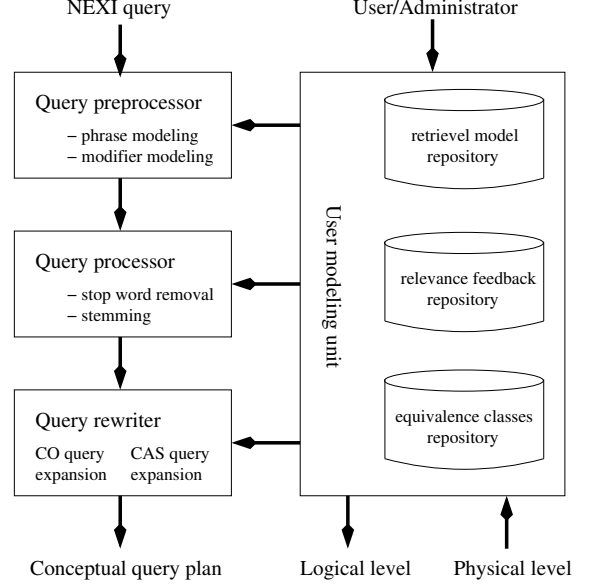


Figure 1: The conceptual level of the TIJAH system.

units, namely query preprocessor, query processor and query rewriter, and the user modeling unit encompassing three repositories: retrieval model repository, relevance feedback repository and equivalence classes repository². Query preprocessor and processor units are responsible for supporting the traditional IR system query processing (see e.g., Introduction in [1]), while the conceptual query rewriter extends the query according to the user specification and based on the content of repository files. Each conceptual unit is explained below.

The retrieval model repository and relevance feedback repository form the input to the preprocessing unit. How the data from the repository is interpreted depends on a specification in the user modeling unit. The retrieval model repository in the TIJAH system currently only stores the parameters of a retrieval model, i.e., smoothing parameter λ for the language model used for retrieval, and the desired size of the retrieved element for estimating the relevance on specificity.

3.1 Handling phrases and modifiers

The query preprocessing unit rewrites NEXI queries based on a user specification on handling phrases and modifiers. Thus, users can specify whether phrases should be considered as phrases or as a set of terms in the query, and whether

²The equivalence classes repository represents a repository that should support the heterogeneous collections retrieval and is still not fully supported in the TIJAH system.

```

CAS queries:
1. ROOT//article[ABOUT(./(abs|kwd), genetic algorithm "genetic algorithm")]//bdy//sec[ABOUT(.,simulated annealing)]
2. ROOT//article[ABOUT(./(abs|kwd), genet algorithm "genet algorithm")]//bdy//sec[ABOUT(.,simul anneal)]
3. ROOT//article[ABOUT(./(abs|kwd), genet algorithm "genet algorithm")
  AND ABOUT(., genet algorithm "genet algorithm")]//bdy//sec[ABOUT(., simul anneal)]
4. ROOT//article[ABOUT(./(abs|kwd), genet algorithm "genet algorithm")
  AND ABOUT(.,genet algorithm "genet algorithm") AND ABOUT(., simul anneal)]
  //bdy//sec[ABOUT(., simul anneal) AND ABOUT(., genet algorithm "genet algorithm")]

CO queries:
1. ROOT//article//*[ABOUT(., +tree +edit +distance +"tree edit distance" +XML -image)]PRIOR
2. ROOT//article[ABOUT(., +tree +edit +distance +"tree edit distance" +XML -image)]
  //*[ABOUT(., +tree +edit +distance +"tree edit distance" +XML -image)]PRIOR
3. ROOT//journal[MATCH(tp,0.34)]//*[MATCH(sec,0.22) or MATCH(p,0.18)]
  [ABOUT(., +tree +edit +distance +"tree edit distance" +XML -image)]PRIOR(856)

```

Figure 2: The conceptual query plans generated from the NEXI CAS and CO queries.

term and phrase modifiers ('+', and '-') should be considered during the query execution and in what way. Modifiers can be interpreted as strict containment conditions or as vague query conditions, stating that some terms or phrases are more important than the others for '+', and that fragments containing some terms or phrases should be "punished" for '-'. The result of the example CAS query processing where we used phrases can be seen in Figure 2 as CAS query 1.

3.2 Stop word removal and stemming

The standard IR query processing consisting of query stop word removal and stemming is done by processing unit. We used the standard Porter stemmer and a publicly available stop word list consisting of 429 stop words. Stemming and stop word removal are applied based on a user specification (in the user modeling unit). In Figure 2 CAS query 2 depicts the outcome of the processing unit in case stemming would have been performed.

3.3 Query expansion

The last step in conceptual query processing is query rewriting and expansion. The conceptual query rewriting unit distinguishes between NEXI content-only (CO) and content-and-structure (CAS) queries.

CO query expansion. NEXI CO queries are transformed into CAS queries according to the user specification. For instance, CO query can be translated in two ways:

- we are looking for any relevant XML element in any of the articles in the collection, including the articles themselves, as depicted in CO query 1 in Figure 2³, or
- we are looking for any relevant element in an article (including article elements themselves) that is about the topic specified, as depicted in CO query 2 in Figure 2.

The PRIOR in the conceptual query plan denotes that we use the relevance on specificity (i.e., result element size) in computing the final score of XML elements. The default is a length prior while for the log-normal prior the mean size should be specified (see CO example 3 in Figure 2).

³Although we used the same abbreviation as for the XPath `descendant::node` step, i.e. `//*`, in TIJAH system it is treated as `descendant_or_self:::node`.

CAS query expansion. Since NEXI CAS queries specify the element that should be retrieved as a result, query rewriting can only be about structural constraints in the *about* clause and about term distribution in different *about* clauses. Therefore, we applied to simple rules to enable elementary CAS query rewriting:

- relaxing the constraint that terms or phrases must be contained by the XML elements specified in the structural part of the *about* clause, as depicted in CAS query 3⁴ in Figure 2;
- further relaxing the structural constraints and allowing that terms or phrases in each subquery are also added to the other subquery (similarly as we had in the TIJAH 2003 approach [7]), as shown in Figure 2, CAS query 4.

4. PHRASE MODELING

For phrase modeling we follow the ideas introduced by Song and Croft [10], where the authors individualized unigram and bigram language models and combined them in an independent way. Thus, for a two-word phrase consisting of terms t_1 and t_2 the score value for a phrase can be computed as:

$$P(t_1, t_2|d) = \alpha_1 P_1(t_1|d) \text{ op } \alpha_2 P_2(t_1, t_2|d)$$

where α_1 and α_2 are parameters that define the relative influence of a unigram and bigram language model in the total score in case *op* is '+'. Operator *op* defines the nature of the combination of these models. We slightly modified the approach and used two interpretations:

- combination of n-gram LMs is modeled as an equally weighted sum: $P(t_1, \dots, t_{n-1}, t_n|e) = P_1(t_1|e) + P_2(t_1, t_2|e) + \dots + P_n(t_1, t_2, \dots, t_n|e)$, and
- combination of n-gram LMs is modeled as a product: $P(t_1, \dots, t_{n-1}, t_n|e) = P_1(t_1|e) \times P_2(t_1, t_2|e) \times \dots \times P_n(t_1, t_2, \dots, t_n|e)$.

In our approach the expression $P_i(t_1, t_2, \dots, t_i|e)$ gives the probability that the XML element e contains the phrase " $t_1 t_2 \dots t_i$ ".

⁴Note that the stemming has been applied in the query processing.

To be able to model phrases in region algebra we had to extend SRA to support phrase specification and more advanced score manipulation operators. For such a purpose we introduced two complex selection operators and two additional containment operators that are defined in Table 2.

The first selection operator makes a union of regions of the same type that have different name attribute. The operator is aimed at modeling search on multiple terms in the same region set. Therefore, it can be viewed as a shorthand notation for a logical expression consisting of a sequence of union operators applied on a selected region sets with the same type and different name attributes. The second operator makes a union of all adjacent sub-regions in the region set that have the same type attribute. The name attributes of these new regions take the names of the regions from the second operand, while their type is now changed to *adj*.

Following this approach and using *xp* as a shorthand for arbitrary SRA expression formed during the NEXI to SRA query translation, phrase "tree edit distance" can be transformed into one of the following logical expressions:

$$xp \sqsupset \otimes \sigma_{t=term, n='tree' \text{ adj } n='edit' \text{ adj } n='distance'}(R)$$

$$xp \sqsupset \oplus \sigma_{t=term, n='tree' \text{ adj } n='edit' \text{ adj } n='distance'}(R)$$

Here, *xp* denotes arbitrary SRA expression formed during the NEXI to SRA query translation.

The $\sqsupset \otimes$ and $\sqsupset \oplus$ operators define how relevance scores from regions with distinct region name attributes (i.e., regions with adjacent terms of different length) are combined to form the resulting score value for regions in the left operand ($r_1 \in R_1$). The exact definition of functions $f_{\sqsupset \otimes}$ and $f_{\sqsupset \oplus}$ is similar to the definition of function f_{\sqsupset} except that they treat adjacent regions with different name attributes in isolation and combine them based on the specification of operators \otimes and \oplus , i.e., multiplying or summing them, respectively.

Furthermore, the scaling operator (\otimes) is introduced in the SRA to model terms with and without modifier '+'. The operator definition is as follows: $R_1 \otimes num = \{r | r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge p := (p_1 * num)\}$. This operator scales down the terms (regions) that are not marked with '+' and, therefore, considered not so important, and scales up terms (regions) marked with '+' which are important terms. In our approach important terms are scaled with a *num* value that is double the *num* value for not so important terms. For example, in a query consisting of 6 terms of which 2 are marked as important (with '+'), important terms are scaled with *num* = 0.25 and not so important terms are scaled with *num* = 0.125.

Based on the defined operators we can now translate the conceptual query plan into the logical query plan. For example, the logical query plan for the CAS query 2 given in Figure 2 looks as given in Figure 3. In the Figure 3 we used capital (representing region names) instead of a selection operator on name and type attributes for brevity and to increase the readability of the query plan. For example, we use **ABS** instead of $\sigma_{t=entity, n='abs'}$ and **GENET_ALGORITHM** instead of $\sigma_{t=term, n='genet' \text{ adj } n='algorithm'}$.

The query plan subtree denoted with dashed lines in Figure 3 can also be expressed using a complex selection operator, and consequently this subexpression would look like:

$$\sigma_{t=entity, n='abs' \text{ or } n='kwd'}(C)$$

Although this query plan subtree, as well as subtrees bounded with dash-dot lines, repeat themselves in the logical query plan they are executed only once on the physical level.

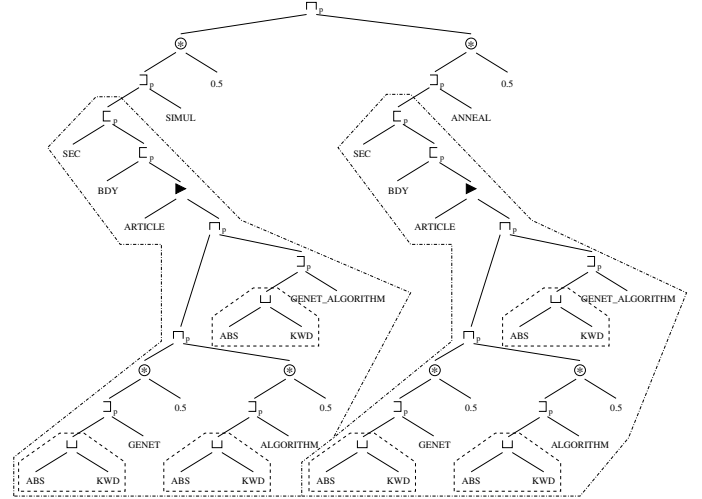


Figure 3: The SRA query plan for the topic 149.

For the phrase modeling on the physical level we implemented two functions as defined below:

$$f_{\sqsupset \otimes}(r, R) = \prod_{R_i} \eta t f(r, R_i) + (1 - \eta) c f(R_i)$$

$$f_{\sqsupset \oplus}(r, R) = \sum_{R_i} \eta t f(r, R_i) + (1 - \eta) c f(R_i)$$

Here, R_i is a set of regions (r_i) that have the same region name attribute (n_i). The sum and the product is defined over all sets R_i with different region names in R . Parameter η is used to specify the influence of foreground and background statistics for different adjacent regions in the region set.

5. RELEVANCE FEEDBACK

Our approach for the relevance feedback track is based on the idea that knowledge of relevant components provides implicit structural *hints* that may help improve the performance of the content-oriented queries. We use a two-step procedure to implement this idea:

- First, we extract the *structural relevance* of the top-ranked elements according to the relevance assessments;
- Second, the content-oriented query is rewritten into a structured one and the priors of the system are tuned based on the relevance feedback information. Then, the new structured query is evaluated in the TIJAH system.

In the following subsections, we define these two steps and explain their implementation in the TIJAH XML-IR system.

Table 2: Complex selection and containment operators.

Operators and operator definitions
$\sigma_{t=type, n=name_1 \text{ or } n=name_2 \text{ or } \dots \text{ or } n=name_n}(R) = \{r r \in R \wedge (t = type \wedge n = name_1) \vee (t = type \wedge n = name_2) \vee \dots \vee (t = type \wedge n = name_n)\}$
$\sigma_{t=type, n=name_1 \text{ adj } n=name_2 \text{ adj } \dots \text{ adj } n=name_n}(R) = \{r (r_1 \in R \wedge r_2 \in R \wedge t_2 = t_1 = type \wedge n_1 = name_1 \wedge n_2 = name_2 \wedge e_1 = s_2 - 1 \wedge (s, e, n, t, p) := (s_1, e_2, n_2, adj, p_2)) \vee \dots \vee (r_1 \in R \wedge r_2 \in R \wedge \dots \wedge r_n \in R \wedge t_n = \dots = t_2 = t_1 = type \wedge n_1 = name_1 \wedge n_2 = name_2 \wedge \dots \wedge n_n = name_n \wedge e_1 = s_2 - 1 \wedge e_2 = s_3 - 1 \wedge \dots \wedge e_{n-1} = s_n - 1 \wedge (s, e, n, t, p) := (s_1, e_n, n_n, adj, p_n))\}$
$R_1 \sqsupset_{\otimes} R_2 = \{r r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge p := p_1 \times f_{\sqsupset_{\otimes}}(r_1, R_2)\}$
$R_1 \sqsupset_{\oplus} R_2 = \{r r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge p := p_1 \times f_{\sqsupset_{\oplus}}(r_1, R_2)\}$

5.1 Extracting structural information

How to extract the *structural relevance* from the top relevant elements is a difficult problem. The semantics of the relevance assessments should be analyzed in depth to decide which kind of *structural hints* should be extracted from the different relevant components and to define what is the best interpretation for the different relevance combinations.

In our first attempt to model the *structural relevance* of a query, we use the *journal* and the *XML tag* name information from the top-ranked elements as well as their relevance values. We also use the size of these elements to update the length prior for the next iteration in the relevance feedback cycle. We believe that with a good combination of these *hints* we can considerably improve the performance of the content oriented results. In Section 6 we give a preliminary analysis of the results of this approach.

The remainder of this section details how the structural information from the top-ranked elements and the results from relevance assessments on exhaustivity and specificity are used to rewrite the query for the next iteration in the relevance feedback cycle.

5.1.1 Journal name

The content of the INEX collection consists of eighteen different journals. Each of these journals contains articles discussing a different computer science related field. We believe that when a component is assessed as relevant for a given topic, the journal it belongs to will contain elements with a similar content information. Therefore, we want to use this information to give a prior to the elements that are contained in that journal.

As an example, consider the relevance assessments for this year. If we consider only highly exhaustive and highly specific elements, marked with (3,3), we find that most of the topics have less than 3 relevant journals. That means, that likely, these are the journals that discuss the topic. Therefore, it is easy to imagine that other elements from these journals will also contain relevant information for that specific topic.

We decided to model the *journal prior* information according to the following formula:

$$P(J) = a + b \cdot \frac{\sum_{r \in top_{20} \sqsubseteq J} E_r}{3 \cdot |\{r \in top_{20} | E_r > 0\}|} + (1 - a - b) \cdot \frac{|J \sqsupseteq top_{20}|}{20},$$

where E_r is the exhaustivity value of the relevant top 20 components ($r \in top_{20}$) that belong (\sqsubseteq and \sqsupseteq , respectively)

to the journal J and a and b are weighting parameters used to tune the importance of this information.

Note that we only use the exhaustivity information to get a prior for the journal. We argue that if a component is somewhat exhaustive, it means that the journal is likely to be about, i.e., to contain the desired information specified in the query (whatever the specificity for that component is). We also reward the journals that have a higher number of elements in the top 20 ranked elements (see the third part of the sum in the equation).

5.1.2 XML tag names

The goal of using the element names for relevance feedback is to push up in the ranked result list the kind of elements we already know can be relevant for the topic ("make sense to be retrieved") and to push down the ones that not.

For modeling the information on the XML tag names extracted from the top-ranked elements (e), we use a similar approach as for the journals:

$$P(e) = a + b \cdot \frac{\sum_{r \in top_{20} \sqsubseteq e} E_r + S_r}{3 \cdot |\{r \in top_{20} | E_r \cdot S_r > 0\}|} + (1 - a - b) \cdot \frac{|e \in top_{20}|}{20}$$

In this case, we also take into account the *specificity* scale (S_r) as it gives information on the size of the element: i.e, if the element was large enough or too small for the information need.

5.1.3 Size

We use the size information to tune the length prior of our retrieval model for the next iteration. We believe that elements similar in length to those that are assessed as relevant have a higher likelihood to be the ones that the user is looking for. Nevertheless, it is not easy to combine the sizes of the top components to estimate a *desired* size to be retrieved.

We decided to use the following formula to define the *desired size* given the elements (r) in the top 20:

$$DesiredSize = \frac{\sum_{r \in top_{20}} size(r) \times SizeModifier_r}{\sum_{r \in top_{20}} sgn(SizeModifier_r)}$$

where $SizeModifier_r$ is defined as:

$$SizeModifier_r = \begin{cases} 1 & \text{if } (E_r, S_r) \in \{(2, 2), (3, 3)\} \\ 0 & \text{if } (E_r, S_r) \in \{(1, 1), (0, 0)\} \\ \frac{3 - E_r + S_r}{3} & \text{otherwise} \end{cases}$$

We based this formula on the assumption that very specific components that are not very exhaustive (i.e., $S_r > E_r$) are likely to be too small to answer the information need and, on the other hand, highly exhaustive components that are not very specific (i.e., $E_r > S_r$) are likely to be too large as an answer. In case there are no relevant elements in the 20 top-ranked elements, or the relevant elements are marginally exhaustive and marginally specific, i.e., marked with (1,1), we use the default value for desired size.

5.2 Rewriting and evaluating the query

After the information extraction on the structural information of the relevant elements and its fusion with the relevance assessments on exhaustivity and specificity, in the first step, the information is stored in the relevance feedback repository⁵. The information is used to rewrite the CO query and evaluate it in the TIJAH system. Assuming that in the relevance feedback repository for topic 166 the journal name specification is `tp` with a relevance prior 0.34, element names are `sec` and `p` with relevance prior values 0.22 and 0.18, respectively, and estimated element size is 856, the conceptual query after rewriting will look like the query given in Figure 2 as CO query plan 3. The `MATCH(e_name, imp)` is used to denote that the elements in the `e_name` have the higher probability to be relevant answers to a query than other elements, and the value `imp` gives their respective relevance prior.

The `MATCH` expression in the conceptual query plan is translated into a combination of a selection and scaling operators on the logical level. For example, query excerpt:

`xp[MATCH(sec,0.22) or MATCH(p,0.18)]`

is expressed on logical level as:

$xp \sqcap_p ((\sigma_{n='sec', t=entity}(C) \otimes 1.22) \sqcup (\sigma_{n='p', t=entity}(C) \otimes 1.18))$

where C is the collection of all regions (all XML elements in the collection). These operators are further translated into physical query plan as defined in previous sections.

6. EXPERIMENTS

In this section we give an overview of the experiments we did with the TIJAH XML-IR system. We present our preliminary results (official and additional runs) for the ad-hoc retrieval task (CO and CAS) and the relevance feedback task.

6.1 INEX ad-hoc track

As this year we used a completely new implementation on the physical level of our system, we design different experiments to evaluate which would be the best parameters for the retrieval model as well as to check if the scenarios used in previous years would produce the same performance on the new implementation.

6.1.1 CO queries

For the CO task we design two main experiments: The first one evaluates the effect of supporting phrases in the TIJAH XML-IR system as explained throughout the paper.

⁵Currently the computation and specification of these values are not completely integrated into the TIJAH system.

Table 3: CO experimentation runs: basic language model without length prior. Effects of supporting phrases. The 'n-gram' column indicates the kind of combination for the n-gram LMs and the 'R' column indicates the way the scores within a region are combined for a final score.

Run	λ	n-gram	R	avg. MAP	overlap
$R_{comp_{nophr}}$	0.35	-	-	0.0446	49.4%
$R_{comp_{phr1}}$	0.35	product	\otimes	0.0437	51.1%
$R_{comp_{nophr}}$	0.5	-	-	0.0496	52.3%
$R_{comp_{phr2}}$	0.5	product	\oplus	0.0502	82.8%
$R_{comp_{phr3}}$	0.5	weighted sum	\oplus	0.0470	82.1%

Table 4: Additional CO experimentation runs: Length priors

Run	length prior	Avg MAP	overlap
R_{comp05}	none	0.0496	52.3%
$R_{comp_{cut5}}$	$res > 5$	0.0534	51.8%
$R_{comp_{cut10}}$	$res > 10$	0.0578	53.2%
$R_{comp_{cut25}}$	$res > 25$	0.0635	55.4%
$R_{comp_{cut50}}$	$res > 50$	0.0645	57%
$R_{comp_{logn}}$	lognormal	0.0526	51.1%
$R_{comp_{logstd}}$	logstandard	0.0985	73.6%

Results of this runs are shown in Table 3. The different columns show the different approaches used to model the phrase search (see section 4 for details). According to the results, supporting phrase search improved the retrieval performance in only one of the runs. Note that this improvement is partially positive as the overlap increased considerably too.

The second experiment was designed to evaluate the effect of including a length prior in the retrieval model. We defined several priors and applied them to the best of our runs. The results are shown in Table 4. In the first four runs, the length prior consists on removing from the result list the elements smaller than a certain threshold. The last two runs use a lognormal and a logstandard distribution to model the length prior. We can see that, whatever the prior is, the performance of the original run improves. As we saw already in previous years, a logstandard distribution works best in our case, reaching a MAP of 1.4 in one of the metrics.

6.1.2 VCAS task

For the VCAS task, we designed three different scenarios: The first one, R_{strict} treats the structural constraints of a query strictly and all the result components must exactly match these structural constraints. The second and the third one, R_{relax} and R_{all} , implement the relaxations of the structural constraints explained in Section 3, queries 3 and 4 in Figure 2. The results of this runs are shown in Table 5. Even if the improvement for the first relaxation (second row) is not significant, we can see in the precision and recall graphs that the relaxation of the structural con-

Table 5: Official VCAS experimentation runs. Note that results in R_{all} have been modified due to an implementation error in the submitted ones

Run	Avg MAP	overlap
R_{strict}	0.0624	22.8%
R_{relax}	0.0694	24.3%
R_{all}	0.0588	23.9%

straints leads to better precision for this run. Contrary to last year, the second relaxation by using all the terms in all the about clauses did not performed as expected. Further analysis should determine if this is just an effect of the different topics for this year or if, in general, the relaxation is not appropriate for our purposes.

6.2 INEX relevance-feedback track

To see the performance of our approach using structural feedback, we designed different runs to try to identify which combination of the different structural and size information would work better. Unfortunately, at the moment of writing this paper, we do not know the exact way the results will be officially evaluated. Nevertheless, an analysis of the preliminary results shows that none of the combinations used improves significantly the performance of the retrieval system. Further experimentation will be required to see whether different values for the parameters of the formulas presented will give a better performance or some other interpretation of the relevance assessments should be done.

7. CONCLUSIONS AND FUTURE WORK

Our participation in INEX is characterized by applying a fully systematic approach able to support different retrieval tasks identified as ad-hoc task (CO and CAS) with simple user modeling and relevance feedback task. We investigated the influence of phrases in retrieval model with respect to the retrieval performance. Furthermore, we experimented with straightforward approaches to (blind) *structural* relevance feedback. Future research includes more extensive experimentation in the area of phrase search and relevance feedback, applying new models for incorporating different aspects of relevance feedback information, and taking more advanced methods for phrase search, by adapting IR approaches such as classifier-thing bigrams [6], by using the WWW as N-gram training data [13], by using vocabulary clustering [9], etc., to XML phrase modeling. Finally, we aim to improve the efficiency of the system on, both memory and CPU wise, using rewriting and optimization rules on the logical level as well as by applying horizontal fragmentation and encoding of data into more compact structures on physical level.

8. REFERENCES

- [1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.
- [2] P. Boncz. *Monet: a Next Generation Database Kernel for Query Intensive Applications*. PhD thesis, CWI, 2002.
- [3] T. Grust, S. Sakr, and J. Teubner. XQuery on SQL Hosts. In *Proceedings of the 30th Int'l Conference on Very Large Data Bases (VLDB)*, 2004.
- [4] T. Grust and M. van Keulen. Tree Awareness for Relational DBMS Kernels: Staircase Join. In H. M. Blanken, T. Grabs, H.-J. Schek, R. Schenkel, and G. Weikum, editors, *Intelligent Search on XML*, volume 2818 of *Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence (LNCS/LNAI)*, pages 179–192. Springer-Verlag, Berlin, New York, etc., August 2003.
- [5] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, Twente, The Netherlands, 2001.
- [6] M. Jiang, E. Jensen, S. Beitzel, and S. Argamon. Choosing the Right Bigrams for Information Retrieval. In *Proceeding of the Meeting of the International Federation of Classification Societies*, 2004.
- [7] 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.
- [8] V. Mihajlović, D. Hiemstra, H. E. Blok, and P. M. G. Apers. An XML-IR-DB Sandwich: Is it Better with an Algebra in Between? In *Proceedings of the SIGIR workshop on Information Retrieval and Databases (WIRD'04)*, pages 39–46, 2004.
- [9] R. Rosenfeld. Two Decades of Statistical Language Modeling: Where do we go from here? In *Proceedings of the IEEE*, 2000.
- [10] F. Song and W. B. Croft. A General Language Model for Information Retrieval. In *Proceedings of the eighth international Conference on Information and Knowledge Management*, pages 316–321, 1999.
- [11] 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.
- [12] M. van Keulen. Relational Approach to Logical Query Optimization of XPath. In *Proceedings of the 1st Twente Data Management Workshop (TDM'04) on XML Databases and Information Retrieval*, pages 52–58, 2004.
- [13] X. Zhu and R. Rosenfeld. Improving Trigram Language Modeling With the World Wide Web. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 533–536, 2001.

Flexible XML Retrieval Based on the Extended Vector Model

Carolyn J. Crouch
Department of Computer Science
University of Minnesota Duluth
Duluth, MN 55812
(218) 726-7607
ccrouch@d.umn.edu

Aniruddha Mahajan
Department of Computer Science
University of Minnesota Duluth
Duluth, MN 55812
(218) 726-7607

Archana Bellamkonda
Department of Computer Science
University of Minnesota Duluth
Duluth, MN 55812
(218) 726-7607

ABSTRACT

This paper describes the current state of our system for structured retrieval. It is based on an extension of the vector space model initially proposed by Fox [5]. The basic functions are performed using the Smart experimental retrieval system [11]. The major advance achieved this year is the inclusion of a flexible capability, which allows the system to retrieve at a desired level of granularity (i.e., at the element level). The quality of the resultant statistics is largely dependent on issues (in particular, ranking) which have yet to be resolved.

1. INTRODUCTION

Our original goal when we began our work with INEX in 2002 was to confirm the utility of Salton's vector space model [12] in its extended form for XML retrieval. Familiarity with Smart [11] and faith in its capabilities led us to believe that it might prove useful in this environment. Early results [2, 3] led us to believe that such a system could be utilized for XML retrieval if particular problems (e.g., flexible retrieval, ranking issues) could be solved. During 2002, much effort was spent on the translation of documents and topics from XML to internal Smart format and then back again into INEX reporting format. In 2003, we produced an operational system, but it did not include a flexible component (that is, it could only retrieve at the document level). During 2004, query formulation (for both CO and CAS queries) was completely automated. CAS queries received special attention to insure that all conditions of the query were met [1]. Our major improvement was the design and implementation of a flexible capability which allows the system to retrieve elements at various degrees of granularity [10]. Investigations with respect to relevance feedback in a structured environment were initiated.

We now have a system which, we believe, has the potential to function well in the XML environment. Significant issues, which stand to impact results markedly, remain open to investigation. These include, in particular, ranking and length normalization.

2. BACKGROUND

One of the basic models in information retrieval is the vector space model, wherein documents and queries are represented as weighted term vectors. The weight assigned

to a term is indicative of the contribution of that term to the meaning of the document. Very commonly, *tf-idf* weights [13] or some variation thereof [14] are used. The similarity between vectors (e.g., document and query) is represented by the mathematical similarity of the corresponding term vectors.

In 1983, Fox [5] proposed an extension of the vector space model—the so-called extended vector space model—to allow for the incorporation of objective identifiers with content identifiers in the representation of a document. An extended vector can include different classes of information about a document, such as author name, date of publication, etc., along with content terms. In this model, a document vector consists of a set of subvectors, where each subvector represents a different class of information. Our current representation of an XML document/query consists of 18 subvectors (i.e., *abs*, *ack*, *article_au_fnm*, *article_au_snm*, *atl*, *au_aff*, *bdy*, *bibl_atl*, *bibl_au_fnm*, *bibl_au_snm*, *bibl_ti*, *ed_aff*, *ed_intro*, *kwd*, *pub_yr*, *reviewer_name*, *ti*, *vt*) as defined in INEX guidelines. (These subvectors represent the properties of the document or article. Of the 18, eight are subjective, that is, contain content-bearing terms: *abs*, *ack*, *atl*, *bdy*, *bibl_atl*, *bibl_ti*, *ed_intro*, *kwd*.) Similarity between extended vectors in this case is calculated as a linear combination of the similarities of the corresponding subjective subvectors. (The objective subvectors serve here only as filters on the result set returned by CAS queries. That is, when a ranked set of elements is retrieved in response to a query, the objective subvectors are used as filters to guarantee that only elements meeting the specified criteria are returned to the user.)

Use of the extended vector model for document retrieval normally raises at least two issues: the construction of the extended search request [4, 6] and the selection of the coefficients for combining subvector similarities. For XML retrieval, of course, the query is posed in a form that is easily translated into an extended vector. The second problem—the weighting of the subvectors themselves—requires some experimentation. Experiments performed with the 2003 INEX topic set identified the following subjective subvectors as being particularly useful for retrieval: *abs*, *atl*, *bdy*, *bibl_atl*, *kwd*. We found our best results were obtained with subvector weights of 1, 1, 2, 2, and 1, respectively, and these same subvector weights were

applied to our 2004 topic set. (The 3 remaining subjective subvectors received 0 weights.) More investigation is required in this area with respect to the 2004 topics.

Another issue of interest here is the weighting of terms within subjective subvectors. Experiments indicated that the best results were achieved for the 2003 topics with the respect to both article and paragraph indexings when *Lnu.ltu* term weighting [15] was used. Our 2004 results are based on *Lnu.ltu* term weighting.

3. SYSTEM DESCRIPTION

Our system handles the processing of XML text as follows:

3.1 Parsing

The documents are parsed using a simple XML parser available on the web.

3.2 Translation to Extended Vector Format

The documents and queries are translated into Smart format and indexed by Smart as extended vectors. We selected the paragraph as our basic indexing unit in the early stages. Other indexing units were later added to include section titles, tables, figure captions, abstracts, and lists. (Thus, thinking in terms of documents, a document is represented in this system by a set of extended vectors—one representing its abstract, the rest representing its paragraphs, section titles, tables, figure captions, and lists.) *Lnu-ltu* term weighting is applied to all subjective subvectors.

3.3 Initial Retrieval

Retrieval takes place by running the queries against the indexed collection using Smart. The result is a list of *elements* ordered by decreasing similarity to the query. Consider all the elements in this list with a non-zero correlation with the query. Each such element represents a terminal node (e.g., paragraph) in the body of a document with some relationship to the query.

3.4 Flexible Retrieval

A basic requirement of INEX is that the retrieval method must return to the user components of documents or elements (i.e., abstract, paragraphs, subsections, sections, bodies, articles, figure titles, section titles, and introductory paragraphs) rather than just the document itself. The object is to return the most relevant element(s) in response to a query. Thus a good flexible system should return a mixture of document components (elements) to the user. These elements should be returned in rank order. The method to determine rank should incorporate both exhaustivity (relevance) and specificity (coverage).

Our flexible retrieval module (which we call Flex), is designed as follows. It takes as input a list of elements

(e.g., paragraphs), rank-ordered by similarity to the query as determined by Smart in the initial retrieval stage. These elements represent the leaves of the trees of articles in the document collection.

Consider Figure 1, which represents the tree structure of a typical XML article. The root of the tree is the article itself, whereas the leaf nodes are the paragraphs. Flexible retrieval should return relevant document components (e.g., <sec>, <ss1>, <ss2>, <p>, <bdy>, <article>, <ip1>, <abs>) as shown in Figure 1. In order to determine which components or elements of a tree to return, the system must first build the tree structure and then populate the tree by assigning a value to each non-terminal node in the tree. (The value in this case represents a function of exhaustivity and specificity.)

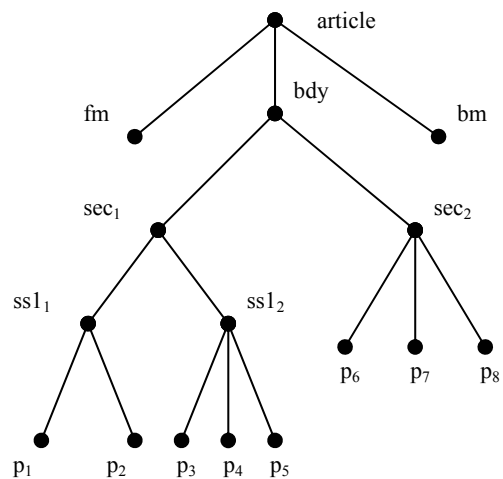


Figure 1. Tree Structure of a Typical XML Document

Flex takes a bottom-up approach. All leaf elements having non-zero correlations with the query have already been assigned similarity values by Smart. Consider the set of all such leaf elements which belong to the same tree. For a particular query, trees are constructed for all articles with leaves in this list. To construct the trees (and deal with the issue of specificity [coverage]), at each level of the tree, the number of siblings of a node must be known. The process is straight-forward. Suppose for example in Figure 1 that p_1 , p_2 , and p_7 were retrieved as leaf elements of the same tree. Flex would then build the tree represented in Figure 2. Any node on a path between a terminal node and the root node (article) is a valid candidate (element) for retrieval.

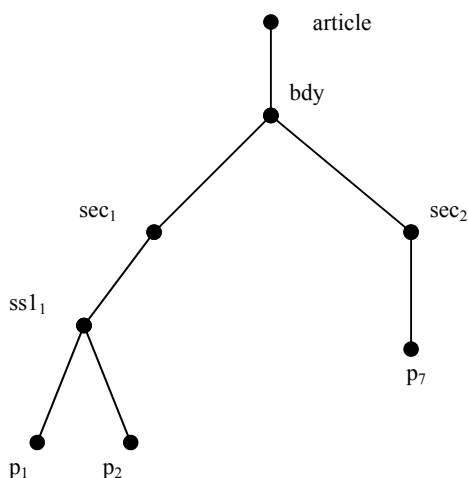


Figure 2. Tree of Relevant Elements

Building the tree is simple; populating it is not. We have weights (similarity values) associated with all terminal nodes. We consider each such value representative of that node's exhaustivity (e-value) with respect to the query. A question that arises at this point is how to produce a corresponding value representing specificity (s-value) for this node. Our current approach assigns an s-value equal to the e-value for that node. Since all the elements in question here (i.e., the terminal nodes) are relatively small in terms of the number of word types contained therein, this appears to be a reasonable initial approach.

Now that all the terminal nodes of the document tree have associated e-values and s-values, populating the tree (i.e., assigning e- and s-values to the rest of the nodes) begins. For every leaf node, we find its parent and determine the e- and s-values for that parent. The values of a parent are dependent on those of its children (i.e., relevance is propagated up the tree, whereas coverage may diminish as a function of the total number of children [relevant and non-relevant] of a node). The process continues until all the nodes on a path from a leaf to the root (the article node) have been assigned e- and s-values.

Our current methods for populating the tree (or propagating the e- and s-values upwards to the root) are primitive. A parent's e-value is set equal to the sum of the e-values of all its children, and its s-value is set equal to the average of the s-values of its children.

After all trees have been populated (and we have e- and s-values associated with each node of each tree), one major problem remains. How can we produce a rank-ordered list of elements (document components)

from these populated trees? We need a method which ranks the document components on the basis of e- and s-value. We considered a number of approaches. Our current method uses the product of e- and s-value to produce a single value for ranking. This is clearly inadequate. Ranking along with the propagation of e- and s-values will be the focus of much attention during the coming year.

Once a rank-ordered list is produced, the elements are reported for INEX evaluation. (The description given here presents the logical view of Flex; see [10] for a detailed view of Flex implementation.)

4. EXPERIMENTS

In the following sections, we describe the experiments performed with respect to the processing of the CO and CAS topics, respectively. In all cases, we use only the topic title as search words in query construction. As reported in Section 2, only five of the eight subjective subvectors (i.e., *abs*, *atl*, *bdy*, *bibl_atl*, and *kwd* with corresponding subvector weights 1, 1, 2, 2, and 1), were used for searching. Term weighting was *Lnu.ltu* in all cases. All indexing is done at the paragraph (basic indexing unit) level unless otherwise specified.

4.1 Using CO Topics

Our initial experiments using flexible retrieval were performed using the 2003 topic set. We used several simple methods for calculating e- and s-values and propagating these values up the tree. Table 1 presents the results of these experiments based on the P-strict and P-gen measures produced by *inex-eval*. The table shows the type of indexing (article or paragraph), type of retrieval (either conventional or flexible retrieval applied after Smart retrieval), the method by which e- and s-values are calculated and propagated, and the final ranking obtained.

Table 1. CO Processing (2003 Topic Set)

Indexing	Weight	Type	Desc	P-strict	P-gen
article	<i>Lnu-ltu</i>	Document	e	0.0648	0.0251
para	<i>Lnu-ltu</i>	flexible	add(e)	0.0717	0.0332
para	<i>Lnu-ltu</i>	flexible	add(e) * avg(s)	0.0847	0.0376

The first entry in the table gives the result when the indexing is performed at the article or document level (i.e., without flexible retrieval). Results are returned in rank order based on correlation with the query. The second entry represents flexible retrieval wherein only e-values were propagated up the tree, the e-value of a parent is the sum of the e-values of all its children, and the results are

sorted on e-value. Specificity was not utilized in this case. The third entry in the table uses both e- and s-value; e-value of a node is calculated as the sum of the e-values of its children whereas its s-value is the average of the s-values of its children. Final ranking of a node is based on the product of its e-value and s-value.

The results, as seen in this table, indicate that flexible retrieval produces an improvement over document retrieval for the 2003 topic set. The approach producing the best result (third line of Table 1) was subsequently applied to the 2004 topic set. The result is reported in Section 4.3.

4.2 Using CAS Topics

We process CAS topics in much the same fashion as CO topics, with some important exceptions. During pre-processing of the CAS queries, the subjective and objective portions of the query and the element to be returned (e.g., abstract, section, paragraph) are identified.

Depending on its syntax, a CAS query can be divided into parts, which can be divided into subparts depending on the number of search fields. Further subdivision, depending on the presence of plus or minus signs (representing terms that should or should not be present) preceding a search term, is also possible. CAS preprocessing splits the query into the required number of parts, each of which is processed as a separate Smart query. For example, suppose the query specifies that a search term is to be searched for in field 1. If the field to be searched is an objective subvector, the search term is distributed in that subvector. If the search field specifies a specific subjective subvector, the search term is distributed in that subvector, otherwise the search takes place in the paragraph subvector. The result in this last case is a set of elements (terminal nodes) returned by the Smart search which is used as input to Flex. Flex produces a ranked set of elements (terminal, intermediate, and/or root nodes) as the final output of this small search. After all subsearches associated with the query are complete, the final result set is produced (i.e., the original query is resolved). The element specified for return in the original query is then returned from each element in the result set. See [1] for more details.

Table 2 shows the results of three experiments involving the 2003 CAS topic set. The first entry in the table shows CAS retrieval based on an article indexing of the collection. The second entry represents the results of flexible retrieval on a paragraph indexing. The last entry in the table represents a hybrid approach, which uses conventional retrieval for queries which return articles and flexible retrieval for all other queries. This result shows a substantial increase in precision at the cost of requiring two separate indexings.

Table 2. CAS Processing (2003 Topic Set)

Indexing	Weight	Type	Desc	P-strict
article	<i>Lnu-ltu</i>	document	e	0.1658
para	<i>Lnu-ltu</i>	flexible	add(e) * avg(s)	0.1121
para	<i>Lnu-ltu</i>	hybrid	e or add(e) * avg(s)	0.2224

The method represented by the second entry in Table 2 (i.e., CAS processing using flexible retrieval based on a paragraph indexing) was subsequently applied using the 2004 CAS topic set. The result is reported below.

4.3 Results

During 2004, our work centered on two important aspects of the INEX ad hoc task: retrieving at the element level (i.e., flexible retrieval) and insuring that the result returned from a CAS search met the search requirements. The results, based on 2004 metrics, are shown in Table 3.

Table 3. CO and CAS Processing (2004 Topic Set)

Task	Average of RP measures	Generalized Recall
CO	0.05	34.94
CAS	0.04	27.99

We note that our results this year are not competitive, whereas last year we placed in the top group several times. In 2003, we were only able to retrieve documents. This year, with flexible retrieval based on a paragraph indexing, our current ranking scheme favors the return of smaller elements rather than larger ones. Yet as Kamps, *et.al.* [8] clearly show, in 2003 the probability of relevance increased with element length. Our method of propagating values and ranking of elements needs to take this into consideration.

4.4 Relevance Feedback in INEX

The importance and value of relevance feedback in conventional information retrieval have long been recognized. Incorporating relevance feedback techniques within the domain of structured retrieval is an interesting challenge.

Working within the constraints of our system, the first question which arises is how to translate the exhaustivity and specificity values associated with each INEX element into an appropriate measure of relevance in our system. Conventional retrieval views relevance assessment as binary. In INEX, we have a range of values for exhaustivity and a corresponding set of values for

specificity. There are many possibilities to consider when mapping these values to relevance.

As an initial approach, we decided simply to recognize as relevant those elements with e-values of 3. In other words, we recognize as relevant to the query all elements that are highly exhaustive (disregarding all other combinations). We were interested in determining the feasibility of this approach, which depends strongly on having enough of these elements available in the top ranks of retrieved elements. Our early experiments in this area are still being formulated; our results are too preliminary to report.

5. CONCLUSIONS

Our system, as a result of work done in the past year, is now returning results at the element level, i.e., retrieving at the desired level of granularity. The incorporation of a flexible retrieval facility is required before meaningful INEX experiments can take place. However, there are a number of problems still to be solved with this system, including in particular the propagation of e- and s-values upwards through the document hierarchy and the ranking of elements based on those values. Various approaches have been suggested [7, 9]. Much of our work in the coming year will focus on this issue. A good working result is important, since regardless of how well it does everything else, in the end all results depend on the system's ability to retrieve good elements. We believe that the initial retrieval through Smart produces valid terminal nodes with meaningful e-values. How well we build on this foundation will determine the final utility of the system.

A second area of interest is the extension of relevance feedback techniques to structured retrieval. There are many interesting questions to be addressed in this area in the coming year.

6. REFERENCES

- [1] Bellamkonda, A. Automation of Content-and-Structure query processing. Master's Thesis, Dept. of Computer Science, University of Minnesota Duluth (2004).
<http://www.d.umn.edu/cs/thesis/bellamkonda.pdf>
- [2] Crouch, C., Apte, S., and Bapat, H. Using the extended vector model for XML retrieval. In *Proc of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, (Schloss Dagstuhl, 2002), 99-104.
- [3] Crouch, C., Apte, S. and Bapat, H. An approach to structured retrieval based on the extended vector model. In *Proc of the Second Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, (Schloss Dagstuhl, 2003), 87-93.
- [4] Crouch, C., Crouch, D. and Nareddy, K. The automatic generation of extended queries. In *Proc. of the 13th Annual International ACM SIGIR Conference*, (Brussels, 1990), 369-383.
- [5] Fox, E. A. Extending the Boolean and vector space models of information retrieval with p-norm queries and multiple concept types. Ph.D. Dissertation, Department of Computer Science, Cornell University (1983).
- [6] Fox, E., Nunn, G. and Lee, W. Coefficients for combining concept classes in a collection. In *Proc. of the 11th Annual International ACM SIGIR Conference*, (Grenoble, 1988), 291-307.
- [7] Fuhr, N. , and GrossJohann, K. XIRQL: A query language for information retrieval in XML documents. In *Proc of the 24th Annual International ACM SIGIR Conference*, (New Orleans, 2001), 172-180.
- [8] Kamps, J., de Rijke, M., and Sigurbjornsson, B. Length normalization in XML retrieval. In *Proc of the 27th Annual International ACM SIGIR Conference* (Sheffield, England, 2004), 80-87.
- [9] Liu, S., Zou, Q., and Chu, W. Configurable indexing and ranking for XML information retrieval. In *Proc of the 27th Annual International ACM SIGIR Conference* (Sheffield, England, 2004), 88-95.
- [10] Mahajan, Aniruddha. Flexible retrieval in a structured environment. Master's Thesis, Dept. of Computer Science, University of Minnesota Duluth (2004).
<http://www.d.umn.edu/cs/thesis/mahajan.pdf>
- [11] Salton, G. *Automatic information organization and retrieval*. Addison-Wesley, Reading PA (1968).
- [12] Salton, G., Wong, A., and Yang, C. S. A vector space model for automatic indexing. *Comm. ACM* 18, 11 (1975), 613-620.
- [13] Salton, G. and Buckley, C. Term weighting approaches in automatic text retrieval. In *IP&M* 24, 5 (1988), 513-523.
- [14] Singhal, A. AT&T at TREC-6. In *The Sixth Text REtrieval Conf (TREC-6)*, NIST SP 500-240 (1998), 215-225.
- [15] Singhal, A., Buckley, C., and Mitra, M. Pivoted document length normalization. In *Proc. of the 19th Annual International ACM SIGIR Conference*, (Zurich, 1996), 21-19.

Relevance Feedback for XML Retrieval

Yosi Mass, Matan Mandelbrod
IBM Research Lab
Haifa 31905, Israel
+972-3-6401627
{yosimass, matan}@il.ibm.com

Abstract

Relevance Feedback (RF) techniques were studied in the context of traditional IR systems where the returned unit is an entire document. In this paper we describe an XML retrieval system that is capable of ranking XML components and we show how to apply existing RF algorithms on top of it to achieve Relevance Feedback for XML. We then demonstrate two example RF algorithms and show results of applying them on our XML retrieval system for the INEX'04 RF Track.

Keywords

XML Search, Information Retrieval, Vector space model, Relevance Feedback, Automatic Query Refinement.

1. Introduction

Relevance Feedback (RF) was studied e.g. in [6, 7] in the context of traditional IR engines that return full documents for a user query. The idea is to have an iterative process where results returned by the search engine are marked as relevants or not relevants by the user and this info is then fed back to the search engine to refine the query. RF algorithms can be also used for Automatic Query Refinement (AQR) by replacing the user with an automatic process that marks the highly ranked results returned by the search engine as relevants for use by subsequent iterations.

In [5] we described an algorithm for XML component ranking and showed how to apply existing AQR algorithms to run on top of it. Our Component ranking algorithm is based on detecting the most informative component types in the collection and creating separate indices for each such type. For example in the INEX[3] collection the most informative components are articles, sections and paragraphs. Given a query Q we run the query on each index separately and results from the different indices are merged into a single sorted result set with all component types.

We showed then in [5] that we can take advantage on this architecture of separate indices and apply existing AQR methods from traditional IR on each index separately without any modification to the AQR algorithm.

In this paper we show how to further exploit the separate indices architecture to achieve Relevance Feedback for XML by applying RF algorithms with real user feedback on

our base XML component ranking algorithm. The paper is organized as follows: In section 2 we describe a general method to apply existing RF algorithms on top of our component ranking algorithm. Then in section 3 we discuss two specific RF algorithms and demonstrate their usage on our XML retrieval using the method from section 2. In section 4 we report experiments with those algorithms for the INEX RF track and we conclude in section 5 with summary and future directions.

2. Relevance Feedback for XML Retrieval

In [5] we described an algorithm for XML component ranking and showed how to apply existing AQR algorithms on top of it. The base algorithm is described in Figure-1 below

1. For each index i
 - i. Compute the result set Res_i of running Q on index i
 - ii. Normalize scores in Res_i to $[0,1]$ by normalizing to $score(Q,Q)$
 - iii. Scale each score by its containing article score from Res_0
2. Merge all Res_i to a single result set Res composed of all components sorted by their score

Figure 1 - XML component ranking

We brief here the algorithm steps while full details can be found in [5]. In step 1 we run the given query on each index separately (step i) and then normalize result scores in each index to be able to compare scores from different indices (step ii). We then apply a DocPivot scaling (step iii) and finally in step 2 we merge the results to a single sorted result set of all components.

Since we have separate indices for different component granularities we showed that we can run the AQR algorithm on each index separately between steps i and ii in the above algorithm without modification of the original AQR algorithm.

Applying real user feedback is somewhat more complicated since the feedback is given on the merged result set and not on results from each index separately. A typical RF algorithm is an iterative process where top N results

returned by the search engine are marked as relevants or not relevants by the user and this info is then fed back to the search engine to refine the query. We still want to use existing RF algorithms on top of our component ranking algorithm without modifying the RF algorithms. We do this by applying the base component ranking algorithm as in Figure 1 above and then continue with the algorithm in Figure 2 below.

The algorithm in Figure 2 describes a single iteration of applying an RF algorithm on our component ranking algorithm. The algorithm works as follows; In step 3 we use the top N results from the merged results and base on the user feedback we select the subset of relevant (R) and non relevants (NR) components. Note that in a traditional RF algorithm the R and NR components are of the same type as the collection namely full documents while here they come from different component types so for each index some of them may be of different granularities than components in that index. We claim that the fact that a component is relevant or not relevant for the query can be used in a typical RF algorithm regardless to its granularity. In next section we demonstrate two specific RF algorithms and show that at least for them the above claim holds.

- | |
|---|
| <ol style="list-style-type: none"> 3. Take the top N results from Res and given their assessments extract R (Relevants) and the NR (Not relevants) from the top N. 4. For each index i <ol style="list-style-type: none"> i. Apply the RF algorithm on (R, NR, Res_i) with any other needed RF specific params and refine Q to Q' ii. Compute the result set Res'_i of running Q' on index i iii. Normalize scores in Res'_i to [0,1] by normalizing to score(Q',Q) iv. Scale each score by its containing article score from Res'₀ 5. Merge all Res'_i to a single result set Res' composed of all components sorted by their score 6. Freeze the original top N from Res as the top N in Res' |
|---|

Figure 2 - XML component ranking with RF

So in step 4 we just apply the existing RF algorithm on each of our indices separately where we give it the R and NR components as if they came from that index. The result is a refined query Q' (step i) and then similar to the AQR case the new query is used to generate a new result set Res_i for each index.

Results are then scaled by the DocPivot as described in [5] and finally the different result sets are merged (step 5) to a single result set of all component types.

To be able to measure the contribution of an RF iteration over the original query we take in step 6 the seen top N components and put them back as the top N in the final merged result. We then remove them from rest of Res' if they appear there again. In the next section we demonstrate two example RF algorithms that we applied on our XML component ranking using the algorithm from Figure 2 above.

3. Examples usage of RF algorithms for XML

3.1 Rocchio for XML

The Rocchio algorithm [6] is the first RF algorithm that was proposed for the Vector Space Model[8]. The idea in the Vector Space model is that both the documents and the query are represented as vectors in the space generated by all tokens in the collection (assuming any two tokens are independent). The similarity of a document to a query is then measured as some distance between the two vectors usually as the cosine of the angle between the two.

The Rocchio formula tries to find the optimal query; one that maximises the difference between the average of the relevant documents and the average of the non-relevant documents wrt the query. The Rocchio equation is given in Figure 3 below

$$Q' = \alpha Q + \beta / n_1 \sum_{i=1}^{n_1} R_i - \gamma / n_2 \sum_{i=1}^{n_2} NR_i$$

Figure 3 - The Rocchio equation

Q is the initial query, Q' is the resulted refined query, {R₁,...R_{n1}} are the set of relevant documents and {NR₁, ..., NR_{n2}} are the non-relevant documents. Since Q, {R_i}, and {NR_i} are all vectors then the above equation generates a new vector Q' that is close to the average of the relevant documents and far from the average of the non-relevant documents. The α , β , γ are tuning parameters that can be used to weight the effect of the original query and the effect of the relevant and the non-relevant documents. The Rocchio algorithm gets an additional parameter k which is the number of new query terms to add to the query.

Note that step 4.i in Figure 2 above is composed in the Rocchio case from two sub steps; In the first sub step new terms are added to the query and then the original query terms are reweighed. We can therefore apply for the Rocchio case two embedding variants into our XML component ranking -

1. Compute the new query terms only in the main index¹ and use them for other indices as well.
2. Compute a different set of new terms to add for each index separately.

¹ We always assume the the first index contains the full documents

In section 4 we report experiments we did with the Rocchio algorithm in our XML component ranking algorithm.

3.2 LA query refinement for XML

In [5] we described a general method to modify our component ranking algorithm with Automatic Query Refinement and described results for a specific such AQR algorithm based on [1]. The idea there is to add to the query Lexical Affinity (LA) terms that best separate the relevant from the non relevant documents with respect to a query Q . A Lexical Affinity is a pair of terms that appear close to each other in some relevant documents such that exactly one of the terms appears in the query.

This AQR algorithm can be used with real user feedback and can be plugged as a RF module in our component ranking algorithm as in Figure 2 above. In section 4 we describe experiments we did with that algorithm for XML retrieval.

4. The RF Track Runs description

4.1 Rocchio for XML

As discussed above the Rocchio algorithm is based on the Vector Space scoring model. Since our component ranking algorithm is also based on that model then we could easily plug the Rocchio algorithm into our component ranking as in Figure 2 above. In our implementation a document d and the query Q are represented as vectors as in Figure 4 below

$$d = (w_d(t_1), \dots, w_d(t_n)), w_d(t_i) = tf_d(t_i) * idf(t_i)$$

$$Q = (w_q(t_1), \dots, w_q(t_n)), w_q(t_i) = tf_q(t_i) * idf(t_i)$$

Figure 4 - Document and Query vectors

Where $tf_q(t)$ is a function of the number of occurrences of t in Q , $tf_d(t)$ is a function of the number of occurrences of t in d and $idf(t)$ is a function of the inverse document frequency of t in the index (exact functions details are described in [5]).

Given α, β, γ we define the Gain of a token t as

$$G(t) = \frac{\beta}{n_1} \sum_{i=1}^{n_1} w_{R_i}(t) - \frac{\gamma}{n_2} \sum_{i=1}^{n_2} w_{NR_i}(t)$$

Figure 5 - Gain of a token

where $w_{R_i}(t), w_{NR_i}(t)$ are the weights of t in each relevant component R_i , non-relevant component NR_i , respectively as defined in Figure 4. It is easy to see that tokens with the highest Gain are the ones that maximize the optimal query Q' as defined by the Rocchio equation in Figure 3 above.

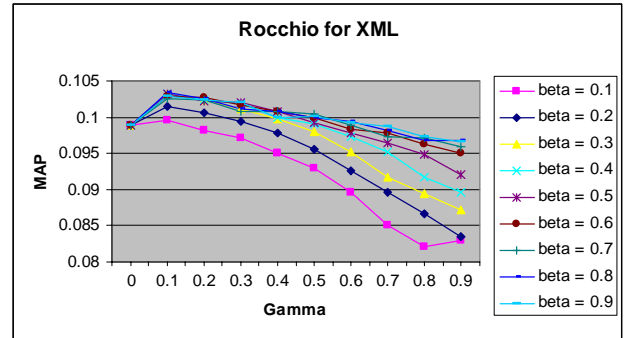


Figure 6 - Rocchio for XML

So in the RF step (4.i) in Figure 2 above we compute $G(t)$ for each new token t in the top N components that is not already in Q . We then select the k tokens with the maximal Gain as the terms to be added to the query Q .

We tried a single Rocchio iteration with $N = 20$, $\alpha = 1$, $\beta = \{0.1, \dots, 0.9\}$, $\gamma = \{0.1, \dots, 0.9\}$ and $k = 3$ on our base CO algorithm. The Mean Average Precision (MAP) values we got using the `inex_eval` aggregate metric for 100 results are summarized in Figure 6.

The Figure shows graphs for the different β values. The value for $\text{Gamma} = 0$ is the value of the base algorithm with no RF round. We can see that the value for $\gamma = 0.1$ gives best results for all β values. The best MAP was achieved at $\beta = 0.8$ which is what we sent for the RF track.

We see that we get very minor improvement (~5%) over the base run. A possible reason can be the Freezing method of the top N results which leave many possible Non-relevant components in the top N so the effect of RF is only for components at rank $N+1$ and lower.

4.2 LA refinement for XML

We tried several parameter combinations but at the time this paper is written we don't have consistent results so we don't report them yet.

5. Summary and Discussion

We have presented an XML retrieval system and showed how to run RF algorithms on top of it to achieve relevance feedback for XML. We then demonstrated two example RF algorithms and reported their usage for the XML RF track. We got relatively small improvements in the Mean Average Precision (MAP) with those algorithms and we still need to explore if it's an algorithm limitations or a possible problem in the metrics used to calculate the MAP.

6. References

- [1] D. Carmel, E. Farchi, Y. Petruschka, A. Soffer, Automatic Query Refinement using Lexical Affinities with Maximal Information Gain. Proceedings of the 25th Annual International ACM SIGIR Conference on

Research and Development in Information Retrieval, 2002.

- [2] D. Carmel, Y. Maarek, M. Mandelbrod, Y. Mass, A. Soffer, Searching XML Documents via XML Fragments, SIGIR 2003, Toronto, Canada, Aug. 2003
- [3] INEX, Initiative for the Evaluation of XML Retrieval, <http://inex.is.informatik.uni-duisburg.de>
- [4] 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
- [5] Y. Mass, M. Mandelbrod, Component Ranking and Automatic Query Refinement for XML retrieval, to appear in the Proceedings of the Third Workshop of the Initiative for The Evaluation of XML Retrieval (INEX), 6-8 December 2004, Schloss Dagstuhl, Germany
- [6] J. J. Rocchio, Relevance Feedback in information retrieval The SMART retrieval system – experiments in automatic document processing, (G. Salton ed.) Chapter 14 pg 313-323, 1971.
- [7] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems, Knowledge Engineering Review, 18(1):2003.
- [8] G. Salton, Automatic Text Processing – The Transformation, Analysis and Retrieval of Information by Computer, Addison Wesley Publishing Company, Reading, MA, 1989.

A Universal Model For XML Information Retrieval

Maria Izabel M. Azevedo
Departament of Computer Science
State University of Montes Claros
Montes Claros, Brazil
Izabel@dcc.ufmg.br

Lucas Pantuza Amorim
Departament of Computer Science
State University of Montes Claros
Montes Claros, Brazil
lucaspantuza@yahoo.com.br

Nivio Ziviani
Departament of Computer Science
Federal University of Minas Gerais
Belo Horizonte, Brazil
nivio@dcc.ufmg.br

ABSTRACT

In this paper we describe an adaptation of the vector space model for information retrieval on XML documents. This adaptation explores the semantic richness of XML markups and its nested structure. We compare results with the standard vector space model applied to the same collection and queries. We demonstrate how it can be applied to non XML documents, to homogeneous collections and to heterogeneous collections, to answer unstructured (CO - content only) and structured (CAS – content and structured) queries.

Keywords

XML Information Retrieval, INEX, Vector Space Model.

1. INTRODUCTION

Studying XML documents structure we can observe two special aspects on information organization: its hierarchical structure corresponding to the nesting of elements in a tree and the presence of markups that describes its content [1]. The first one is important for information retrieval because words on different levels may have different importance for expressing the information content in the whole tree. More over, if *markup describes its content*, it must have being defined semantically related to the information it delimits, making the second aspect especially important.

Another important aspect on XML documents is that it introduces a new division of information. We do not have only documents and collections anymore, now we have elements that can be inside of another element and also contain many others. Consequently, the unit of information to be returned to users can vary. If one element satisfies a query then its ancestor or descendent may also satisfy.

Besides, with XML documents, the user can propose queries that explore specific elements. There will be two types of queries, those with structural constraints, called CAS (Content and Structure), and those without constraints called CO (Content Only) [2].

In this paper we propose an adaptation of the vector space model

that considers both aspects (nested structure and markup that describes content) of XML structure in order to improve the vector space model performance, for CO e CAS queries, dealing with varying length units of information.

Although this model has being conceived to explore the semantic link of XML markup with its content, we demonstrate that it can be applied to non XML documents, preserving the vector space model performance and can also be applied to homogeneous collections, where homogeneous DTDs will not always allow appropriate semantic link between markups and information.

2. UNIT OF INFORMATION

The first challenge we face when studying XML Information Retrieval is what will be the ideal unit of information to be returned to the user, the one that best solve his information need. In one XML document, there are many possibilities: we can return a whole document or any of its sub-elements. But, what is the best one? It depends on the user query and each element content. So we need to evaluate each possibility in front of each query.

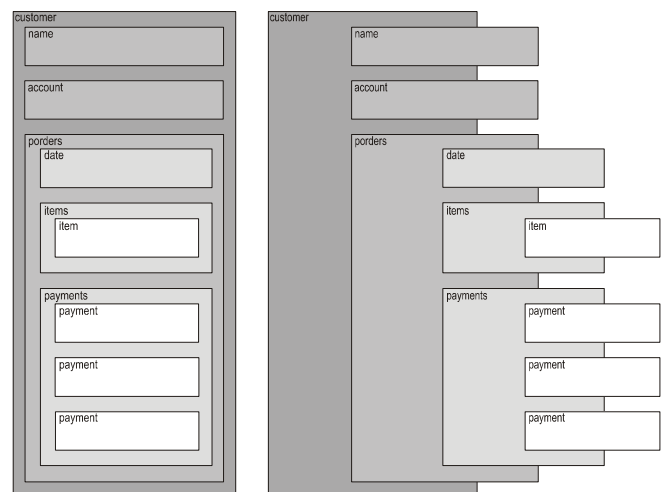


Figure 1 – Units of Information

To make it possible we should index all components as an information unit. In the example of Figure 1, our original collection will be expanded from 1 document and 10 sub-elements to 11 units of information, each one with an entry in the inverted list. Each element statistic will consider all text inside of its sub-elements.

3. STATISTICS MEASURES

From the vector space model, we have that the relevance of document D to query Q , $\rho(Q,D)$, is given by the cosine measure.

$$\rho(Q,D) = \sum_{ti \in Q \cap D} \frac{w_Q(ti) * w_D(ti)}{\|Q\| * \|D\|}$$

Formula 1

Where,

$$w_D(ti) = \log(tf(ti)) * \log(N * idf(ti))$$

Formula 2

To adapt this model to XML documents we introduce changes that express their characteristics, as follows.

First, we will consider each element as a new division of information and $tf(ti)$ and $idf(ti)$ are taking for each of them, becoming $tf(ti,e)$ and $idf(ti,e)$, so:

- $tf(ti,e)$ = number of occurrences of a term ti in a element e
- $idf(ti,e)$ = inverse of the number of elements that contain ti
- N = total number of elements in the collection

Now, we consider the nested structure of XML document. One term will be counted for elements where it appears in textual content and for the ancestor elements of e , as showed bellow in figure 2:

- $tf(january, \text{payment id}="P1") = 1$
- $tf(january, \text{payment id}="P2") = 1$
- $tf(january, \text{payments}) = 2$
- $idf(january) = 1/3$

```
<customer id="C1"/>
  <name> JOHN DOE </name>
  <account id="A1"> 1894654</account>
  <porders>
    <porder id="P01" acct="A1">
      <items>
        <item id="I1"> shoes </item>
      </items>
      <payments>
        <payment id="P1"> due january 15 </payment>
        <payment id="P2"> due january 20 </payment>
        <payment id="P3"> due february 15 </payment>
      </payments>
    </porders>
  </customer>
```

Figure 2 – One XML document

But it will imply that $\rho(january, \text{payments})$ will be greater than $\rho(january, \text{payment})$. This statistic will favor upwards information

units as their content include sub-elements content, and $tf(ti,e)$ will increase. This subject has been treated by Fuhr [3] using the concept of *augmentation*. We will apply this idea, using a factor (fnh) that will down weight terms contribution depending on its position on XML tree as explained later on.

4. THE MODEL

At this point, we have already 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 one new factor ($fxml$), which will explore XML characteristics, attributing different weights for each term contribution, depending on its occurrence on the tree that represents each document, resulting in Formula 3.

$$\rho(Q,D) = \sum_{ti \in Q \cap D} \frac{w_Q(ti) * w_D(ti,e) * fxml(ti,e)}{\|Q\| * \|D\|}$$

Formula 3

where

$$fxml(ti,e) = fnh(ti,e) * fstr(ti,e) * focr(ti,e)$$

Formula 4

The Nesting Factor, denoted by fnh , expresses the relevance of terms considering its position on the XML tree, and is given by:

$$fnh(ti,e) = 1/(1 + nl)$$

Formula 5

where

- nl = number of levels from element e to its sub-element containing term ti

The nesting factor can vary between the following two values:

- $fnh(ti,e) = 1$, for terms directly in element e , to
- $fnh(ti,e) = 1/nd$, nd being the depth of the XML tree

This factor will reduce the term contribution for distant elements (upwards) in XML tree.

The Structure Factor, denoted by $fstr$, expresses how the query structural constraints are satisfied by the context¹ of an element and is given by:

$$fstr(ti,e) = (common_markups + 1)/(nr_qmarkups + 1)$$

Formula 6

¹ Context is the complete path from the root to the element containing the textual content.

where,

- $common_markups$ = number of markups common in the query structural constraints and in the context of element e that contains ti .
- $nr_qmarkups$ = number of markups in the query structural constraints.

It can vary from:

- $fstr(ti,e) = 1/(nr_qmarkups+1)$, when no query's constraints appears in the context of ti , to
- $fstr(ti,e) = 1$, when all query's structural constraints markups appears in the context of ti .

This factor will valorize a context that better satisfies structural constraints present in the query. It is important on the CAS query, where users express elements that will better fit its information need. For CO queries it will be equal to 1, and will not influence the relevance equation.

The last factor, Co-occurrence Factor, denoted by **focr**, expresses the semantic link between markups and its content, and is given by:

$$focr(ti,e) = cf(ti,e) * idf(ti) * N * icf(e)$$

Formula 7

where,

- $cf(ti,e)$ = 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.
- $idf(ti,e)$ = inverse of the number of elements e that contain ti .

Then,

- $cf(ti,e) * idf(ti,e)$, is the reason between the number of times term ti appears with m for the numbers of elements contain ti in the collection;
- $icf(e)$ = inverse of the number of times markup m appears in the collection.
- N = total number of elements in the collection

Finally, $icf(e) * N$, express the popularity of markup m in the collection.

The co-occurrence factor valorizes co-occurrence of terms and markups, considering the popularity of markups.

With this factor, we intend to explore a XML characteristic, originated from its conception: the presence of *markups* that describes its *content* [1].

Concluding, XML factor (*fxml*) explores XML characteristics looking for the semantic of terms, looking for information behind words.

5. NON-XML DOCUMENTS

Considering that a real world collection may contain XML documents and non_XML documents, we will demonstrate that the same model can be applied on those documents, preserving the vector space model performance.

Examining Formula 3, we conclude that to satisfy this condition, XML factor must be equal to 1. Then:

$$fxml(ti,e) = fnh(ti,e) * fstr(ti,e) * focr(ti,e) = 1$$

Here we will consider that the whole content of a non-XML document will be delimited by one special markup, for example `<article>and </article>`.

Analyzing each of the *fxml* factors:

$$fnh(ti,e) = 1/(1+nl)$$

In one non-XML document exists only one level where all textual content is, so $nl = 0$ and $fnh(ti,e)$ will be equal to 1.

$$fstr(ti,e) = (common_markups + 1) / (nr_qmarkups + 1)$$

For a non-XML document, the numerator will be equal 1 because it has no markup. Denominator will depend on the query type:

- For CO $\rightarrow q_markups = 0$ and $fstru(ti,e) = 1$
- For CAS $\rightarrow nr_qmarkups$ may vary depending on the number of structural constraints in the query.

One non-XML document will never satisfy CAS query structural constraints because it is not structured, then its relevance will be decreased compared with those that can satisfy query constraints.

$$focr(ti,e) = cf(ti,e) * idf(ti) * N * icf(e)$$

For one non-XML:

- $cf(ti,e)$, the number of times ti appears with markup m , will be the number of times ti appears in the collections because all documents have the same special markup `<article>` and only this markup. So $cf(ti,e)$ is the inverse of $idf(ti)$, making $cf(ti,e) * idf(ti) = 1$.
- $icf(e)$, the inverse number of times markup m appears in the collection, will be equal $1/N$, the number of documents in the collection, because all documents in the collection will have the same special markup `<article>`. So $N * icf(e)$ will be equal 1, making $focr(ti,e) = 1$.

Non-XML documents are a special case and the model will converge to vector space model.

6. HOMOGENEOUS COLLECTIONS

An homogeneous collection is defined as a collection where all documents have the same DTD. In this section we will analyze the implications it has over our model.

Examining Formula 4 let us discuss each equation term.

$$Fstr(ti,e)$$

This factor will be analyzed only for CAS queries because for CO queries it will be always 1, as stated in Section 4.

As all documents have the same DTD, all of them will have the same elements and $fstr(ti,e)$ will not favor anyone. Any document will have the same probability to have an element return to user.

Within one document, those elements with more similarity with the query structural constraints will have greater relevance and will have better chance to be return to the user.

- ***Focr(ti,e)***

As will be the same DTD in all documents, no document will have its elements favor by this factor. But within a document, those elements which markups appear more times with a term will be favor, reinforcing the fact that *markups* describes its *content*.

7. HETEROGENEOUS COLLECTION

An heterogeneous collection is defined as a collection where documents may have different DTD. Our Model does not use any information that comes from DTD. It just indexes elements, terms and markups, collecting statistics that measure the relation between them, so we do not need to make any change for dealing with heterogeneous collections. But it is important to analyze how the heterogeneity of markups will influence the relevance ranking of our model.

- ***Fstr(ti,e)***

As stated before, this factor is always 1 for CO query. So let us analyze CAS queries. CAS queries impose a structural constraint, and will have greater relevance those elements that satisfy then. So, documents with DTDs similar to query's DTDs will be ranked first. Comes from this that if a user asks:

```
//article[about(../author, Jonh Smith)]
```

One document with an element:

```
<author> Jonh Smith </author>
```

Or even

```
<author>
  <first name> Jonh</first name>
  <last name> Smith</last name>
</author>
```

Will be better ranked then one contenting

```
<title> Jonh Smith Biography </title>
```

coming across information need of the user.

One document with a markup `<author>` will be better ranked then one with `<writer>` on its DTD, maybe not returning one important document for the information need of the user. But the user chooses the query constraints and it can make it more flexible using an OR operator including `<writer>`, for example.

- ***Focr(ti,e)***

This factor tries to explore the fact that *markups* describe its *content*. Considering that in an heterogeneous collection different DTD will allow better link between each document structure and its content, it will help to explore different meaning of the same words in different contexts.

But here appears the following language problem: which markup is semantically closer to Jonh Smith, `<author>` or `<writer>`? `<author>` OR `<autor>`?

Factor *focr* also ponders the frequency of markups in the collection by $N * icf(e)$. So if `<author>` is more wide spread than `<writer>` and `<autor>` it will have more chance to appear with Jonh Smith, but it will be compensated by *icf(e)*, in a similar way that common terms in one collection will be reduced by *idf(t)* in standard vector space model.

8. RESULTS

The proposed model was ran over the homogeneous and heterogeneous INEX collections. For the homogeneous collection, the effect of each term of *Fxml factor* was observed.

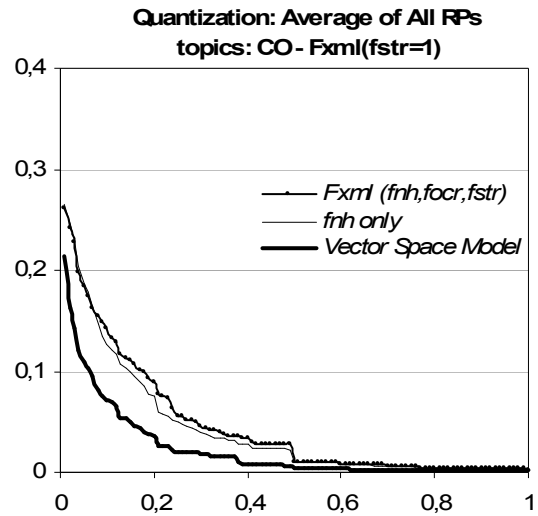


Figure 3 – Recall/Precision Curves comparing Vector Space Model and Adapted Model

The *Fnh* factor improves considerably the vector space model performance (Figure 3), as expected. Upwards elements accumulate all sub-elements contribution and without this consideration many of them would have been ranked first than more important sub-elements.

We also compared different values of *Fnh*, concluding that when it changes for elements in different level of XML tree precision improves a little bit.

Subsequently, we introduced the *Focr* factor and observed a small improvement (Figure 4), which can be attributed to the fact that in homogeneous collection this factor will not vary much, because all documents have the same structure.

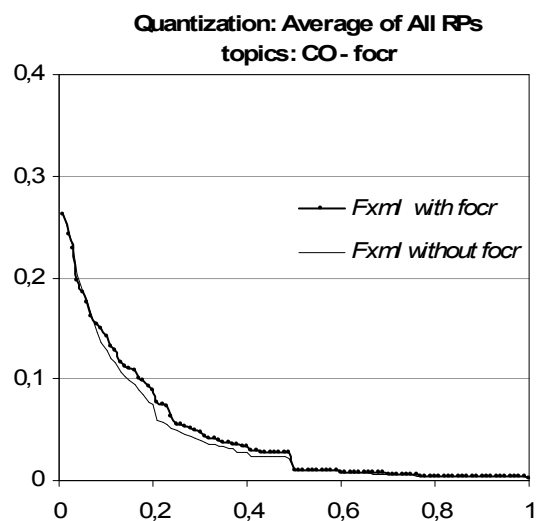


Figure 4 – Recall/Precision Curves changes with Focr

For CAS queries the factor *Fstr* was introduced and also caused some improvement (Figure 5).

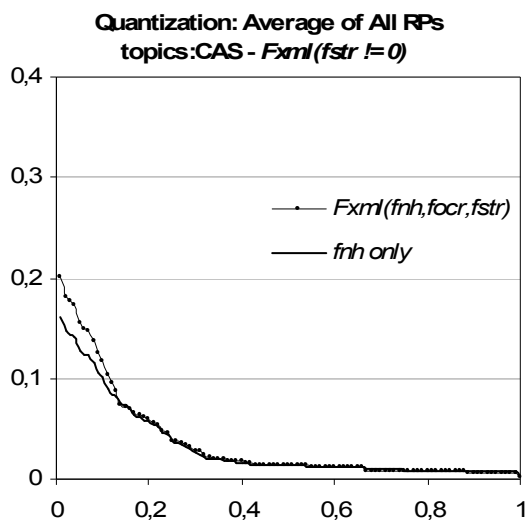


Figure 5 – Recall/Precision Curves changes with *Fstr*

We submitted runs to the heterogeneous collection, 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 the present model can deal with different DTDs.

For query:

```
//article[about(../author, Nivio Ziviani)]
```

We get the following answer:

```
<topic topic-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[23957]/author[1]/person[1]
</path>
<rank> 30</rank>
</result> ...
<result>
<subcollection name="hcibib" />
<file>hcibib</file>
<path>/file[1]/entry[22966]/article[1]/author[1]</path>
<rank> 139</rank>
</result>
```

9. CONCLUSION AND FUTURE WORK

We have shown a universal model for dealing with information retrieval on XML documents. It can be applied to non-XML documents, homogeneous and heterogeneous collections, to answer structured (CAS – content and structured) and no-structured (CO – content only) queries. The major contribution of this work is its universality, achieved in a completely automatic process.

Although all introduced factors behave as expected, the overall result was not good. The average precision stays around 0.05 and needs to be improved, demanding further investigation. *Fstr factor* should be better adjusted to query constraints and, for an appropriated assessment of *Focr factor* it would be better to have a real heterogeneous collection, with documents from different knowledge areas. The inverted index will increase to about three times the standard vector space model index demanding a specific research.

10. REFERENCES

- [1] S. Abiteboul, P. Buneman and D. Suciu. *Data on the Web – From Relations to Semistructured Data in XML*. Morgan Kaufmann Publishers, San Francisco, California, 2000.
- [2] G. Kazai, M. Lalmas and S. Malik. INEX'03 Guidelines for Topic Development. In *INEX 2003 Workshop Proceedings*, pages 153-154.
- [3] K. Grobjochnann, M. Abolhassani and N. Fuhr, Fröhlich. Content-oriented XML Retrieval with HyREX. In *INEX 2002 Workshop Proceedings*.
- [4] M. Mandelbrod and Y. Mass. Retrieving the most relevant XML Components. INEX'02. In *INEX 2003 Workshop Proceedings*.

Cheshire II at INEX '04: Fusion and Feedback for the Adhoc and Heterogeneous Tracks

Ray R. Larson
School of Information Management and Systems
University of California, Berkeley
Berkeley, California, USA, 94720-4600
ray@sherlock.berkeley.edu

ABSTRACT

This paper describes the retrieval approach used by UC Berkeley in the adhoc and heterogeneous tracks for the 2004 INEX evaluation. As in previous INEX evaluations, the main technique we are testing is the fusion of multiple probabilistic searches against different XML components using both Logistic Regression (LR) algorithms and a version of the Okapi BM-25 algorithm in conjunction with Boolean constraints for some elements. This year we also re-estimated the LR parameters for different components of the document collection 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.

Keywords

Information Retrieval, IR Evaluation, XML Retrieval

1. INTRODUCTION

Following the 2003 INEX evaluation[7] we continued to experiment with fusion approaches to XML retrieval, culminating in some quite respectable results that are reported in the forthcoming special INEX issue of the Journal of Information Retrieval[9]. The best performing of those approaches were used in this year's INEX adhoc task with no modification, as well as some additional experiments using blind feedback and applying Logistic regression alone using re-estimated parameters based on the relevance judgements from INEX 2003. In addition, element and collection fusion were used for the heterogeneous track (results are still being evaluated). In this paper we will describe the techniques used in these tracks for INEX 2004 and discuss their relative performance.

The basic approach taken in this and previous INEX evaluation is based on some early work done in TREC, where it was found that fusion of multiple retrieval algorithms provided an improvement over a single search algorithm[13, 2]. Later analyses of these fusion approaches[10, 1] indicated that the greatest effectiveness improvements appeared to occur between relatively ineffective individual methods, and the fusion of ineffective techniques, while often approaching the effectiveness of the best single IR algorithms, seldom exceeded them for individual queries and never exceeded their average performance. In our analysis of fusion approaches for XML retrieval[9], based on runs conducted after the 2003 INEX meeting, we conducted analyses of the overlap between re-

sult sets across algorithm and also examined the contributions of different XML document components to the results.

In this paper we will first discuss the algorithms and fusion operators used in our official INEX 2004 adhoc runs (as well as the not-quite-official runs submitted a bit late to the heterogeneous track). 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

In [9] we conducted an analysis of the overlap between the result lists retrieved by our Logistic Regression algorithm and the Okapi BM-25 algorithm. We found that on average, over half of the result lists retrieved by each algorithm in these overlap tests were both non-relevant *and* unique to that algorithm, fulfilling the main criteria for effective algorithm combination suggested by Lee[10]: that the algorithms have similar sets of relevant documents and different sets of non-relevant. This section is largely a repetition of the material presented in [9] with additional discussion of the re-estimation of the LR parameters for different XML components and indexes used in the INEX 2004 tests.

In the remainder of this section we describe the Logistic Regression and Okapi BM-25 algorithms that were used for the evaluation and we also discuss the methods used to combine the results of the different algorithms. The algorithms and combination methods are implemented as part of the Cheshire II XML/SGML search engine [7, 8, 6] which also supports a number of other algorithms for distributed search and operators for merging result lists from ranked or Boolean sub-queries. Finally, we will discuss the re-estimation of the LR parameters for a variety of XML components of the INEX test collection.

2.1 Logistic Regression Algorithm

The basic form and variables of the *Logistic Regression* (LR) algorithm used was originally developed by Cooper, et al. [4]. It provided good full-text retrieval performance in the TREC ad hoc task and in TREC interactive tasks [5] and for distributed IR [6]. As originally formulated, the LR model of probabilistic IR attempts to estimate the probability of rel-

evance 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 $P(R | Q, D)$ 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 $P(R | Q, D)$ uses the “log odds” of relevance given a set of S statistics, s_i , derived from the query and database, such that:

$$\log O(R | Q, D) = b_0 + \sum_{i=1}^S b_i s_i \quad (1)$$

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:

$$P(R | Q, D) = \frac{e^{\log O(R|Q,D)}}{1 + e^{\log O(R|Q,D)}} \quad (2)$$

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 $\langle article \rangle$) that contains front matter ($\langle fm \rangle$), a body ($\langle bdy \rangle$) and optional back matter ($\langle bm \rangle$). The front matter ($\langle fm \rangle$), in turn, can contain a header $\langle hdr \rangle$ and may include editor information ($\langle edinfo \rangle$), author information ($\langle au \rangle$), a title group ($\langle tig \rangle$), abstract ($\langle abs \rangle$) and other elements. A title group can contain elements including article title ($\langle atl \rangle$) the page range for the article ($\langle pn \rangle$), 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:

$$\begin{aligned} \log O(R | Q, C) = & b_0 + \left(b_1 \cdot \left(\frac{1}{|Q_c|} \sum_{j=1}^{|Q_c|} \log qtf_j \right) \right) \\ & + \left(b_2 \cdot \sqrt{|Q|} \right) \\ & + \left(b_3 \cdot \left(\frac{1}{|Q_c|} \sum_{j=1}^{|Q_c|} \log tf_j \right) \right) \quad (3) \\ & + \left(b_4 \cdot \sqrt{cl} \right) \\ & + \left(b_5 \cdot \left(\frac{1}{|Q_c|} \sum_{j=1}^{|Q_c|} \log \frac{N - nt_j}{nt_j} \right) \right) \\ & + \left(b_6 \cdot \log |Q_d| \right) \end{aligned}$$

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 j th term in a specific document component,

qtf_j is the frequency of the j th term in Q ,

nt_j is the number of components (of a given type) containing the j th 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 [3]. The b_i coefficients in the “Base” version of this algorithm were estimated using relevance judgements and statistics from the TREC/TIPSTER test collection. In INEX 2004 we used both this Base version and a version where the coefficients for each of the major document components were estimated separately and combined through component fusion. The coefficients for the Base version were $b_0 = -3.70$, $b_1 = 1.269$, $b_2 = -0.310$, $b_3 = 0.679$, $b_4 = -0.021$, $b_5 = 0.223$ and $b_6 = 4.01$. We will discuss the re-estimated coefficients for the various document components and indexes later in this section.

2.2 Okapi BM-25 Algorithm

The version of the Okapi BM-25 algorithm used in these experiments is based on the description of the algorithm in Robertson [11], and in TREC notebook proceedings [12]. As with the LR algorithm, we have adapted the Okapi BM-25 algorithm to deal with document components :

$$\sum_{j=1}^{|Q_c|} w^{(1)} \frac{(k_1 + 1)tf_j}{K + tf_j} \frac{(k_3 + 1)qt_j}{k_3 + qt_j} \quad (4)$$

Where (in addition to the variables already defined):

K is $k_1((1 - b) + b \cdot dl/avcl)$

k_1 , b and k_3 are parameters (1.5, 0.45 and 500, respectively, were used),

$avcl$ is the average component length measured in bytes

$w^{(1)}$ is the Robertson-Sparck Jones weight:

$$w^{(1)} = \log \frac{\left(\frac{r+0.5}{R-r+0.5} \right)}{\left(\frac{nt_j - r + 0.5}{N - nt_j - R - r + 0.5} \right)}$$

r is the number of relevant components of a given type that contain a given term,

R is the total number of relevant components of a given type for the query.

Our current implementation uses only the *a priori* version (i.e., without relevance information) of the Robertson-Sparck Jones weights, and therefore the $w^{(1)}$ value is effectively just an IDF weighting. The results of searches using our implementation of Okapi BM-25 and the LR algorithm seemed sufficiently different to offer the kind of conditions where data fusion has been shown to be most effective [10], and our overlap analysis of results for each algorithm (described in the evaluation and discussion section) has confirmed this difference and the fit to the conditions for effective fusion of results.

2.3 Boolean Operators

The system used supports searches combining probabilistic and (strict) Boolean elements, as well as operators to support various merging operations for both types of intermediate result sets. Although strict Boolean operators and probabilistic searches are implemented within a single process, using the same inverted file structures, they really function as two parallel *logical* search engines. Each logical search engine produces a set of retrieved documents. When a only one type of search strategy is used then the result is either a probabilistically ranked set or an unranked Boolean result set. When both are used within in a single query, combined probabilistic and Boolean search results are evaluated using the assumption that the Boolean retrieved set has an estimated $P(R | Q_{bool}, C) = 1.0$ for each document component in the set, and 0 for the rest of the collection.

The final estimate for the probability of relevance used for ranking the results of a search combining strict Boolean and probabilistic strategies is simply:

$$P(R | Q, C) = P(R | Q_{bool}, C)P(R | Q_{prob}, C)$$

where $P(R | Q_{prob}, C)$ is the probability of relevance estimate from the probabilistic part of the search, and $P(R | Q_{bool}, C)$ is the Boolean. In practice the combination of strict Boolean “AND” and the probabilistic approaches has the effect of restricting the results to those items that match the Boolean part, with ranking based on the probabilistic part. Boolean “NOT” provides a similar restriction of the probabilistic set by removing those document components that match the Boolean specification. When Boolean “OR” is used the probabilistic and Boolean results are merged (however, items that only occur in the Boolean result, and not both, are reweighted as in the “fuzzy” and merger operations described below.

A special case of Boolean operators in Cheshire II is that of proximity and phrase matching operations. In proximity and phrase matching the matching terms must also satisfy proximity constraints (both term order and adjacency in the case of phrases). Thus, proximity operations also result in Boolean intermediate result sets.

2.4 Result Combination Operators

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 [13] and used by Lee [10]. 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.

2.5 Recalculation of LR coefficients for component indexes

Using LR coefficients derived from relevance analysis of TREC data for INEX is unlikely to provide the most effective performance given the differences in tasks, queries and their structure, and relevance scales.

In order to begin to remedy this we have re-estimated the coefficients of the Logistic regression algorithm based on 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 index/component of the collection. These formulae were used in only one of the official *ad hoc* runs submitted for the INEX 2004 evaluation, in order to have a basis of comparison with the fusion methods used in INEX 2002 and 2003. In this section we focus on the re-estimation and the values obtained for the new coefficients. Later we will discuss the effectiveness of the new coefficients (or rather, the *lack* of effectiveness) and several possible reasons for it.

For re-estimation purposes we submitted the INEX 2003 CO queries using the “Base” LR algorithm, which was the best performing LR-only experiment as reported in [9] (which was able to obtain 0.0834 *mean average precision* under the strict quantization, and 0.0860 under the generalized quantization). In addition we performed separate runs using only searches on single indexes (which may combine multiple document elements, as described in Tables 2 and 4). For all of these runs we captured the values calculated for each of the variables described in equation 4 for each document element retrieved. Then the *strict* relevance/non-relevance of each of these documents was obtained from the INEX 2003 relevance judgements and the resulting relevance/element data was analyzed using the SPSS logistic regression procedure to obtain re-estimations of the variable coefficients (b_i) in equation 4. The resulting coefficients for the various components/indexes are shown in Table 1, where the “Base” row is the default TREC-estimated coefficients and the other rows are the estimates for the named index. Not all indexes were reestimated because they (e.g., *pauthor*) tend to be used as purely Boolean criteria, or were components of another index and/or not present in all articles (e.g., *kwd*).

Testing these new coefficients with the INEX 2003 queries and relevance judgements we were able to obtain a mean average precision of 0.1158 under the strict metric and 0.1116 for the generalized metric, thus exceeding the best fusion results reported in [9]. However, the data used for training the LR model was obtained using the relevance data associated with the same topics, and it appears very likely that the model was *over-trained* for that data, or that a different set of variables needs to be considered for XML retrieval.

Index	b_0	b_1	b_2	b_3	b_4	b_5	b_6
Base	-3.70	1.269	-0.310	0.679	-0.021	0.223	4.01
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 Logistic Regression Model

3. INEX 2004 ADHOC APPROACH

Our approach for the INEX 2004 adhoc task was quite similar to that used for INEX 2003 runs. This section will describe the indexing process and indexes used, and also discuss the scripts used for search processing. The basic database was unchanged from last year’s. We will summa-

rize the indexing process and the indexes used in the adhoc task for reference in the discussion.

3.1 Indexing the INEX Database

All indexing in the Cheshire II system is controlled by an 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 [?]).

Name	Description	Contents
docno	Digital Object ID	//doi
pauthor	Author Names	//fm/au/snm //fm/au/fnm
title	Article Title	//fm/tig/atl
topic	Content Words	//fm/tig/atl //abs //bdy //bibl/bb/atl //app
topicshort	Content Words 2	//fm/tig/atl //abs //kwd //st
date	Date of Publication	//hdr2/yr
journal	Journal Title	//hdr1/ti
kwd	Article Keywords	//kwd
abstract	Article Abstract	//abs
author_seq	Author Seq.	//fm/au @sequence
bib_author_fnm	Bib Author Forename	//bb/au/fnm
bib_author_snm	Bib Author Surname	//bb/au/snm
fig	Figure Contents	//fig
ack	Acknowledgements	//ack
alltitles	All Title Elements	//atl, //st
affil	Author Affiliations	//fm/aff
fno	IEEE Article ID	//fno

Table 2: Cheshire Article-Level Indexes for INEX

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 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.

Name	Description	Contents
COMP_SECTION	Sections	//sec
COMP_BIB	Bib Entries	//bib/bibl/bb
COMP_PARAS	Paragraphs	//ilrj //ip1 //ip2 //ip3 //ip4 //ip5 //item-none //p //p1 //p2 //p3 //tmath //tf
COMP_FIG	Figures	//fig
COMP_VITAE	Vitae	//vt

Table 3: Cheshire Components for INEX

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.

Component or Name	Description	Contents
COMP_SECTION		
sec_title	Section Title	//sec/st
sec_words	Section Words	//sec
COMP_BIB		
bib_author	Bib. Author	//au
bib_title	Bib. Title	//atl
bib_date	Bib. Date	//pdt/yr
COMP_PARAS		
para_words	Paragraph Words	*†
COMP_FIG		
fig_caption	Figure Caption	//fgc
COMP_VITAE		
vitae_words	Words from Vitae	//vt

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 2004 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 5 retrieval runs for the INEX 2004 adhoc task, three CO runs and 2 VCAS runs. This section describes the individual runs and general approach taken in creating the queries submitted against the INEX database and the scripts used to do the submission. The paragraphs below briefly describe Berkeley’s INEX 2004 runs.

Berkeley_CO_FUS_T_CMBZ (FUSION): This run uses automatic query generation with both Okapi BM-25 and Logistic regression retrieval algorithms combined using a score-normalized merging algorithm (MERGE_CMBZ). Results from multiple components were combined using MERGE_CMBZ as well. Separate retrieval of Articles, Sections and paragraphs were combined using score normalized merges of these results. Only Titles were used in generating the queries, which also included Boolean operations for proximity searching and ”negated” terms. This run was based on the most effective fusion method found in our post-INEX 2003 analysis and reported in [?] and was intended as a baseline for comparison with the other runs.

Berkeley_CO_FUS_T_CMBZ_FDBK (FEEDBACK): This run is fundamentally the same as the previous run, with the addition of ”blind feedback” where the `<kwd>` elements from top 100 results were retrieved, extracted and the top 30 most frequently occurring keyword phrases were used as an addition to the base query generated by the previous query.

Berkeley_CO_PROB_T_NEWPARMS (NEWPARMS): This run used automatic query generation with only the Logistic regression retrieval algorithm where the new coefficients for each of the indexes, as noted in Table 1, were used.

Berkeley_VCAS_FUS_T_CMBZ (FUSVCAS): This was a VCAS automatic run using only the topic title. This run uses automatic query generation from the NEXI expression in, and like the Berkeley_CO_FUS_T_CMBZ run, uses both Logistic Regression and the Okapi BM-25 ranking. Results from multiple components were combined using MERGE_CMBZ merging of results.

Berkeley_VCAS_PROB_T_NEWPARMS (NEWVCAS): This run also uses automatic query generation and is very similar to Berkeley_CO_PROB_T_NEWPARMS. Results from multiple components were also combined using MERGE_CMBZ merging of results. This run used only the LR algorithm with the new LR coefficients as shown in Table 1.

3.2.1 Query Generation and Contents

All of the Cheshire client programs are scriptable using Tcl or Python. For the INEX test runs we created scripts in the Tcl language that, in general, implemented the same basic sequence of operations as described in the INEX 2003 paper[?]. For VCAS-type queries, the NEXI specification was used to choose the indexes (and components) to be searched, and the RESTRICT operators described above were used to validate proper nesting of components. For each specified

“about” clause in the XPath, a merger of phase, keyword, Boolean and ranked retrieval was performed, depending on the specifications of the NEXI query.

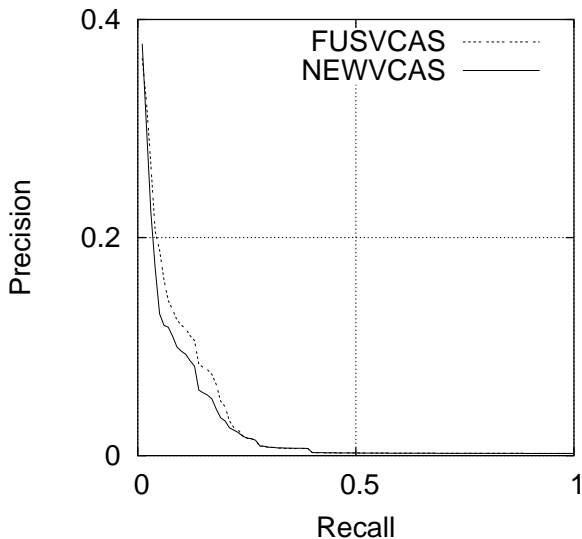


Figure 1: Berkeley VCAS Runs – Strict Quantization

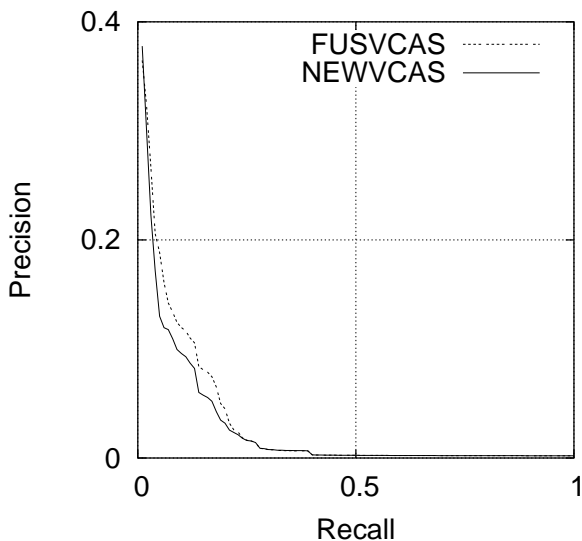


Figure 2: Berkeley VCAS Runs – Generalized Quantization

3.3 INEX '04 Heterogeneous Track Runs

The Heterogeneous Track for INEX 2004 is attempting to test the ability to perform searches across multiple XML collections with different structures and contents. Unfortunately we were late in submitting our runs (due to travel for another conference), so they are “unofficial” submissions. The results are still pending, and so they cannot be discussed here. In this section we briefly describe the approach taken for the track and the system features used in the implementation.

Our approach to the Heterogeneous Track was to treat the different collections as separate database with their own DTDs (simple “flat” DTDs were generated for those collections lacking them). The runs relied on Cheshire’s “Virtual Database” features, in which multiple physical databases can be treated as if they were a single database. In addition we used the search attribute mapping features of the Z39.50 protocol, so that each physical database configuration file could specify that some subset of tags was to be used for “author” searches, another for “title”, etc., for each as many of the index types described in Tables 2 and 4. Thus, when an “author” search was submitted to the virtual database, the query was forwarded to each of the physical databases, processed, and the results returned in a standardized XML “wrapper”. Thus we were able to run scripts similar to those used for the adhoc track “CO” runs against the virtual database requesting the LR algorithm and obtain a result from all of the physical database sorted by their estimated probability of relevance. In effect, the virtual search implements a form of distributed search using the Z39.50 protocol.

The only difficulty in this implementation was that all collections consisted of a single XML “document”, including one of the databases where that single document was 217Mb in size. We ended up treating each of the main sub-elements of these “collection documents” as separate documents (another feature of Cheshire). The difficulty was then generating the actual full XPath for the elements in order to report results. This was eventually handled by a script that, in most cases, was able to infer the element from the internal document ID, and in the case of the 217Mb document (with multiple different subelements for the collection document) this involved matching each of the subtypes in separate databases. Until the evaluation is complete, we won’t know whether this mapping was actually accurate.

4. EVALUATION

The summary average precision results for the runs described above are shown in Table 5. The table includes an additional row (...POST_FUS_NEWPARMS) for an unofficial run that essentially used the Berkeley_CO_FUS_T_CMBZ structure of combining LR and Okapi searching along with the new LR coefficients. This combination performed a bit better than any of the official runs.

Run Name	Short name	Avg Prec (strict)	Avg Prec (gen.)
...CO_FUS_T_CMBZ	FUSION	0.0923	0.0642
...CO_FUS_T_CMBZ_FDBK	FEEDBACK	0.0390	0.0415
...CO_PROB_T_NEWPARMS	NEWPARMS	0.0853	0.0582
...VCAS_T_CMBZ	FUSVCAS	0.0601	0.0321
...VCAS_PROB_T_NEWPARMS	NEWVCAS	0.0569	0.0270
...POST_FUS_NEWPARMS	POSTFUS	0.0952	0.0690

Table 5: Mean Average Precision for Berkeley INEX 2004

Figures 1 and 2 show, respectively, the Recall/Precision curves for strict and generalized quantization of each of the officially submitted Berkeley “VCAS” runs. Figures 3 and 4 show, respectively, the Recall/Precision curves for strict and generalized quantization of each of the officially submitted

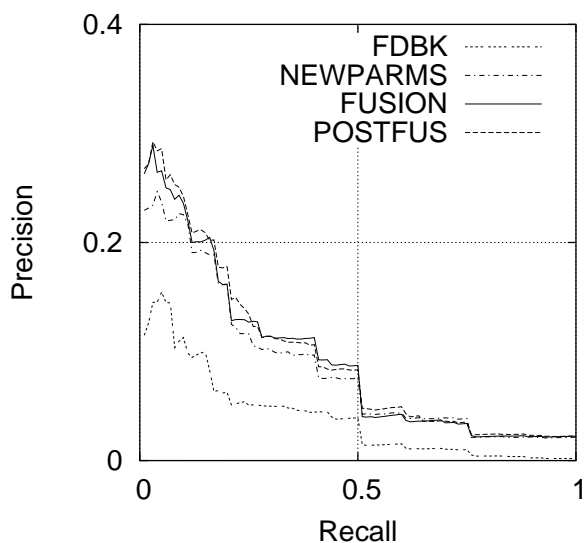


Figure 3: Berkeley CO Runs – Strict Quantization

Berkeley “CO” runs. No Berkeley runs appeared in the top ten for all submitted runs. None of these runs was in the top 10, though the “FUSION” run was close (ranked 14th in aggregate score).

Our attempt at “blind feedback” performed very poorly (which was expected, given that it was very much a last-minute attempt, and we had no time to attempt to determine the optimal number of records to analyze or the number of retrieved <kwd> phrases to include in the reformulated query). More interesting was the fact that the re-estimated LR parameters, when used alone did not perform as well as the basic fusion method. However, when combined with in a fusion approach the new coefficients do improve the results over the basic Fusion method using the “Base” coefficients.

5. CONCLUSIONS AND FUTURE DIRECTIONS

We still need to perform a number of analyses of alternative combinations, but it appears that the re-estimated LR coefficients, although not as effective as the submitted FUSION approach when LR alone is used, do provide additional improvement when combined in a similar fusion approach. Blind feedback using only assigned keywords in articles doesn’t appear to offer a benefit, though we plan to experiment a bit further using the framework developed for the “relevance feedback” track. Now with two years of usable and comparable relevance evaluations for INEX we can once again re-estimate the LR parameters from the INEX 2003 results and now test on the 2004.

6. REFERENCES

[1] S. M. Beitzel, E. C. Jensen, A. Chowdhury, O. Frieder, D. Grossman, and N. Goharian. Disproving the fusion hypothesis: An analysis of data fusion via effective information retrieval strategies. In *Proceedings of the 2003 SAC Conference*, pages 1–5, 2003.

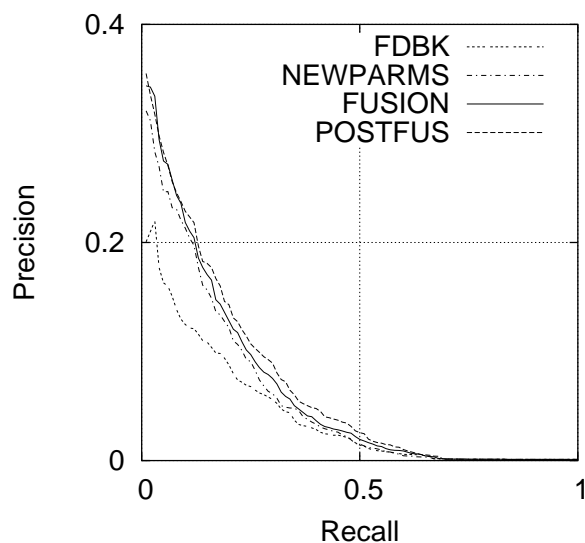


Figure 4: Berkeley CO Runs – Generalized Quantization

[2] N. Belkin, P. B. Kantor, E. A. Fox, and J. A. Shaw. Combining the evidence of multiple query representations for information retrieval. *Information Processing and Management*, 31(3):431–448, 1995.

[3] 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.

[4] 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.

[5] R. R. Larson. TREC interactive with cheshire II. *Information Processing and Management*, 37:485–505, 2001.

[6] 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.

[7] 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.

[8] R. R. Larson. Cheshire II at INEX 03: Component and algorithm fusion for XML retrieval. In *INEX 2003 Workshop Proceedings*, pages 38–45. University of Duisburg, 2004.

[9] R. R. Larson. A fusion approach to xml structured document retrieval. *Journal of Information Retrieval*, pages ??–??, 2005? (in press).

[10] 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.

- [11] S. E. Robertson and S. Walker. On relevance weights with little relevance information. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 16–24. ACM Press, 1997.
- [12] S. E. Robertson, S. Walker, and M. M. Hancock-Beauliee. OKAPI at TREC-7: ad hoc, filtering, vlc and interactive track. In *Text Retrieval Conference (TREC-7), Nov. 9-1 1998 (Notebook)*, pages 152–164, 1998.
- [13] 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.

Using a relevance propagation method for Adhoc and Heterogeneous tracks in INEX 2004

Karen Sauvagnat
IRIT-SIG
118 route de Narbonne
31 062 Toulouse Cedex 4
France
sauvagna@irit.fr

Mohand Boughanem
IRIT-SIG
118 route de Narbonne
31 062 Toulouse Cedex 4
France
bougha@irit.fr

ABSTRACT

This paper describes the evaluation of the XFIRM system in INEX 2004 framework. The XFIRM system uses a relevance propagation method to answer queries composed of content conditions and/or structure conditions. Runs were submitted to the ad-hoc (for both CO and VCAS task) and heterogeneous tracks.

Keywords

XML retrieval, ad-hoc task, heterogeneous task, relevance propagation method

1. INTRODUCTION

Long documents may have heterogeneous content from different domains. In that case, selecting a whole document as answer unit is not necessary useful for the user. He/she may require document parts, which are of higher precision and finer granularity. XML documents, combining structured and unstructured (i.e.) text data, allow the processing of information at another granularity level than the whole document.

The challenge in an IR context is to identify and retrieve relevant parts of the document. In other words, the aim is to retrieve the most exhaustive and specific information units answering a given query.

The approach we used for our participation in INEX 2004 is based on the XFIRM system and on a relevance propagation method. The idea of relevance propagation (or augmentation) is also undertaken in [2], [6], [5]. In our approach, all leaf nodes are used as a starting point of the propagation, because even the smallest leaf node can contain relevant information (it can be a *title* or *sub-title* node for example). Advantages of such an approach are twofold: 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.

In this paper, we present the methodology we use within the context of INEX'2004. Section 2 presents the XFIRM model, namely the data representation model and the associated query language. Section 3 describes the search approach used for the ad-hoc track for both CO and VCAS sub-tasks and section 4 presents the XFIRM model in the heterogeneous track context.

2. THE XFIRM MODEL

2.1 Data representation

2.1.1 Logical Data Representation Model

A structured document sd_i is a tree, composed of simple nodes n_j , leaf nodes ln_j and attributes a_j .

Structured document: $sd_i = (tree_i) = (n_{ij}, ln_{ij}, a_{ij})$

This representation is a simplification of Xpath and Xquery data model [4], in which a node can be a document, an element, text, a namespace, an instruction or a comment.

In order to easily browse the document tree and to quickly find ancestors-descendants relationships, the XFIRM model uses the following representation of nodes and attributes, based on the Xpath Accelerator approach [7]:

Node : $n_{ij} = (pre, post, parent, attribute)$

Leaf node : $ln_{ij} = (pre, post, parent, \{t_1, t_2, \dots, t_n\})$

Attribute: $a_{ij} = (pre, val)$

A node is defined thanks to its pre-order and post-order value (*pre* and *post*), the pre-order value of its parent node (*parent*), and depending on its type (simple node or leaf node) by a field indicating the presence or absence of attributes (*attribute*) or by the terms it contains ($\{t_1, t_2, \dots, t_n\}$).

A simple node can either contain other simple nodes, leaf nodes, or both. This last case is not really a problem, because as each node owns a *pre* and *post* order value independently of its type (simple node or leaf node), the reconstruction of the tree structure can be done in an easy way. An attribute is defined by the pre-order value of the node containing it (*pre*) and by its value (*val*). Pre-order and post-order values are assigned to nodes thanks respectively to a pre-fixed and post-fixed traversal of the document tree (see [12] for having an example).

If we represent nodes in a two-dimensions space based on the *pre* and *post* order coordinates, we can exploit the following properties, given a node n :

- all ancestors of n are to the upper left of n 's position in the plane
- all its descendants are to the lower right,
- all preceding nodes in document order are to the lower left, and
- the upper right partition of the plane comprises all following nodes (regarding document order)

Xpath Accelerator is well-suited for the navigation in XML documents with Xpath expressions. In contrast to others path index structures for XML, it efficiently supports also path expressions that do not start at the document root. Moreover, this data representation allow the processing of all XML documents, without necessarily knowing their DTD. This implies the ability of the model to process *heterogeneous* collections.

2.1.2 Physical Data Representation Model

As explained in our previous work [9], all data are stored in a relational database.

The *Path Index* (PI) allows the reconstruction of the document structure (thanks to the Xpath Accelerator model): for each node, its type, and its pre and post-order values are stored. The *Term Index* (TI) is a traditional inverted file, i.e. for each term, the index stores the nodes containing it and its positions in the different nodes. The *Element Index* (IE) describes the content of each leaf node, i.e. the total number of terms and also the number of different terms it contains, and the *Attribute Index* (AI) gives the values of attributes.

Finally, the *Dictionary* (DICT) provides for a given tag the tags that are considered as equivalent. It is useful in case of heterogeneous collections (i.e. XML documents with different DTD) or in case of documents containing similar tags, like for example, *title* and *sub-title*. This index is built manually.

2.2 The XFIRM Query language

A query language is associated to the model, allowing queries with simple keywords terms and/or with structural conditions [11].

The user can express hierarchical conditions on the document structure and choose the element he/she wants to be returned (thanks to the *te:* (target element) operator).

Examples of XFIRM queries:

- (i) // te: p [weather forecasting systems]
- (ii) // article[security] // te: sec ["facial recognition"]
- (iii) // te: article [Petri net] //sec [formal definition] AND sec [algorithm efficiency]
- (iv) // te: article [] // sec [search engines]

respectively mean that (i) the user wants paragraphs about weather forecasting systems, (ii) sections about facial recognition in articles about security, (iii) articles about Petri net containing a section giving a formal definition and another section talking about algorithm efficiency, and (iv) articles containing a section about search engines. When expressing the eventual content conditions, the user can use simple keywords terms, eventually preceded by + or - (which means that the term should or should not be in the results), and

connected with boolean operators. Phrases are also processed by the XFIRM system.

Concerning the structure, the query syntax allows the formulation of vague path expressions. For example, the user can ask for "article[//sec]" (he/she so knows that article nodes have sections nodes as descendants), without necessarily asking for a precise path, i.e. article[//bdy//sec]. Moreover, thanks to the Dictionary index, the user does not need to express in his/her query all the equivalent tags of the tag he/she's looking for. He/she can ask for example for a *section* node, without saying he/she is also interested in *sec* nodes. User can also express conditions on attribute values, as explained in [11].

3. AD-HOC TASK

This year, within the ad-hoc retrieval task, two sub-tasks were defined: the Content-Only (CO) task and the Vague Content-and-Structure (VCAS) task.

3.1 Answering CO queries

CO queries are requests that ignore the document structure and contain only content related conditions, e.g. only specify what a document/component should be about (without specifying what that component is). In this task, the retrieval system has to identify the most appropriate XML elements to return to the user.

3.1.1 Query processing

The first step in query processing is to evaluate the relevance value of leaf nodes ln according to the query. Let $q = t_1, \dots, t_n$ be this query. Relevance values are computed thanks to a similarity function called $RSV_m(q, ln)$ (Retrieval Status Value), where m is an IR model. The XFIRM system authorizes the implementation of many IR models, which will be used to assign a relevance value to leaf nodes. As shown in [10], a simple adaptation of the *tf-idf* measure to XML documents seems to perform better in case of content and structure queries. So:

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

with $w_i^q = tf_i^q * ief_i$ and $w_i^{ln} = tf_i^{ln} * ief_i$

And where :

- tf_i is the term frequency in the query q or in the leaf node ln

- ief_i is the inverse element frequency of term i , i.e. $\log(N/n+1)+1$, where n is the number of leaf nodes containing i and N is the total number of leaf nodes.

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 seems to be more significant for the root element than ones on deeper levels of the subtrees. It seems so intuitive 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 modeled in our propagation formula by the use of the $dist(n, ln_k)$ parameter, which 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 . The relevance value r_n of a node n is computed

according the following formula:

$$r_n = \sum_{k=1..N} \alpha^{dist(n,ln_k)} * RSV(q,ln_k) \quad (2)$$

where ln_k are leaf nodes being descendant of n and N the total number of leaf nodes being descendant of n .

To avoid the retrieval of nodes that do not supply information (like *title* nodes for example), we introduce the following rule: *Let n be a node and $ln_i, i \in [1..N]$ be its descendant leaf nodes having a non-zero relevance score. Let L be the sum of the length of ln_i (i.e. the sum of the number of terms contained in ln_i). If L is smaller than a given value x , n will be considered as not relevant.* This rule can be formalised as follows:

$$r_n = \begin{cases} \sum_{k=1..N} \alpha^{dist(n,ln_k)} * RSV(q,ln_k) & \text{if } L > x \\ 0 & \text{else} \end{cases} \quad (3)$$

$$\text{where } L = \sum_{i=1..N} \text{length}(ln_i) \text{ with } RSV(q,ln_i) > 0 \quad (4)$$

3.1.2 Experiments and results

Table 1 shows the results obtained with different values of α and L . All runs were performed using the title field of topics. Our official run is in bold characters.

Average precision decreases when factor L is considered (run *xfirm_co_05_25_o* processed with $L = 25$ obtains lower precision than runs processed with $L = 0$). This surprising observation can be mainly explained by the fact that some very small elements may have been judged relevant during the INEX assessments although they do not supply information.

Run *xfirm_co_05_25_wo* was processed without any node overlap. Results (for all metrics) are lower, because of the *overpopulated recall-base* [8].

Run *xfirm_co_1_0_o* is processed with α set to 1, which is equivalent to do a simple sum of leaf nodes relevance weights without down-weighting them during propagation. As a consequence, highest nodes in the document structure have a higher relevance value and are better ranked than deeper nodes (because they have a greater number of leaf nodes as descendants). As highest nodes in the document structure are also the biggest ones, the specificity criteria of the CO task is not respected. However, results are still relatively good, which is quite surprising. The same observation can be done on XFIRM results on INEX 2003 CO topics.

In a general manner, performances are lower than those obtained with the same parameters on the INEX 2003 CO topics, even for the "old" metrics. We need to conduct more experiments to evaluate the impact of all our parameters on each metric.

3.2 Answering VCAS queries

The VCAS (Vague Content and Structure) task consists in content-oriented XML retrieval based on content-and-structure (CAS) queries, where the structural constraints of a query can be treated as vague conditions. CAS queries are topic statements, which contain explicit references to the XML structure, and explicitly specify the contexts of the users interest (e.g. target elements) and/or the contexts of certain search concepts (e.g. containment conditions). The idea behind the VCAS sub-task is to allow the evaluation of

XML retrieval systems that aim to implement approaches, where not only the content conditions within a user query are treated with uncertainty but also the expressed structural conditions.

3.2.1 Query processing

A VCAS query evaluation in XFIRM is carried out as follows:

1. INEX (NEXI) queries are translated into XFIRM queries
2. XFIRM queries are decomposed into sub-queries and elementary sub-queries
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

Query translation

The transformation of INEX CAS queries to XFIRM queries was fairly easy. Table 2 gives some correspondences.

Query decomposition

Each XFIRM query can be decomposed into sub-queries SQ_i as follows:

$$Q = //SQ_1//SQ_2//...//te:SQ_j//...//SQ_n \quad (5)$$

Where *te:* indicates the tag name of the target element. Each sub-query SQ_i can then be re-decomposed into elementary sub-queries $ESQ_{i,j}$, eventually linked with boolean operators and of the form:

$$ESQ_{i,j} = tg[q] \quad (6)$$

Where *tg* is a tag name and $q = \{t_1, \dots, t_n\}$ is a set of keywords, i.e. a content condition. For example, topic 156 is decomposed as follows:

$$\begin{aligned} SQ_1 &= \text{article[] AND abs["spatial join"]} \\ ESQ_{1,1} &= \text{article[]} \\ ESQ_{1,2} &= \text{abs["spacial join"]} \\ SQ_2 &= \text{sec["performance evaluation"]} \end{aligned}$$

Evaluating leaf nodes relevance values

As for CO topics, formula 1 is used.

Elementary sub-queries $ESQ_{i,j}$ processing

The relevance values assigned to leaf nodes are then propagated upwards in the document tree until nodes having the

Table 1: Average precision of our runs for CO topics

Run	α	L	Average precision	Overlap	Rank
xfirm_co_05_25_o	0.5	25	0.0660	77,4%	19/70
xfirm_co_06_0_o	0.6	0	0.0758	81,8%	17/70
xfirm_co_09_0_o	0.9	0	0.0754	83,8%	17/70
xfirm_co_1_0_o	1	0	0.0781	83,8%	
xfirm_co_05_25_wo	0.5	25	0.0143	0%	48/70

Table 2: Transformation of INEX topics into XFIRM queries

Topic	INEX	XFIRM
138	//article [about(.,operating system) and about(./sec,thread implementation)]	// te: article [operating system] // sec [thread implementation]
145	//article[about(.,information retrieval) p[about(.,relevance feedback)]]	//article [information retrieval] // te: p [relevance feedback]
156	//article[about(./abs,"spatial join")]// bdy // sec [about(., "performance evaluation")]	//article[] AND abs["spatial join"] // te: sec ["performance evaluation"]

asked tag name are found. The result set of an elementary sub-query $tg[q]$ is so composed of nodes having tg as tag name (or having a tag name equivalent to tg according to the DICT index) and their associated relevance values, which are obtained thanks to the propagation.

Formally, the result set $R_{i,j}$ of $ESQ_{i,j}$ is a set of pairs (*node*, *relevance*) defined as follows:

$$R_{i,j} = \{(n, r_n) / n \in \text{construct}(tg) \text{ and } r_n = F_k(RSV_m(q, nf_k), \text{dist}(n, nf_k))\} \quad (7)$$

Where :

- r_n is the relevance weight of node n
- the $\text{construct}(tg)$ function allows the creation of the set of all nodes having tg as tag name
- the $F_k(RSV_m(q, nf_k), \text{dist}(n, nf_k))$ function allows the propagation and aggregation of relevance values of leaf nodes nf_k , descendants of node n , in order to form the relevance value of node n . This propagation is function of distance $\text{dist}(n, nf_k)$ which separates node n from leaf node nf_k in the document tree (i.e. the number of arcs that are necessary to join n and nf_k).

In our INEX 2004 experiments, we choose to use the following function :

$$F_k(RSV_m(q, nf_k), \text{dist}(n, nf_k)) = \sum_k \alpha^{\text{dist}(n, nf_k)} * RSV(q, nf_k) \quad (8)$$

α is set to 0.9, which is the optimal value for experiments presented in [12] on INEX 2003 SCAS topics.

Sub-queries SQ_i processing

Once each $ESQ_{i,j}$ has been processed, sub-queries SQ_i are then evaluated as explained below. Let R_i be the result set of SQ_i .

- if sub-query SQ_i is composed of one elementary sub-query $ESQ_{i,j}$ then the result set of SQ_i is the same than the one of $ESQ_{i,j}$

$$\text{If } SQ_i = ESQ_{i,j}, \text{ then } R_i = R_{i,j} \quad (9)$$

- if sub-query SQ_i is composed of elementary sub-queries $ESQ_{i,j}$ linked by the Boolean operator AND, the result set

of SQ_i is composed of nodes being the nearest common ancestors of nodes belonging to the result sets of elementary sub-queries $ESQ_{i,j}$. The associated relevance values are obtained thanks to propagation functions. Formally,

$$\text{If } SQ_i = ESQ_{i,j} \text{ AND } ESQ_{i,k}, \text{ then } R_i = R_{i,j} \oplus_{AND} R_{i,k} \quad (10)$$

with \oplus_{AND} defined as follow:

DEFINITION 1. Let $N = \{(n, r_n)\}$ and $M = \{(m, r_m)\}$ be two sets of pairs (*node*, *relevance*).

$$N \oplus_{AND} M = \{(l, r_l) / l \text{ is the nearest common ancestor of } m \text{ and } n, \text{ or } l = m \text{ (respectively } n) \text{ if } m \text{ (resp. } n) \text{ is ancestor of } n \text{ (resp. } m), \forall m, n \text{ being in the same document and } r_l = \text{aggreg}_{AND}(r_n, r_m, \text{dist}(l, n), \text{dist}(l, m))\} \quad (11)$$

Where $\text{aggreg}_{AND}(r_n, r_m, \text{dist}(l, n), \text{dist}(l, m)) = r_l$ defines the way relevance values r_n and r_m of nodes n and m are aggregated in order to form a new relevance r_l .

According to the results obtained in [12] for the INEX'2003 SCAS topics, we use the following function in this year experiments:

$$\text{aggreg}_{AND}(r_n, r_m, \text{dist}(l, n), \text{dist}(l, m)) = \frac{r_n}{\text{dist}(l, n)} + \frac{r_m}{\text{dist}(l, m)} \quad (12)$$

- if sub-query SQ_i is composed of elementary sub-queries $ESQ_{i,j}$ linked by the Boolean operator OR, the result set of SQ_i is an union of the result sets of elementary sub-queries $ESQ_{i,j}$.

$$\text{If } SQ_i = ESQ_{i,j} \text{ OR } ESQ_{i,k}, \text{ then } R_i = R_{i,j} \oplus_{OR} R_{i,k} \quad (13)$$

with \oplus_{OR} defined as follow:

Table 3: Average precision of our runs for VCAS topics

Run	Dict	Average precision	Overlap	Rank
xfirm_vcas_09_vague	INEX	0.0346	17,8%	26/51
xfirm_vcas_09_vague_dict	DICT1	0.0475	38,5%	16/51
xfirm_vcas_09_vague_dict2	DICT2	0.0686	62,6%	8/51
xfirm_vcas_09_vague_dict3	DICT3	0.0694	68,3%	7/51

DEFINITION 2. Let $N = \{(n, r_n)\}$ and $M = \{(m, r_m)\}$ be two sets of pairs (node, relevance).

$$N \oplus_{OR} M = \{(l, r_l) / l = n \in N \text{ and } r_l = r_n \\ \text{ or } l = m \in M \text{ and } r_l = r_m\} \quad (14)$$

Whole query processing

The result set of sub-queries SQ_i are then used to process the whole query. In this query, a target element is specified, as defined above.

$$Q = //SQ_1//SQ_2//\dots//te : SQ_j//\dots//SQ_n$$

The aim in whole query processing will be to propagate the relevance values of nodes belonging to the results sets R_i of sub-queries SQ_i to nodes belonging to the result set R_j , which contains the target elements. Relevance values of nodes belonging to R_i where $i \in [1 \dots j - 1]$ are propagated downwards in the document tree, while relevance values of nodes belonging to R_i where $i \in [j + 1 \dots n]$ are propagated upwards. This is obtained thanks to the non-commutative operators Δ and ∇ defined below:

DEFINITION 3. Let $R_i = \{(n, r_n)\}$ and $R_{i+1} = \{(m, r_m)\}$ be two sets of pairs (node, relevance)

$$R_i \Delta R_{i+1} = \{(n, r_n) / n \in R_i \text{ is descendant of } m \in R_{i+1} \\ \text{ and } r_n = prop_agg(r_n, r_m, dist(m, n))\} \quad (15)$$

$$R_i \nabla R_{i+1} = \{(n, r_n) / n \in R_i \text{ is ancestor of } m \in R_{i+1} \\ \text{ and } r_n = prop_agg(r_n, r_m, dist(m, n))\} \quad (16)$$

Where $prop_agg(r_n, r_m, dist(m, n)) \rightarrow r_n$ allows the aggregation of relevance weights r_m of node m and r_n of node n according to the distance that separates the 2 nodes, in order to obtain the new relevance weight r_n of node n .

The result set R of a query Q is thus defined as follows :

$$R = R_j \nabla (R_{j+1} \nabla (R_{j+2} \nabla \dots)) \\ R = R_j \Delta (R_{j-1} \Delta (R_{j-2} \Delta \dots)) \quad (17)$$

For the experiments presented here and according to the results obtained in [12], we use:

$$prop_agg(dist(m, n), r_n, r_m) = \frac{r_n + r_m}{dist(n, m)} \quad (18)$$

3.2.2 Experiments and results

We processed several runs, using different Dictionary indexes. This way, we are able to control the notion of *uncertainty* on structure constraints.

Table 3 shows the average precisions of all runs. All runs

were performed using the title field of topics. Our official runs are in bold characters.

Equivalencies in the *INEX Dict* are given in the INEX guidelines [1]. For example, *ss1*, *ss2*, and *ss3* nodes are considered as equivalent to *sec* nodes. Equivalencies are then gradually extended in the other DICT indexes. For example, *sec*, *ss1*, *ss2*, *ss3* nodes and *p* nodes are considered as equivalent in DICT1, whereas in DICT3 *sec*, *ss1*, *ss2*, and *ss3* nodes are equivalent to both *p* and *bdy* nodes. The use of a very extended DICT index increases the percentage of nodes overlap but increases also the average precision. This is not really surprising because as the structure conditions are treated with uncertainty, the recall-base obtained during the assessments is overpopulated, as it is the case for the CO task.

4. HETEROGENEOUS TRACK

4.1 Motivation

The idea behind the heterogeneous track is that an information seeker is interested in semantically meaningful answers irrespectively to the structure of documents.

The Het collection consists in documents from different sources and thus with different DTDs (one can cite the original INEX Collection, the Computer Science database of FIZ Karlsruhe or the Digital Bibliography and Library Project in Trier). The additional collections are mainly composed of bibliographic references. Documents' sizes are also very "heterogeneous" : smallest documents have a few Kb whereas the biggest is 300 Mb.

Heterogeneous collection poses new challenges: (i) for CO queries, DTD-independent methods should be developed; (ii) for CAS queries, there is the problem of mapping structural conditions from one DTD onto other (possibly unknown) DTDs. Methods from federated databases could be applied here, where schema mappings between the different DTDs are defined manually. However, for a larger number of DTDs, automatic methods must be developed, e.g. based on ontologies.

This year, different topic types were defined for answering the different retrieval challenges in heterogeneous collections [3]:

- CO queries;
- BCAS (Basic CAS) queries : these topics focus on the combination of singular structural constraints with a content-based constraint;
- CCAS (Complex CAS) queries : they are the het track equivalent of the CAS topics of the ad-hoc track, specified using the NEXI language;

- ECCAS (Extended Complex CAS) queries : these topics assume that the user is able to express the probability of the likelihood of a given structural constraint.

4.2 Experiments

As this is the first year the het track is proposed in the INEX framework, the track was mainly explorative. Participants to the Het track proposed 10 Co topics, 1 BCAS topic and 13 CCAS topics.

As the index structure of XFIRM is designed to handle heterogeneous collections, the indexing process was quite easy. We submitted one run per topic category. For CO queries, we used the same formula as for the ad-hoc task. For BCAS and CCAS queries, a new Dict index was built manually, comparing the different DTDs.

Results are not known yet.

5. PERSPECTIVES

For this year INEX evaluation campaign, we have submitted runs for both the ad-hoc task and the heterogeneous track. Runs were performed with the XFIRM system using a propagation method.

Results obtained for the CO task are lower than those obtained with the same parameters on INEX 2003 CO topics. Results obtained for the VCAS track are relatively good when using a very extended Dictionary index. In both cases, we need to conduct more experiments to evaluate the impact of all our parameters on all INEX metrics.

Concerning the Heterogeneous track, the task this year was mainly explorative. Runs we submitted were performed using a Dictionary index built manually by comparing the different DTDs. Some het challenges are still open: how can we build a Dictionary index automatically? Do we need to adapt our formulas for taking into account the gap between document sizes of the different collections?

6. REFERENCES

- [1] Guidelines for topic development. Proceedings of INEX 2003, Dagstuhl, Germany, december 2003.
- [2] V. N. Anh and A. Moffat. Compression and an ir approach to xml retrieval. In *Proceedings of INEX 2002 Workshop, Dagstuhl, Germany*, 2002.
- [3] V. Dignum and R. van Zwol. Guidelines for topic development in heterogeneous collections. Guidelines of INEX 2004, 2004.
- [4] M. Fernandez, A. Malhotra, J. Marsh, M. Nagy, and N. Walsh. Xquery 1.0 and xpath 2.0 data model. Technical report, World Wide Web Consortium (W3C), W3C Working Draft, may 2003.
- [5] N. Gövert, M. Abolhassani, N. Fuhr, and K. Grossjohann. Content-oriented xml retrieval with hyrex. In *Proceedings of the first INEX Workshop, Dagstuhl, Germany*, 2002.
- [6] T. Grabs and H. Scheck. ETH zürich at INEX: Flexible information retrieval from XML with powerdb-xml. In *Proceedings of the first INEX Workshop, Dagstuhl, Germany*, 2002.
- [7] 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.
- [8] G. Kazai, M. Lalmas, and A. P. de Vries. The overlap problem in content-oriented XML retrieval evaluation. In *Proceedings of SIGIR 2004, Sheffield, England*, pages 72–79, July 2004.
- [9] K. Sauvagnat. Xfirm, un modèle flexible de recherche d’information pour le stockage et l’interrogation de documents xml. In *Proceedings of CORIA’04 (Conférence en Recherche d’Information et Applications), Toulouse, France*, pages 121–142, march 2004.
- [10] K. Sauvagnat and M. Boughanem. The impact of leaf nodes relevance values evaluation in a propagation method for xml retrieval. In R. Baeza-Tates, Y. Marek, T. Roelleke, and A. P. de Vries, editors, *Proceedings of the 3rd XML and Information Retrieval Workshop, SIGIR 2004, Sheffield, England*, pages 13–22, July 2004.
- [11] K. Sauvagnat and M. Boughanem. Le langage de requête xfirm pour les documents xml: De la recherche par simples mots-clés à l’utilisation de la structure des documents. In *Proceedings of Inforsid 2004, Biarritz, France*, may 2004.
- [12] K. Sauvagnat, M. Boughanem, and C. Chrisment. Searching XML documents using relevance propagation. In A. Apostolico and M. Melucci, editors, *SPIRE 04, Padoue, Italie*, pages 242–254. Springer, 6-8 october 2004.

A Test Platform for the INEX Heterogeneous Track

Serge Abiteboul
INRIA Futurs
France
serge.abiteboul@inria.fr

Ioana Manolescu
INRIA Futurs
France
ioana.manolescu@inria.fr

Benjamin Nguyen
PRISM, Univ. Versailles
France
benjamin.nguyen@prism.uvsq.fr

Nicoleta Preda
INRIA Futurs
France
nicoleta.preda@inria.fr

ABSTRACT

This article presents our work within the INEX 2004 Heterogeneous Track. Our focus within this track has been on taming the structural diversity within the INEX heterogeneous bibliographic corpus.

We demonstrate how semantic models and associated inference techniques can be used to solve the problems raised by the structural diversity within a given XML corpus. Our approach starts by automatically extracting a set of *concepts* from each class of INEX heterogeneous documents. We then compute an *integrated set of concepts*, which synthesizes the interesting concepts from the whole corpus. We connect individual corpora to the integrated set of concepts via *conceptual mappings*. This approach is implemented as an application of our KADOP platform for peer-to-peer warehousing of XML documents. While our work caters to the structural aspects of XML information retrieval, the extensibility of the KADOP system makes it an interesting test platform in which components developed by several INEX participants could be plugged, exploiting the opportunities of peer-to-peer data and service distribution.

1. CONTEXT

Our work is situated in the context of the INEX Heterogeneous Track (which we will denote as *het-track* throughout this paper). The *het-track* is very young: has been held in 2004 for the first time. The *het-track* has built a collection of *heterogeneous data sets*, all representing bibliographic entries in various encodings. This collection includes:

- Berkeley: The particularity of this data set is to include *several* classifications or codes for each entry.
- CompuScience: Bibliographic entries in CompuScience format.
- BibDB Duisburg: Bibliographic data from the Duisburg university.

- DBLP: The well-known database and logical programming data source.
- HCIBIB: Bibliographic entries from the field of Human-Computer Interaction.
- QMUL

A set of topics have also been proposed, which are largely similar (in structure and scope) to those formulated within the relevance feedback track. The topics include:

- *Content-only (CO)* topics, of the form “database query”; XML fragments pertinent to the specified keywords must be returned.
- *Content-and-structure (CAS)* topics, such as `//article[about(./body, "XML technology")]`
In this case, the search for pertinent data fragments is confined by specific structural criteria.

Answering an IR query on a structurally heterogeneous corpus raises two main challenges. First, computing the relevance of a data fragment for a given keyword or set of keywords; this task is no different from the main relevance assessment track. Second, taking into account the structural hints present in the topic, in the case of CAS topics.

In the presence of a heterogeneous corpus, the second task becomes particularly difficult. This is due to the fact that semantically similar information is encoded in very different XML formats; furthermore, DTDs may or may not be available for the corpus. The work we present has specifically focused on this second task.

Contributions

Our work within the *het-track* makes the following contributions.

First, we present an approach for *integrating the heterogeneous structures* of the heterogeneous data sources under an *unified structure*. This approach relies on simple semantic-based techniques, and on our experience in building semantic-based warehouses of XML resources [2, 3]. The result of this integration on the *het-track* corpus is an *unified DTD*, and a *set of mappings* from individual sources to this DTD. CAS topics against the *het-track* corpus can now

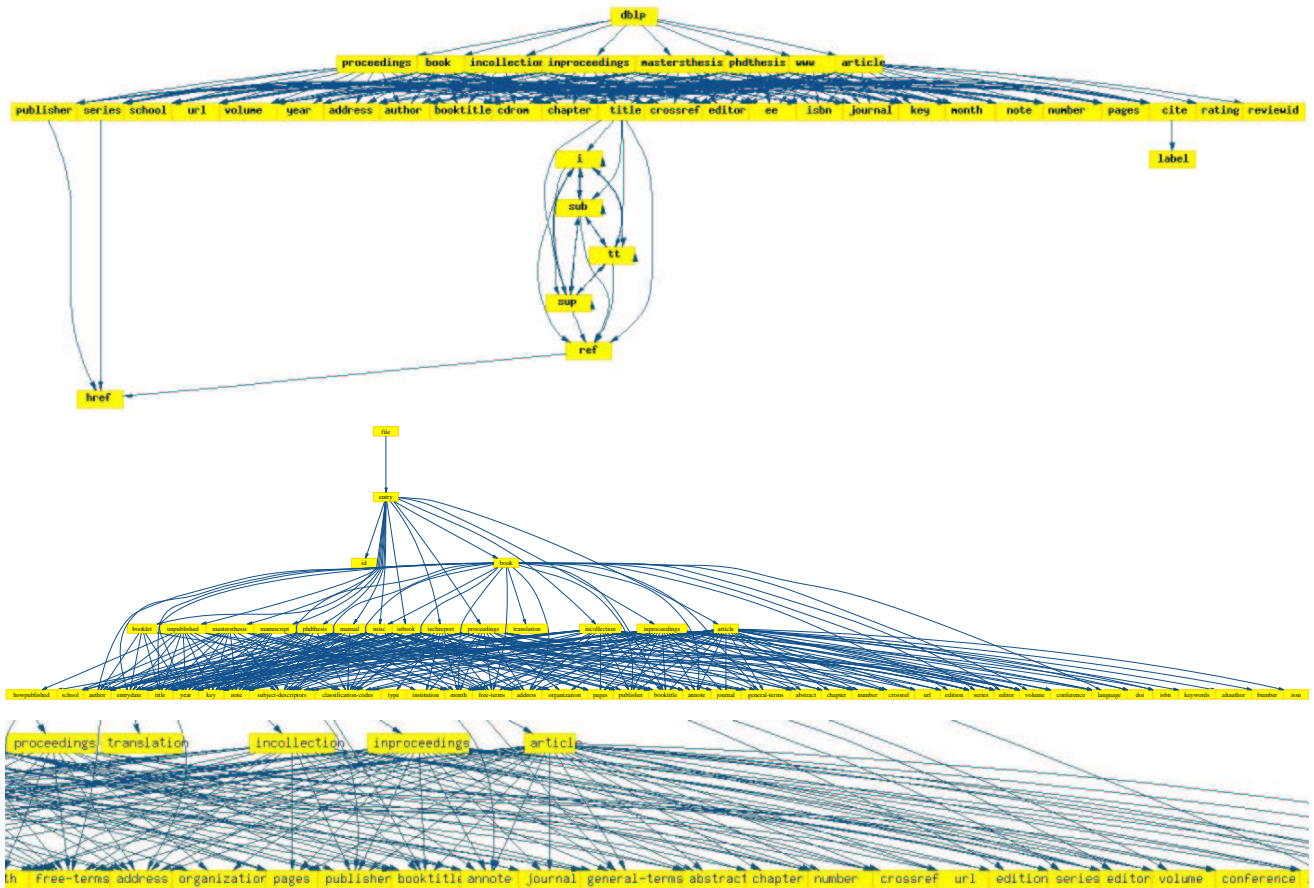


Figure 1: XSum drawing of the DBLP DTD (top), Duisburg DTD (middle), and zoom-in on Duisburg DTD articles (bottom).

be expressed in terms of the unified DTD, and get automatically translated into a union of topics over each data set. Thus, solving a CAS topic on a heterogeneous corpus is reduced to solving several CAS topics against the individual component data sets.

Second, we present XSum [11], a free XML and DTD visualization tool, that we developed as part of our work in INEX. XSum helped us get acquainted to the complex structure of the heterogeneous collection, and devise semi-automatic integration strategies.

Finally, we outline the architecture of a peer-to-peer platform for processing XML queries or IR searches, over a set of distributed, potentially heterogeneous XML data sources. This platform has the advantage of being *open* and *inherently distributed*, allowing to take advantage of the data sources and capabilities of each peer in the network in order to solve a given query or search. In particular, we show this platform may be used as a testbed for the XML IR methodologies developed within the het-track, by allowing to test and combine the various implementations of the about functions developed by INEX participants.

This document is structured as follows. Section 2 describes our semantic-based approach for XML information retrieval over a heterogeneous corpus. Section 3 details the result we obtained by applying this approach on the INEX het-track corpus. Section 4 outlines the peer-to-peer generic platform we propose, and describes how it could be used as a testbed for the het-track in the future. Section 5 draws our conclusion and outlines future work.

2. OUR APPROACH FOR HETEROGENEOUS XML INFORMATION RETRIEVAL

Dealing with structural diversity in heterogeneous sources has been a topic of research in the field of databases and in particular of *data integration*. The purpose of a data integration system is to provide the user the illusion of a single, integrated database, on which the user can pose queries (in our case, IR queries, or topics). Behind the uniform interface, the system will process these queries by translating them into the formats specific to each data source, processing them separately, and integrating the results into a single one.

Traditionally, data integration operates at the level of *schemas*.

A source schema characterizes the structure of each data source, and an integrated schema is provided to the user. This approach has been thoroughly investigated in the case of relational data sources and schemas.

In the case of heterogeneous, complex, potentially schema-less data sources, this approach is no longer applicable. Instead, we chose to draw from the experience obtained in *semantic-based data integration* [?], to integrate sources pertinent to a specific domains, such as the het-track corpus, under a single *conceptual model*. The building bricks of our conceptual model are:

- *Concepts*, which are the notions of relevance for a given application. For instance, in the het-track corpus, useful concepts are: “publication”, “author”, etc.
- *IsA* relationships represent specialization relationships between concepts. For instance, “book IsA publication” represents the fact that books are a kind of publication.
- *PartOf* relationships represent composition (aggregation) relationships between concepts. For instance, “title PartOf book” represents the fact that a title is a component of a book.

It has been noted [4] that XML DTDs are a good basis for a conceptual model of the XML documents conforming to the DTDs. Thus, our approach starts by extracting a conceptual model from each source. For the sources for which DTDs are available, the process is straightforward: we extract a concept for each type in the DTD, including element and attributes (among which we do not make a distinction). For sources for which DTDs are not available, we start by extracting a “rough” DTD, including all element names. Furthermore, whenever we encounter in the data an element labeled l_1 as a child of an element labeled l_2 , we mention in the DTD that the type l_1 can appear as a child of the type l_2 . After having extracted this DTD, we compute from it a set of concepts as in the previous case.

At the end of this stage, we have obtained a set of conceptual data source models. Our purpose then is to construct a unified conceptual model characterizing all sources, and mappings between each conceptual model to the unified one.

Extracting the unified conceptual model

To build the unified conceptual model, we identify groups of concepts (each one in different conceptual source models) that represent semantically similar data items. We do this in a semi-automatic manner, as follows.

First, the names of concepts from different source models are compared for similarity, to identify potential matches. This can be done automatically, with the help of a tool such as WordNet [5]. If simple matches such as the one between “book” (DBLP) and “book” (HCI BIB) can be automatically detected, more subtle ones such as the similarity between “editor” (HCI BIB) and “Edition” (Berkeley) require the usage of tools such as WordNet. Having identified clusters of concepts which potentially represent the same thing,

we create one concept in the unified model, for each cluster of source model concepts above a given similarity threshold; human intervention is required at this point in setting the similarity threshold.

At the end of this process, it may happen that some source model concepts have not been clustered with any others. This may be the case, for instance, of concepts called “Fld012”, “Fld245”, etc. from the Berkeley data source. These concepts are difficult to cluster, since their names (standing for “Field number 012”, “Field number 245”, etc.) do not encapsulate the meaning of the concept, instead, this meaning is included in plain-text comments prior to the DTD description of the respective type. To deal with such concepts, we need to capture the DTD comments preceding the type, and feed those descriptions to the word similarity-based clustering. This way, we may learn that “Fld245” stands for “Title Statement”, and cluster “Fld245” with similarly named concepts from other DTDs.

Once the clusters of similar (and supposedly semantically close) concepts have been extracted, we create a concept for each such cluster, in the unified conceptual model.

Extracting mappings between the source and unified conceptual models

We add an ISA relationship going from each source model concept, to the unified model concept that was derived from its clusters. If a source model participates to several clusters, this may yield several ISA relationships.

3. RESULTS OF OUR PARTICIPATION TO INEX

In this section, we report on the results that we obtained in our work in the framework of the het-track.

3.1 Unified conceptual model for the INEX corpus

In this section, we discuss the techniques used to construct a unified DTD in order to query the documents of the heterogeneous track. An important factor is that the documents are all about the same topic : a bibliography. Five of them are quite verbose, and the labels are self descriptive, while one (Berkeley DB) has labels that convey no semantic significance whatsoever. Some examples of such labels would be : Fld001, Fld002, etc...

In this article, we do not take into account this DTD, therefore the unified DTD we propose does not include elements from the Berkeley DB for the moment. We are currently experimenting the use of a tool that displays the DTD graph, by clustering together labels that have the same parent, in order to bring some semantics into this DTD. We intend to propose groups of labels and compare them with the existing element clusters in order to determine their semantics.

The method used in order to determine a unified DTD is the following :

- We first of all create mappings between elements that have the same syntax, but that originate from differ-

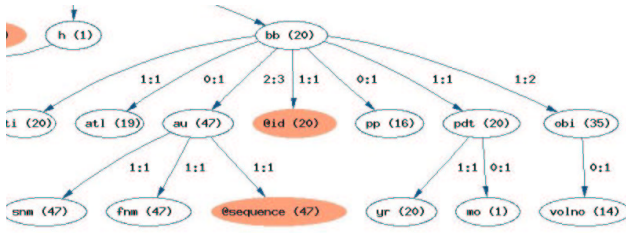


Figure 2: Fragment of a path summary computed from an article in the INEX main corpus (IEEE CS).

ent DTDs. For instance, we might find two *article* elements, one from DBLP, the other from BibDB Duisburg. If there are several elements with the same (or very close) syntax in multiple DTDs, we group them all together. These are one to one mappings, and the output of this phase is a group of clusters of syntactically close elements.

- For each cluster, we then check the parent nodes, and group them together in a new parent cluster.
- For all these automatically constructed clusters, we manually check the correctness of these groupings, and chose a name for the cluster, generally of the form `nameOfElementC`.

We give on our website the full resulting DTD.

Using the unified DTD : The Unified DTD is to be used when asking queries over the heterogeneous data set. The querying mechanism is as follows.

- The INEX queries must be written taking into account the unified DTD. The Unified DTD elements represent the predicates to be used in the path expressions. We call this query a *generic query*.
- The generic query is then converted into specific queries, with a specific structure for each database, and the queries are then run separately on all the databases.
- Given the unified DTD, the answers returned are clustered together in a common structure, in order to use only a single DTD for browsing means.

Further steps: We intend on focusing on merging the BerkleyDB into the Unified DTD, and we intend to propose an easy to use integration platform, in order to include any other bibliographical semi-structured database, by incrementally clustering its elements with those already clustered.

3.2 XSum: a simple XML visualization tool

We have developed a simple XML visualization tool, called *XSum* (from *XML Summary Drawer*). XSum can be used in two ways.

First, if given an XML document, with or without a DTD, XSum extracts a tree-shaped structural summary of the document, and draws it. This structural summary contains a node for each distinct path in the input document [7], and is the equivalent of a strong DataGuide [6] for XML data (DataGuides were initially proposed for graph-structured OEM data). XSum enhances this structural representation with:

- Node counts: XSum records the number of nodes on a given path in the XML document, and correspondingly may show this number in the summary node corresponding to that path.
- Leaf types: XSum attempts to “guess” the type (String, integer or real number), of each leaf node, whether #PCDATA or attribute value, and depicts the corresponding node in a color reflecting its type.
- Edge cardinalities: XSum records the minimum and maximum number of children of a given tag, that a node on a given path may have. XSum may depict these numbers on the edge connecting the two corresponding summary nodes.

A sample summary representation produced by XSum from a XML-ized article from the INEX IEEE CS corpus is depicted in Figure 2. The fragment shown here reflects the references at the end of an article, including authors, titles, and publication information for the relevant references.

Second, when given a DTD, XSum draws a simple graph, representing each attribute or element type from the DTD as a node, and adding an edge from a node to another whenever a type may appear inside another in the DTD. Figure 1 shows the drawing extracted by XSum from the DTD of the Duisburg data source, and a zoomed-in fragment around the node corresponding to the “article” type in that data source.

From our experience using XSum with the INEX standard and heterogeneous corpus, we draw some remarks. First, graph corresponding to DTDs tend to have relatively few nodes, but large number of edges, which cross each other in the drawing (Figure 1), which may make the image difficult to read. In contrast, graphs derived directly from the data are guaranteed to be tree-shaped, and thus planary (no edge crossing). Second, both DTD and summary drawings tend to be large for documents of the complexity we are dealing with, typically larger than the screen or a normal printer format. Understanding the image requires “sliding” over it to see one part at a time. We have introduced in XSum some options which allow to omit leaf nodes and/or cardinality annotations, which simplifies the graphs. We welcome the feedback of INEX participants on how to modify the graph drawing logic to produce better images.

XSum is implemented in Java, and is based on GraphViz, a well-known graph drawing library developed at AT&T,

freely available under the BSD licence. XSum is freely available and can be downloaded from [11]. The graphs produced by XSum, for all DTDs in the het-track corpus, are available at [9].

4. PEER-TO-PEER SEARCH AND XML INFORMATION RETRIEVAL PLATFORM

In this section, we briefly describe the KADOP peer-to-peer XML resources management platform, which serves as the framework for our work. A more detailed presentation can be found in [3].

The KADOP platform allows constructing and maintaining, in a decentralized, P2P style, a warehouse of *resources*. By resource, we mean: data items, such as XML or text documents, document fragments, Web services, or collections; semantic items, such as simple hierarchies of concepts; and relationships between the data and semantic items. KADOP's functionality of interest to us are:

- *publishing* XML resources, making them available to all peers in the P2P network;
- *searching* for resources meeting certain criteria (based on content, structure as well as semantics of the data).

KADOP leverages several existing technologies and models. First, it relies on a state-of-the-art Distributed Hash Table (DHT) implementation [10] to keep the peer network connected. Second, it is based on the ActiveXML (AXML) [8] platform for managing XML documents and Web services. A full description of ActiveXML is out of the scope of this work, see [1]. For our purposes here, AXML is an XML storage layer, present on each peer.

The KADOP data model comprises the types of resources that can be published and searched for in our system. We distinguish two kinds of resources: *data items*, and *semantic items*. *Data items* correspond to various resource types:

- A *page* is an XML document. Pages may have associated *DTDs* or *XML schemas* describing their type; we treat DTDs as sources of semantic items (see further). Other formats such as PDF can be used; we ignore them here.
- We consider data with various granularities. Most significantly, we model: *page fragments*, that is, results of an XML query on a page, and *collection*, as user-defined sets of data items. Inside pages, we also consider element *labels*, attribute *names*, and *words*.
- Finally, a *web service* is a function taking as input types XML fragments, and returning a typed XML fragment.

Any data item is uniquely identified by an PID (peer ID) and a *name*. The PID provides the unique name (logical identifier) of the peer that has published the data item, and where the item resides; names allow distinguishing between data items within the peer. Data items are connected by

PartOf relationships, in the natural sense: thus, a word is part of a fragment, a fragment part of a page etc. Furthermore, any type of data items can be part of collections. A data item residing on one peer may be part of a collection defined on another peer.

Semantic items consist of *concepts*, connected by two types of relationships: IsA, and PartOf. A graph of concepts, connected via IsA or PartOf links, is called a *concept model*. We derive a source concept model from each particular data source, as described in Section 2.

InstanceOf statements connect data items with concepts. In particular, all elements from an XML document, of given type τ (obtained as the result of the XPath query $//\tau$), are implicitly connected by InstanceOf statements to the concept derived from the type τ .

The KADOP query language allows retrieving *data items*, based on constraints on the data items, and on their relationship with various concepts. Queries are simple tree

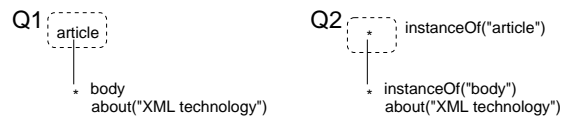


Figure 3: Sample KadoP queries.

patterns, and return the matches found for a single query node (in the style of XPath and the CAS INEX topics). For instance, the query in Figure 3 at left allows retrieving all “article” elements such that they have a “body” element, and the body is about XML technology. This corresponds to the sample CAS topic in Section 1. The dashed box designates the node for which matches will be returned.

Such a query, however, needs specific names (that is, element tags) for its nodes. In the case of the heterogeneous corpus, such queries are no longer helpful, due to the varied structures encountered in different documents.

The approach we take for solving INEX heterogeneous CAS topics is based on the unified conceptual model. The idea is to drop name conditions from the queries, and instead use conditions of the form “instanceOf c ”, where c is a concept from the unified model. On our example query, this leads to the KadoP query at right in Figure 3, where we assume that “article” and “body” are part of the unified conceptual model. This query is processed as follows:

1. The elements directly declared as instance of the concepts “article” and “body” are found.
2. We search for concepts c_a such that c_a IsA “article”, and concepts c_b such that c_b IsA “body”. This will lead to retrieving all the concepts from the source concept models, which have been mapped to the unified concepts “article” and “body”.
3. We search for elements declared as instances of the concepts c_a and c_b obtained as above.

These steps lead to matching the structural conditions posed by the CAS query against the heterogeneous corpus. They do not, however, apply the “about” condition, since implementing this condition is out of the scope of our work. We next explain how others’ implementations of the “about” function could be plugged in our work.

Integrating “about” functions

In the KADOP framework, “about” can be integrated as a Web service, offered by one or several peers. The implementation of this function is typically complex. From the KADOP perspective, all that is needed is that one or several participants make available a Web service named “about”, obeying to a well-defined interface. Then, the KADOP query processor can invoke one of these services to evaluate the pertinence of an XML fragment for a given set of keywords. The user may specify which service to use; this is helpful when we want to compare the results of different implementations. Or, she may let the system choose an implementation.

It is worth stressing that the KADOP framework is based on a concept of *openness* and *extensibility*: new data sets, new concepts, or new semantic statements can be added by any participant, and refer to any data item on any peer. Finally, the KADOP framework is by nature distributed: any Web service (thus, any “about” function) can be invoked on XML fragments originating from any peer.

5. CONCLUSION AND PERSPECTIVES

6. REFERENCES

- [1] Serge Abiteboul, Omar Benjelloun, and Tova Milo. The activexml project: an overview. Gemo research report no. 344, 2004.
- [2] Serge Abiteboul, Gregory Cobena, Benjamin Nguyen, and Antonella Poggi. Construction and maintenance of a set of pages of interest (spin). In *Bases de Donnees Avancees*, Evry, 2002.
- [3] Serge Abiteboul, Ioana Manolescu, and Nicoleta Preda. Constructing and querying a peer-to-peer warehouse of XML resources. In *Proceedings of the Semantic Web and Databases Workshop (in collaboration with VLDB)*, Toronto, CA, 2004.
- [4] Sophie Cluet, Pierangelo Veltri, and Dan Vodislav. Views in a large scale XML repository. In *VLDB*, 2001.
- [5] Christine Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
- [6] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *VLDB*, pages 436–445, Athens, Greece, 1997.
- [7] I. Manolescu, A. Arion, A. Bonifati, and A. Pugliese. Path Sequence-Based XML Query Processing. In *Bases de Donnees Avancees (French database conference)*, Montpellier, France, 2004. Informal proceedings only.
- [8] The ActiveXML home page. Available at www.axml.net, 2004.
- [9] Gemo and PRiSM at the inex heterogeneous track. Available at www-rocq.inria.fr/gemo/Gemo/Projects/INEX-HET, 2004.
- [10] The FreePastry system. Available at www.cs.rice.edu/CS/Systems/Pastry/FreePastry/, 2001.

- [11] XSum: The XML summary drawer. Available at www-rocq.inria.fr/gemo/Gemo/Projects/SUMMARY, 2004.

EXTIRP 2004: Towards heterogeneity

Miro Lehtonen

Department of Computer Science
P. O. Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki
Finland

Miro.Lehtonen@cs.Helsinki.FI

ABSTRACT

The effort around EXTIRP 2004 focused on the heterogeneity of XML document collections. Since the subcollections of the het-track did not offer us a suitable testbed, we successfully applied methods independent of any document type to the INEX test collection. By closing our eyes to the DTD, we created comparable runs and discovered that the results improved. Some problematic areas were also identified. One of them is score combination which enables us to return bigger elements as answers given the relevance scores for the smaller ones.

1. INTRODUCTION

One of our goals for the INEX 2004 project was to adapt our system to support heterogeneous XML collections without losing the retrieval accuracy achieved in 2003. Our system for XML retrieval — EXTIRP — has now been successfully modified: it is independent of any document type and, based on tests with the topics of 2003, the accuracy has even improved. However, not all components of EXTIRP adjusted to the changes equally well. For example, the score combination algorithm of EXTIRP showed its weakness in a significant decline in average precision. Consequently, the best result sets consisted of disjoint answers at the finest level of granularity. Another factor having a negative impact on EXTIRP 2004 was the lack of manpower due to which we gave up a previous success story: query expansion.

This paper is organised as follows. XML terminology and related vocabulary have had various interpretations in the history of INEX. Section 2 clarifies the terminology that is necessary in order to fully understand the rest of the paper. In Section 3, EXTIRP is briefly described. The problem of too big documents is addressed in Section 4, and the challenge of score combination in Section 5 (unfinished). Our runs for 2004 topics are described in Section 6 after which we draw the conclusions in Section 7.

2. COMMON MISCONCEPTIONS

The purpose of this section is to make our paper accessible to readers who are not familiar with XML terminology. Because of the confusion with vocabulary, the INEX document collection is often misrepresented, see [5, 6]. We will now explain what the collection looks like to XML-oriented people in order to have a common terminological basis with the reader.

Number of XML documents. An *XML Document* is the biggest logical unit of XML. It can be stored in several XML files which are part of the physical structure of the document. When parsed into DOM¹ trees, each XML document only has one Document Node in the tree. The DOM trees representing the whole INEX collection have 125 Document Nodes because the collection consists of 125 XML documents. Each XML document contains one volume of an IEEE journal.

Number of articles. The concept of a *document* has changed because of XML. A document is no longer considered the atomic unit of retrieval. However, XML should have no effect on the concept of an *article*. It is true that there are 12,107 `article` elements in the document collection, but the number of articles is smaller. According to the common perception, many article elements do not have article content. Instead, they contain a number of other papers such as errata, lists of reviewers, term indices, or even images without any text paragraphs.

Number of tags. The specification for XML² defines three different kind of tags: start tags, end tags, and empty element tags. A DTD does not define any tags, but it does define *element types*. In the XML documents, though, each non-empty element contains two different tags. Counting the different tags in the collection (361) is very different from counting the different element type definitions in the DTD (192), different element types inside the article elements (178), or different element types in the whole collection (183).

Number of content models. The content models of the collection are defined in the DTD. Each element type definition contains the name of the element followed by its content model. Altogether 192 content models are defined in the DTD, but only 65 of those are different. Of the 65 different content models, only 59 appear in the articles of the collection. For example, the content models of element types `journal` and `books` are not allowed in article content, and elements such as `couple`, `line`, and `stanza` are not included in the collection at all.

DTD-independent methods. One of the goals of the het-track has been the development of DTD-independent

¹<http://www.w3.org/DOM/>

²<http://www.w3.org/TR/REC-xml/>

methods although methods that are independent of the DTD are necessary only when no DTD is available. The problem of documents with several different DTDs or schema definitions is far more common. Methods that are independent of the document type may read the DTDs or schema definitions and try to make the most of it. For example, link relations inside one document are easily resolved with a DTD by reading which attributes are of the ID type and which of the type IDREF or IDREFS. Moreover, independence of the DTD does not exclude the dependence on a schema definition.

3. SYSTEM OVERVIEW

EXTIRP specialises in full-text search of XML documents and does not support any structural conditions in the queries. Only full-text is indexed, and only full-text is queried. It is thus natural to focus on CO-type topics when evaluating our system.

EXTIRP uses two static indices: an inverted word index and an inverted phrase index. The *key* in the word index is the identifier of a stemmed word and the corresponding *value* contains a list of XML fragment identifiers indicating where in the collection the word occurs. The phrase index contains similar information about those phrases that are considered *Maximal Frequent Sequences* [2].

Before the static indices are built, we divide the document collection into document fragments which results in a fragment collection. Each fragment represents the smallest atomic unit of content that can be retrieved. In other words, only the finest level of granularity is indexed. The fragments are not leaf nodes in the document tree but whole sub-trees that contain element and text nodes. How the indexed fragments are selected is described in Section 4 in more detail.

Upon query processing, two normalised similarity scores are computed for each fragment: word similarity and phrase similarity. These two scores are aggregated into a Retrieval Status Value (RSV), according to which the fragments are ranked. At this point, the result list contains a ranked list of relatively small answers for each query. By combining the scores of these fragments, we can replace them with bigger fragments in the list. For example, the score of a section is computed using the scores of each child element, e.g. a paragraph, inside the section. If the section seems more relevant than the paragraphs in the light of RSVs, the paragraph-size fragments are replaced with the particular section-size fragment and ranked accordingly. By combining the scores of adjacent fragments, all the scores are propagated upward in the document hierarchy all the way to the article level. This process has turned out remarkably challenging with the fragment collections of 2004.

A more detailed description of the 2003 version of EXTIRP was presented in [4]. A novelty in EXTIRP 2004 is its independence of any document type. The major changes from 2003 are described in Section 4.2.

4. DIVISION INTO FRAGMENTS

In recent research, several different purposes have been presented for dividing structured documents into fragments.

We will first briefly look into the state of the art and see whether these algorithms could be applied to XML retrieval. Then we will describe how EXTIRP selects the fragments to be indexed.

4.1 Related work

Ramaswamy et al. presented a fragment detection algorithm motivated by performance issues [9]. Web pages were divided into fragments according to certain criteria, e.g. how many other fragments share its content, whether it is maximal or content of another fragment, and how frequently the content is updated in comparison with other fragments on the web page. A minimum size was also set on the qualifying fragments which were considered cost-effective cache units. This algorithm is not directly applicable to a static collection of XML documents because we cannot measure any lifetime characteristics in an unchanging set of documents. Moreover, the potential fragments in the INEX test collection are unique and not shared among other fragments. Some noteworthy details in their studies include the *minimum size of a detected fragment* which is used to exclude the smallest segments of web pages from being detected as candidate fragments. The values of 30 bytes and 50 bytes seemed reasonable in their experiments. Note also the unit of the minimum size: it is not the number of words, terms, or tokens but simply the bytesize which roughly corresponds to the number of characters in the element content.

In 1999, Jon Kleinberg introduced the HITS algorithm [8] that categorises web pages into *hubs* and *authorities*. A page with a good collection of links has a high hub score whereas a popular page or an authoritative source of information has a high authority score. However, hubs rarely contain links related to a single topic. On the contrary, hubs can cover wide ranges of topics which makes them *mixed hubs*. Chakrabarti developed an algorithm that disaggregates web pages considered mixed hubs into coherent regions by segmenting their DOM trees [3]. He uses the HITS algorithm for topic distillation by computing the hub score for each subtree in the DOM tree instead of computing the hub score for the whole document. The resulting fragments are pure hubs that are highly specific answers to appropriate queries. Topic distillation is a proper but not sufficient criterion for dividing XML documents into fragments. Applying Chakrabarti's hyperlink-based algorithm to any of the INEX test collections is, however, pointless.

Another need for document segmentation comes from devices that can only display a small amount of information at a time, either because of a small display size or a low resolution. Hoi et al. developed a document segmentation and presentation system (DSPS) that automatically divides a web page into logical segments [7] based on the display size, and document structure and content. The segments have their own branches in a *content tree* into which HTML documents are first converted. The HTML tags are classified into different categories and interpreted accordingly. Applying this algorithm to arbitrary XML documents requires a thorough analysis of the document type. The maximum size of an HTML segment is also rather small. A major difference from the algorithms of Ramaswamy and Chakrabarti is that the resulting HTML segments do not cover all of the original documents: Segments that are very small or very

different from the adjacent segments are removed from the content tree.

4.2 Size-based division

In 2003, the indexed fragments were selected by the name of the corresponding XML element. After carefully studying the DTD of the collection, we could see which element types represented section-level fragments (sec, ss1, ss2, etc.) and which element types were common at the paragraph-level (p, ip1, etc.). Similar approaches have been common among other participants. For example, the selection of index nodes in HyREX system is strictly based on element names [1]. Approaches relying on element names do not scale well to fit the needs of heterogeneous document collections. Analysing each DTD is hardly an option as the number of document types increases. Furthermore, it will shortly be shown that better results can be achieved with methods that are independent of the document type.

The EXTIRP algorithm for dividing XML documents into fragments has two parameters: the maximum and minimum size of an indexed fragment. The fragments are selected by traversing the document tree in preorder. If the current node is small enough and qualifies as a full-text fragment, it will be added to the fragment collection, the following node will be tested. The fragments in the fragment collection are disjoint because all the subtrees of qualifying fragments are skipped. If the current node does not qualify, its children will be tested until they are either small enough or too small. Consequently, irrelevant fragments, e.g. those that are unlikely answers to any full-text query, are discarded in a similar fashion that the DSPS by Hoi et al. removes irrelevant segments. The algorithm on the whole is independent of any document type and also applicable to the INEX test collection.

We compare the precision of four different runs drawn with a solid line in Figure 1. Three of the runs were our official submission in 2003, and the fourth one is based on a fragment collection with the minimum size of a fragment set to 150 characters and maximum to 8,000 characters in text nodes. The other curves represent the official submissions of other participants for the CO topics. Only the first 100 recall answers of each run are considered here. The run that was based on EXTIRP 2004 methods shows the best performance (see the thickest solid line). Our official runs of 2003 have a significantly lower precision at most recall levels. The curves for the generalised quantisation show similar results.

More evidence for the results can be found in Table 1 where the Generalised Recall measure is shown for each run. The online evaluation tool³ was used for computing the GR score. No score combination method is applied to the disjoint fragments in the run 'Fragments150-8k', which shows in the 0.0 List-Based Overlap (LBO).

5. SCORE COMBINATION

A method called Upward Propagation with the Upward Propagation Factor (UPF) as a parameter.

A fragments collection was created with the maximum size

³<http://inex.lip6.fr/2004/metrics/>

UPF	strict -o	strict -s	generalised -o	generalised -s
2.0	0.0058	0.0057	0.0081	0.0079
1.0	0.0270	0.0288	0.0277	0.0305
0.7	0.0536	0.0437	0.0500	0.0446
0.6	0.0584	0.0443	0.0451	0.0379
0.5	0.0565	0.0408	0.0395	0.0318
0.4	0.0546	0.0379	0.0351	0.0275
0.2	0.0509	0.0351	0.0294	0.0219
—	0.0954	0.0728	0.0705	0.0562

Table 2: Upward propagation applied to a fragment collection "Fragments200-20k".

set to 20,000 and minimum size to 200 characters. After the RSV was computed for each fragment, the score combination method was applied. The results with different values for the UPF are shown in Table 2. The best average precision was achieved without score combination.

6. OUR RUNS

The results presented in earlier sections of this paper were not available at the time of run submission. We have, however, learned since then and found reasons for the decline in the precision of the three runs for the CO topics that we submitted. Three different factors have been identified:

Query expansion Although query expansion turned out to improve the results in 2003, we did not have enough resources to repeat the success.

Score combination As seen in Section 5, the results deteriorated after the score combination process. Adjusting the parameters did not have a great impact on the results.

Query processing Only the title of the topic was used for the runs of 2004. In 2003, also the description of the topic was used for similarity computation. This change should not show in the overall results because it concerned all the participants of 2003.

We did not submit any runs for the het-track topics. As EXTIRP 2004 specialises in full-text search, it also does not index any data-oriented content. The XML documents consisting of bibliographic data have no such full-text content that qualifies for the fragment index. It was not found meaningful to submit runs for a CO topic that would be identical with the corresponding runs for the ad-hoc track.

7. CONCLUSIONS

8. REFERENCES

- [1] M. Abolhassani, N. Fuhr, and S. Malik. HyREX at INEX 2003. In *INEX 2003 Workshop Proceedings*, pages 49–56, Dec. 2003.
- [2] H. Ahonen-Myka. Finding All Frequent Maximal Sequences in Text. In *Proceedings of the 16th International Conference on Machine Learning ICML-99 Workshop on Machine Learning in Text Data Analysis, Ljubljana, Slovenia*, pages 11–17. J. Stefan Institute, eds. D. Mladenic and M. Grobelnik, 1999.

INEX 2003: quantization strict

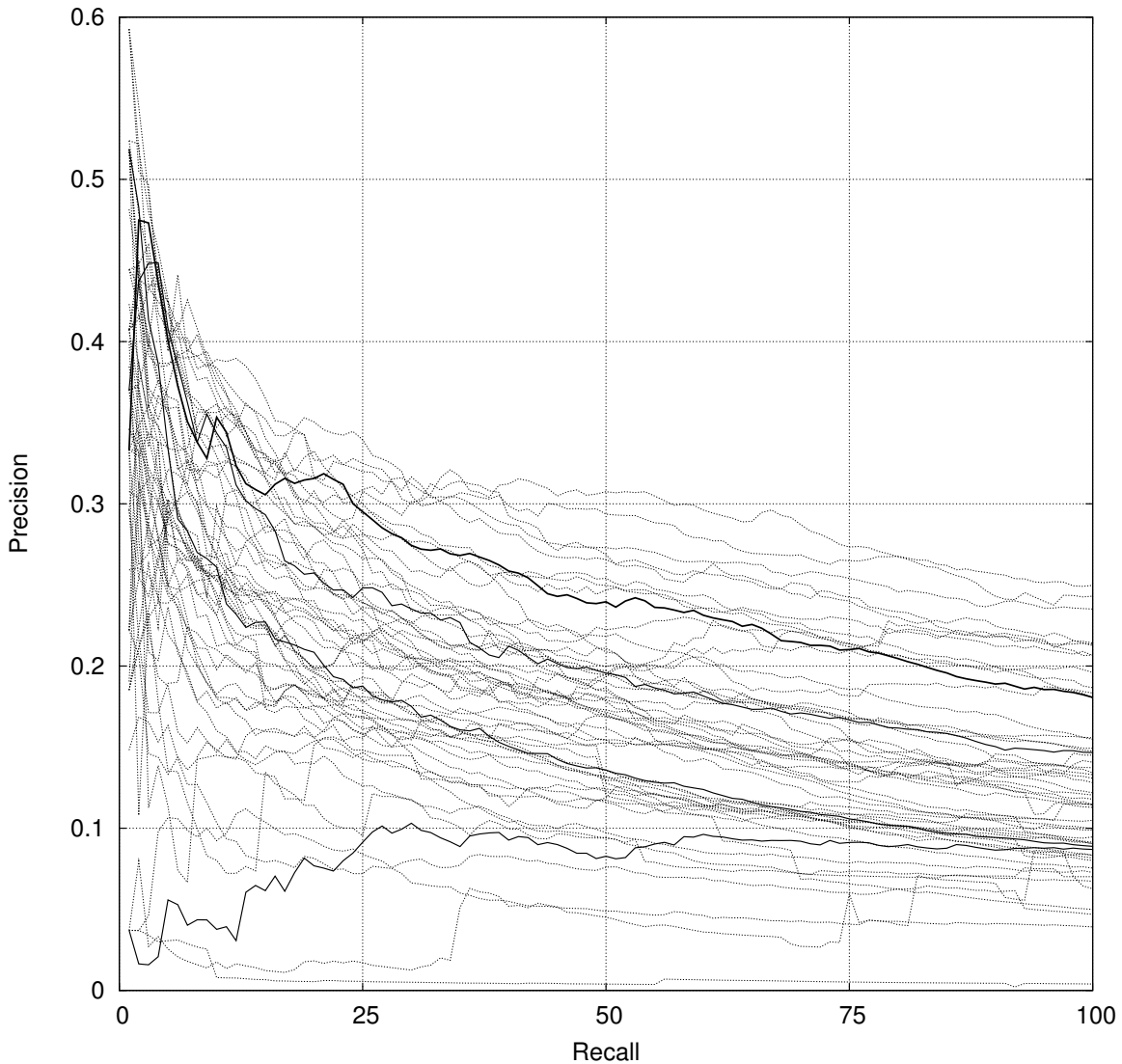


Figure 1: Precision_o of all runs at recall levels 1-100.

Run	LBO	strict -o	strict -s	generalised -o	generalised -s	GR
UHel-Run1	39.6	0.0484	0.0358	0.0340	0.0270	9.70
UHel-Run2	28.0	0.1135	0.0866	0.0716	0.0586	11.63
UHel-Run3	10.5	0.1058	0.0787	0.0537	0.0418	8.60
Fragments150-8k	0.0	0.1170	0.0924	0.0831	0.0658	27.68

Table 1: A run with size-based division and no score combination or query expansion compared with the official runs of University of Helsinki.

- [3] S. Chakrabarti. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In *Proceedings of the tenth international conference on World Wide Web*, pages 211–220. ACM Press, 2001.
- [4] A. Doucet, L. Aunimo, M. Lehtonen, and R. Petit. Accurate Retrieval of XML Document Fragments using EXTIRP. In *INEX 2003 Workshop Proceedings*, pages 73–80, Schloss Dagstuhl, Germany, 2003.
- [5] N. Fuhr, N. Goevert, G. Kazai, and M. Lalmas, editors. *INEX: Evaluation Initiative for XML retrieval - INEX 2002 Workshop Proceedings*, DELOS Workshop, Schloss Dagstuhl, 2003.
- [6] N. Fuhr and M. Lalmas. Report on the INEX 2003 Workshop, Schloss Dagstuhl, 15-17 December 2003. *SIGIR FORUM*, 38(1):42–47, June 2004.
- [7] K. K. Hoi, D. L. Lee, and J. Xu. Document visualization on small displays. In *Proceedings of the 4th International Conference on Mobile Data Management (MDM 2003)*, pages 262–278, Berlin, Germany, 2003. Springer-Verlag.
- [8] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
- [9] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglass. Automatic detection of fragments in dynamically generated web pages. In *13th World Wide Web Conference (WWW - 2004)*, pages 443–454, May 2004.

NLPX at INEX 2004

Alan Woodley
Centre for Information Technology Innovation
Faculty of Information Technology
Queensland University of Technology
GPO Box 2434 Brisbane Q 4001 Australia
ap.woodley@student.qut.edu.au

Dr Shlomo Geva
Centre for Information Technology Innovation
Faculty of Information Technology
Queensland University of Technology
GPO Box 2434 Brisbane Q 4001 Australia
s.geva@qut.edu.au

ABSTRACT

In order to optimally fulfil their information need users of information retrieval (IR) systems require an interface that is powerful yet easy-to-use. This problem is especially applicable IR systems that handle structured documents such as XML, since users expect the system to return only the relevant portions of documents. This paper presents a natural language interface that is user friendly enough to be can be used intuitively, but sophisticated enough to be able to handle complex structured queries. This paper presents a solution to this paper via a natural language interface. The interface accepts queries written in a natural language that express both users content and requirements. It uses a set of grammar templates to derive the structural and content requirements of users. The system was developed for participation in the NLP Track of the INEX 2004 Workshop, its performance results are presented in the results chapter of this paper.

1. INTRODUCTION

The widespread use of Extensible Markup Language (XML) documents in digital libraries has led to development of information retrieval (IR) methods specifically designed for XML collections. While traditional IR systems are limited to whole document retrieval, XML IR systems are able to satisfy both the content and structural needs of users by retrieving only highly relevant information. However, in order to properly satisfy users' needs, IR systems require an interface that is powerful enough to thoroughly express the users information need, but user-friendly enough that it can be used intuitively.

Historically two types of queries have been supported by the INEX Workshop: Content Only (CO) and Content and Structure (CAS), each with their own interface. CO queries only express users' content requirements so their interface consists of a list of keywords. In comparison, CAS queries express both the structural and content requirements of users, and therefore require a more sophisticated interface. To meet this requirement CAS queries have been formatted using complex query languages (XPath [2] in 2003, NEXI [11] in 2004). Unfortunately, in a structured IR system neither interface optimally addresses users' needs. Keyword based systems are too simplistic, since they do not allow users

to express their structural requirements. Alternatively formal query languages are too difficult to use, and require users to have an intimate knowledge of a documents structure.

In this paper we present an alternative interface for XML IR systems, one that allows users to express their need in natural language. This type of interface is very applicable to the INEX collection since each topic already contains a description element that expresses users' content and structural needs in natural language. There already exists an extensive body of research into natural language processing in the specific area of Information Retrieval, largely thanks to The Text Retrieval Conference (TREC) [10] and the Special Interest Group for Information Retrieval (ACM-SIGIR) [9]. However, work on an XML-IR interface is still largely un-documented and many problems remain unsolved. What follows begins by outlining some of the motivating factors for a natural language interface for IR systems. We then present our own NLPX system, which participated in the INEX 2004 NLP Track, including the methodology used to process Natural Language Queries (NLQs), how we tested the system and finally our results from the INEX 2004 Workshop. We conclude with a short discussion on where our system is headed and how the number of participants in the NLP track can increase.

2. MOTIVATION

This section outlines several motivating factors behind the development of a natural language interface for XML-IR systems. These factors are specific to the domain of structured IR, and are therefore closely related to the CAS than CO task. There are also motivating factors that closely related to the CO task, however these have already been covered in The Text Retrieval Conference (TREC) [10] and the Special Interest Group for Information Retrieval (ACM-SIGIR) [9] publications.

The main motivation for an XML IR natural language interface is that formal query languages are too difficult for users to accurately express their information need. A very good example of this occurred at the INEX 2003 Workshop: more than two-thirds of the proposed queries had major semantic or syntactic errors [5]. Furthermore the erroneous queries were difficult to fix, requiring 12 rounds of corrections. 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. However, we feel that users should be able to intuitively express their information need in a natural language.

The second motivation is that users require an intimate knowledge of a document's structure in order to properly express their structural requirements. For instance, if users wish to request elements at the section or paragraph level from the INEX collection, they need to know that those elements correspond to the *sec* and *p* tag respectively. And, 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 the document's DTD or Schema. However, in a natural language interface the underlying document structure can be completely hidden from users, who only require a conceptual document model when formulating queries. So instead of requesting *sec* and *p* tags, users will be able to request sections or paragraphs.

The final motivation is that formal queries do not scale well across multiply or heterogenous collections, even if the collection falls within a single domain. For example, most journal articles conceptually follow the same format, that is, they start with elements such as titles and authors, have a body with sections and paragraphs, and finish with a list of references. So, while these collections are conceptually the same, it is unlikely that they will have the same DTD or Schema. For instance one collection may use the tag *p* to denote paragraphs while another collection may use the tag *para*. A similar situation exists in a heterogenous system where multiple DTDs or Schemas are used in the same collection. However, this problem will be resolved via natural language since as noted in the previous motivation, users will express their information needs conceptually.

3. NLPX SYSTEM

```
<inex_topic topic_id="XX" query_type="CO">
<title>
  "multi layer perceptron" "radial basis
  functions" comparison
</title>
<description>
  The relationship and comparisons between
  radial basis functions and multi layer
  perceptrons
</description>
</inex_topic>
```

Figure 1 A CO Topic

```
<inex_topic topic_id="XX"
query_type="CAS">
<title>
  //article[about(.,information
  retrieval)]//sec[about(.,compression)]
</title>
<description>
  Find sections about compression in
  articles about information retrieval.
</description>
</inex_topic>
```

Figure 2 A CO Topic

Figures 1 and 2 are examples of CO and CAS topics. Both the description and title elements express the users' information need. The description expresses users' need in a natural language (e.g. English). The title expresses users' information need in either a list of keywords/phrases (CO) or as a formal XPath-like language (CAS) called Narrowed Extended XPath I (NEXI) [11].

We developed our natural language interface to accept the description element from an INEX topic. This was the obvious choice, since the description element expresses users' content and structural requirements in natural language, and is meant to be a faithful 1:1 translation of the title element. We had already developed a system for participation in the Ad-hoc track. Therefore, instead of developing a completely new system for participation in the NLP track, we developed a natural language query to NEXI translator. We did this for three main reasons:

- First, by developing a NLQ-to-NEXI translator we were able to use our existing retrieval engine as the backend to system.
- Secondly, since the description element is a 1:1 Translation of the title element, then NLP systems will be able use the existing set of Ad-hoc topics and assessments for testing. Furthermore, future NLP tracks will be able to use the same topics and assessments in future Ad-hoc tracks, resulting in very little extra work for future INEX organisers.
- Finally, we can output the translated queries and compare them with the original NEXI queries to evaluate the successes of the translator.

The syntax of NEXI is similar to XPath, however, it only uses XPath's descendant axis step, and extends XPath by incorporating an 'about' clause to provide an IR-like query. NEXI's syntax is `//A[about(//B,C)]` where **A** is the context path, **B** is the relative path and **C** is the content requirement. Conceptually each 'about' clause in a NEXI query represents an individual information request. So conceptually the query `//A[about(//B,C)]//X[about(//Y,Z)]` contains two requests: `//A[about(//B,C)]` and `//A//X[about(//Y,Z)]`. However, in NEXI only elements matching the leaf (i.e. rightmost) 'about' clause, here the second request, are

returned to the user. We refer to these requests and elements as ‘return requests’ and ‘return elements’. Elements that match the other ‘**about**’ clauses, here the first request, are used to support the return elements in ranking. We refer to these requests and elements as ‘support requests’ and ‘support elements’.

The following subsections describe how a natural language query is first translated to NEXI format and then how a NEXI query is handled by the backend system.

3.1 Natural Language Queries to NEXI Translator

Suppose that the following natural language queries (NLQ) are submitted to the system.

NLQ 1: The relationship and comparisons between radial basis functions and multi layer perceptions

NLQ 2: Find sections about compression in

Figure 3 CO and CAS Natural Language Query

3.1.1 Lexical and Semantic Tagging

Suppose that the contents of Figure 3 are input into the system as natural language queries (NLQ). Translating the NLQs into NEXI format takes several steps. First each word is tagged as either as a special connotation or by its part of speech. Special connotations are words of implied semantic significance within the system. Our system uses three types of special connotations: structural words that indicate the structural requirement of the user (e.g. article, section, paragraph, etc.), boundary words that separate the user’s structural and content requirements (e.g. about, containing) and instruction words that indicate if we have a return or support request. All other words are tagged by their part of speech. In theory, any part of speech tagger could perform this task; however, our system uses the Brill Tagger [1]. The Brill Tagger is a trainable rule-based tagger that has a success rate comparable to state of the art stochastic taggers (>95%). The Brill Tagger defines tags as specified by the Penn Treebank [7]. Figure 4 presents an example of the NLQ after tagging.

NLQ 1: The/DT relationship/NN and/CC comparisons/NNS between/IN radial/JJ basis/NN functions/NNS and/CC multi/NNS layer/NN perceptions/NN

NLQ 2: Find/XIN sections/XST about/XBD compression/NN in/IN articles/XST about/XBD

Figure 4 Tagged CO and CAS Natural Language Query

3.1.2 Template Matching

The second task of the translator is to derive information requests from the tagged NLQ. This is performed by

matching the tagged NLQ to a predefined set of grammar templates. The grammar templates were developed by inspection of previous years’ INEX queries. Initially it may seem that a large number of templates would be required to fully capture the semantics of natural language. However, as natural language queries are written in the same context, comprehending them can be viewed as a subset of classical natural language understanding. Therefore, a system that interprets natural language queries requires fewer rules than a system that attempts to understand natural language in its entirety.

This theory was verified by the inspection of previous INEX queries and recognising that the format of most queries corresponded to a small set of patterns. By extracting these patterns we were able to formulate a small set of grammar templates that match the majority of queries. Figure 5 shows an example of some of the grammar templates.

Query: Request+

Request : CO_Request | CAS_Request

CO_Request: NounPhrase+

CAS_Request: SupportRequest | ReturnRequest

SupportRequest: Structure [Bound] NounPhrase+

ReturnRequest: Instruction Structure [Bound]

Figure 5 Grammar Templates

Conceptually each grammar template corresponds to an individual information request. Once the semantically tagged text was matched to grammar template, we derived information requests from the query. Each information request contains three separate attributes. **Content:** A list of terms or phrases that express the content requirements of the user. This is derived from the noun phrases from the matched grammar template. **Structure:** A logical XPath expression that describes the structural constraints of the request. This value is derived via a function that maps structural words (e.g. section) to the XML tags (e.g. /article/sec) as specified in the document’s DTD. **Instruction:** “R” if we have a return request or “S” if we have a support request. Figure 6 shows an example of the information requests derived from the templates.

3.1.3 Produce NEXI Queries

The final step in the translator is to merge the information request into a single NEXI query. Return requests are output in the form **A[about(.,C)]** where A is the request structural attribute and C is the request content attribute. Support requests are then added to the output in two stages; however, first the system must locate the correct position within the output to place the support request. For example if the internal representation is **X/Y/Z[about(.,C)]** and the support request has a structural attribute of **X/Y/A**, then the structural request should be placed in position Y. Using a string matching function, a comparison is made between the internal

NLQ 1:
Structure: /*
Content: relationship, comparisons, radial basis functions, multi layer perceptions
Instruction: R

NLQ 2:
Request 1
Structural: /article/sec
Content: compression
Instruction: R
Request 2
Structural: /articlec
Content: information retrieval

Figure 6 A Derived Information Requests

representation and the support request structural attribute, to determine the correct position of the support request. Then the request is added to the internal representation in the form A[about(B,C)] where A is the longest matching string, B is the remainder of the support request structural attribute and C is the support requests content attribute.

Figure 7 is how the NEXI queries would appear after the information requests for each NLQ have been merged.

NLQ 1:
 /*[about(.,relationship, comparisons, radial basis functions, multi layer perceptions)]

NLQ 2:
 //article[about(.,information retrieval)]//sec[about(.,compression)]

Figure 7 NLQ-to-NEXI Queries

3.2 GP-XOR Backend

3.2.1 NEXI Interface

Once NEXI queries are input into the system they are converted into an intermediate language called the RS query language. The RS query language converts NEXI queries to a set of information requests. The format of RS queries is

Request: Instruction 'I' Retrieve_Filter 'I' Search_Filter 'I' Content.

The Instruction and Content attributes are the same as they were in the previous section; however, the Structural attribute has been divided into a Retrieve and Search Filter. While both are logical XPath expressions the **Retrieve Filter** describes which elements should be retrieved by the system, while, the **Search Filter** describes which elements should be searched by the system. Figure 8 presents an example of the queries introduced earlier converted to RS queries. For the NEXI query, //A[about(//B,C)] the retrieve filter is its context path //A while its search filter is its relative path, is //A//B.

RS Query 1:
 R/*/*/*| relationship, comparisons, radial basis functions, multi layer perceptions
RS Query 2:
 R//article//sec//article//sec|compression
 S//article//article|information retrieval

Figure 8 Example of an RS Query

3.2.2 System Structure and Ranking Scheme

We index the XML collection using an inverted list. Given a query term we can derive the filename, physical XPath and the ordinal position within the XPath that it occurred in. From there we construct a partial XML tree containing every relevant leaf element for each document that contains a query term. Further information on our structure can be found in [4].

Elements are ranked according to their relevance. Data in an XML tree is mostly stored in leaf elements. So first we calculate the score of relevant leaf elements, then, we propagate their scores to their ancestor branch elements.

The relevance score of leaf elements is computed from term frequencies within the leaf elements normalised by their global collection frequency. The scoring scheme rewards elements with more query terms. However, it penalises elements with frequently occurring query terms, and rewards elements that contain more distinct query terms.

The relevance score of a non-leaf is the sum of the children scores. However, leaf element scores are moderated by a slight decay factor as they propagate up the tree. Branch elements with multiple relevant children are likely to be ranked higher than their descendents – as they are more comprehensive – while branch elements with a single relevant child will be ranked lower than the child element as they are less specific.

4.0 TESTING

4.1 Testing Methodology

Our experiments were conducted using the INEX 2003 set of topics and evaluation metrics. The results from 2003 INEX queries were submitted into the official INEX evaluation program that calculated the recall/precision graphs. As the results comprised of XML elements, the precision value was calculated over two dimensions: exhaustiveness, which measures the extent to which a component discusses the information request; and specificity, which measures the extent to which a component is focused on information request.

We aimed to execute two runs for both the CO and CAS topic sets. The first run was to accept NEXI queries, that is topic's title tag, as input while the second run was to accept natural language queries, that is topic's description tag, as input. These runs corresponded with INEX's Ad-hoc and NLP tracks, and allowed us to compare how well our system performed with and without our NLP-to-

NEXI frontend. It was hoped that the results of these two runs would be fairly similar, however, we identified one major obstacle that hindered our progress.

4.1 Testing Obstacles

INEX guidelines specifically state that the description tag should be as close as possible to a 1:1 natural language translation of the title. However, during the execution of this run, it became clear that many of the descriptions are not faithful 1:1 title translations. Therefore it would be impossible for a system with a NEXI interface to produce similar results as a system with a NLQ, even if they used the same backend. From our observations we have identified three major discrepancies between the title and description tags: inconsistent content requirements, inconsistent structural requirements, and vague structural requirements.

4.1.1 Inconsistent Content Requirements

The first discrepancy is when the title and description contain different content requirements. An example of this occurs in topic 93, where the title mentions the term ‘institute’ while the description does not.

```
<title>
  "Charles Babbage" -institute -inst.
</title>
<description>
  The life and work of Charles Babbage.
</description>
```

Figure 9 Topic 93

Naturally, participants in the NLP track would be greatly concerned when the title contains more information than the description. However, Ad-hoc participants should also be concerned about this discrepancy, since as topic 104 shows, sometimes the description can contain more content information than the title.

```
<title>
  Toy Story
</title>
<description>
  Find information on the making of the
  animated movie Toy Story, discussing
  the used techniques, software, or
  hardware platforms.
</description>
```

Figure 10 Topic 104

4.1.2 Inconsistent Structural Requirements

A second discrepancy is when the title and description contain different structural requirements. For example in topic 76 the title element specifically requests the retrieval of section elements, however no mention of this is made in the description element.

```
<title>
  //article[(./fm//yr = '2000' OR
  ./fm//yr = '1999') AND about(.,
  "intelligent transportation
  system")] //sec[about(., 'automation
  +vehicle')]
</title>
<description>
  Automated vehicle applications in
  articles from 1999 or 2000 about
  intelligent transportation systems.
</description>
```

Figure 11 Topic 76

Likewise the title element of topic 81 specially requests its content requirements occur at the paragraph level, but no mention of this is made in the description. Nor is any mention made in the title to bibliographic references, as it is in the description element.

```
<title>
  //article[about(./p, "multi
  concurrency control") AND
  about(./p, 'algorithm') AND
  about(./fm//at1, 'databases')]
</title>
<description>
  We are interested in articles that can
  provide information or reference to
  information on algorithms for
  multiversion concurrency control in
  databases.
</description>
```

Figure 12 Topic 81

4.1.3 Vague Structural Requirements

The final discrepancy is when the description element contains information that is too vague for an NLP system to understand. A perfect example occurred in topic 89 where the description asks for articles “from recent years”. We can not expect an NLP system to understand such vague concepts as “recent years”, and the description should be rewritten to quantify the year of publication. Likewise in topic 85 the description tag asks for articles “with many figures”, but at least they also quantify the number of figures as “at least 10”.

```
<title>
  //article[about(./bdy, 'clustering
  "vector quantization" +fuzzy
  +k-means +c-means -SOFM -
  SOM')] //bm//bb[about(., "vector
  quantization" +fuzzy clustering +k-
  means +c-means') AND
  about(./pdt, '1999') AND ./au/snm !=
  'kohonen']
</title>
<description>
  find articles about vector
  quantization or clustering and return
  bibliography details of cited
  publications about clustering and
  vector quantization methods, from
  recent years, not authored by Kohonen.
</description>
```

Figure 13 Topic 89

```

<title>
  //article[./fm//yr >= 1998 and
  .//fig//no > 9]//sec[about(./p,'VR
  "virtual reality" "virtual
  environment" cyberspace "augmented
  reality"')]
</title>
<description>
  Find sections about Virtual Reality.
  Retrieve sections only from articles
  that were published in or after the
  year 1998 and have many figures (at
  least 10).
</description>

```

Figure 14 Topic 85

To overcome these obstacles we modified the description elements so that they were a faithful 1:1 translation of the title elements. It must be stressed that in this modification to descriptions we were very careful not to generate descriptions that favour the natural language processing that we implemented. Modifications mostly ensured that the same keywords and structural tag names that appeared in the NEXI formulation also appeared in the natural language description.

4.3 Testing Results

Three runs were submitted for both the CO and CAS topics sets: one with NEXI queries, with the original NLQ description elements, and one with the altered NLQ description elements. The plots for these three runs are displayed in Figure 15-18. A fourth plot is the recall/precision line of the University of Amsterdam's systems that achieved the best results at INEX 2003 in the CO and SCAS tracks [8]. This allowed us to compare the quality of our system with the best official INEX alternative. Two metrics were used to formulate the recall-precision values, the strict metric that evaluates highly relevant and highly precise results, and the generalized metric that evaluates results based on a graded measure (or degree) of relevancy and/or precision. Further details on the metrics are available in [3].

The plots for each task and quantization are listed in figures 15-18. The solid line is the Amsterdam submission, the dotted line is the Ad-hoc submission, the dashed line is the NLP submission and the dash-dotted line is the altered-NLP submission.



Figure 15 The INEX 2003 SCAS Strict R/P Curve

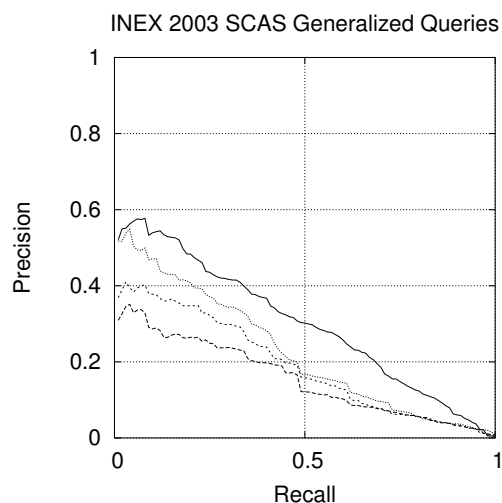


Figure 16 The INEX 2003 SCAS Generalized R/P Curve



Figure 17 The INEX 2003 CO Strict R/P Curve

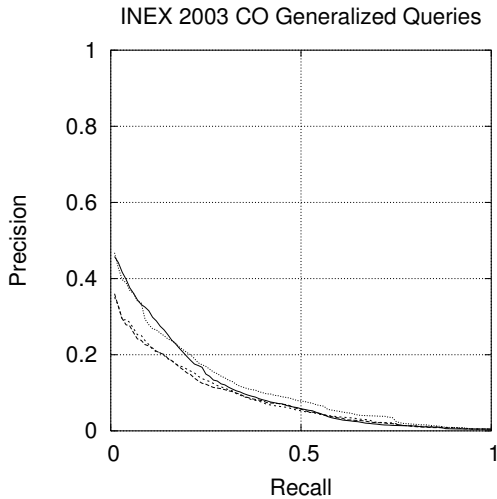


Figure 18 The INEX 2003 CO Generalized R/P Curve

5.0 RESULTS

The system was entered into both the Ad-hoc and NLP tracks at INEX2004. In the Ad-hoc track the system ranked 1st from 52 submitted runs in the VCAS task, and 6th from 70 submitted runs in the CO task. In the NLP track the system was ranked 1st in the VCAS task and 2nd in the CO task. While the NLP track was limited to 9 participants initially, of which only 4 made official submissions, the most encouraging outcome was that the NLP system outperformed several Ad-Hoc systems. In fact, if the NLP submission was entered in the Ad-hoc track it would have ranked 12th from 52 in VCAS and 13th from 70 in CO. This seems to suggest that in structured IR, natural language queries have the potential to be a viable alternative, albeit not as precise as a formal query language such as NEXI.

The Recall/Precision Curves for the Ad-hoc track, along with the R/P curve for our NLP runs are presented in Figures 19 and 20. The top bold curve is the Ad-hoc curve, the lower is the NLP curve, and the background curves are of all the official Ad-hoc runs at INEX 2004.

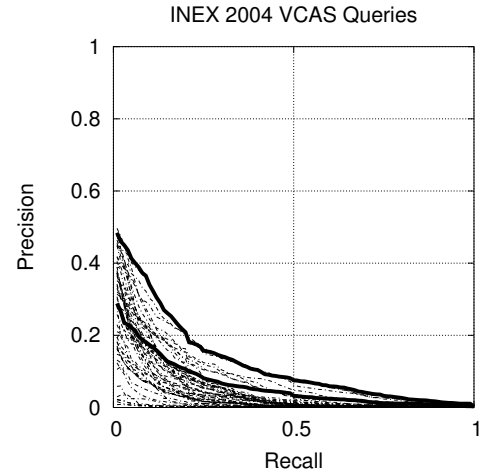


Figure 19 The INEX 2004 VCAS R/P Curve

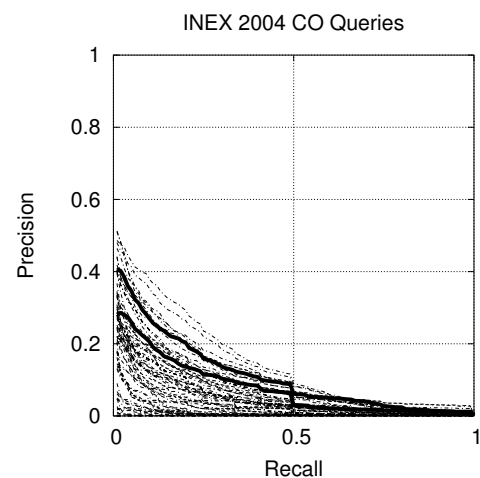


Figure 20 The 2004 INEX CO R/P Curve.

6.0 FUTURE OUTLOOK

The most promising aspect of our participation in this year's NLP track was that our system was able to outperform the majority of Ad-hoc systems, thereby verifying that natural language queries have the potential to be a viable alternative to complex formal queries languages such as NEXI. However, there is still progress to be made, and we shall strive to improve the performance of our NLP system.

However, the most disappointing aspect of this years track was the lack of submissions. INEX should encourage more participation in the NLP track in order to broaden the knowledge in the domain of natural language interfaces to XML IR and to strengthen INEX as a whole. Future participants will most likely come from two areas. The first will be existing INEX participants in the Ad-hoc and other tracks who will develop a natural language interface to their existing system, similar to the approach we took. The second will be from participants in workshops such as TREC [10] and SIGIR [9], which are traditionally strong in the domain of natural language IR systems. Both types of competitors are likely to bring

different perspectives on the problem and their participation should be welcomed.

We also feel that the INEX organiser should strive to ensure that the title and description elements are faithful 1:1 translations, both in terms of structural and content requirements.

7.0 CONCLUSION

This paper presents a natural language interface to XML-IR system. The interface uses template matching to derive users' content and structural requests from natural language query. The interface then translates the NLQ to a formal query language, allowing the request to be processed by many existing systems. Our backend system responds to user queries with relevant and appropriately sized results in a timely manner and our ranking scheme is comparable with the INEX best alternatives. While the NLP interface requires further development, however, initial results are promising.

BIBLIOGRAPHY

- [1] E. Brill. A Simple Rule-Based Part of Speech Tagger. In *Proceedings of the Third Conference on Applied Computational Linguistics (ACL)*, Trento, Italy, 1992.
- [2] Clark, J. and DeRose, S., "XML Path Language XPath version 1.0.", Technical report, W3C, 1999. W3C Recommendation available at <http://www.w3.org/TR/xpath>.
- [3] N. Fuhr and S. Malik. Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2003. In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17, 2003, pages 1-11. 2004.
- [4] S. Geva and M. Spork. XPath Inverted File for Information Retrieval, In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17, 2003, pages 110-117. 2004.
- [5] R. O'Keefe and A. Trotman, "The Simplest Query Language That Could Possibly Work", In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17, 2003, pages 19-26. 2004.
- [6] C. D. Manning. and D. Schutze "Foundations of Statistical Natural Language Processing", MIT Press, Cambridge, 1999.
- [7] M. Marcus, B. Santorini. and M. Marcinkiewicz, Building a large annotated corpus of English: The Penn Treebank, In. *Computational Linguistics*, 1993.
- [8] B. Sigurbjornsson, J. Kamps, M. de Rijke, *An Element-based Approach to XML Retrieval*, In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17, 2003, pages 19-26. 2004.
- [9] Special Interest Group on Information Retrieval (SIGIR) Homepage, <http://www.acm.org/sigir/>.
- [10] Text REtrieval Conference (TREC) Homepage, <http://trec.nist.gov/>.
- [11] A. Trotman and B. Sigurbjörnsson, *Narrowed Extended XPath I (NEXI)*, <http://www.cs.otago.ac.nz/postgrads/andrew/2004-4.pdf>, 2004.
- [12] R. J. Van Rijsbergen, R. J., *Information Retrieval*, Butterworths, Second Edition, 1979.

Analysing Natural Language Queries at INEX 2004

Xavier Tannier
École Nationale Supérieure
des Mines
158 Cours Fauriel
F-42023 Saint-Etienne, France
tannier@emse.fr

Jean-Jacques Girardot
École Nationale Supérieure
des Mines
158 Cours Fauriel
F-42023 Saint-Etienne, France
girardot@emse.fr

Mihaela Mathieu
École Nationale Supérieure
des Mines
158 Cours Fauriel
F-42023 Saint-Etienne, France
mathieu@emse.fr

ABSTRACT

This article presents the contribution of the “École Nationale Supérieure des Mines de Saint-Etienne (France)” to the new Natural Language Processing special Track of the third Initiative for Evaluation of XML Retrieval (INEX 2004). It discusses the place of NLP in XML retrieval and presents a method to analyse natural language queries.

1. INTRODUCTION

If the eXtended Markup Language (XML) becomes – as expected – a universally accepted standard for exchange of information, querying these structured documents in natural language rather than in a structured query language will soon turn into a necessity.

The aim of the new INEX NLPX Track (Natural Language Processing for XML Information Retrieval) is to promote “interaction among researchers in the field of Natural Language Processing (NLP) and XML Information Retrieval”. Our participation to this track lies within this scope: the purpose is to add our contribution to the general reflection about the applications of NLP methods to XML retrieval. Our objective at INEX 2004 is clearly not (yet) the demonstration of the retrieval effectiveness of a system, but the implementation of a technique for analysing a natural language query.

This article considers the benefits that can be gained from using some natural language processing methods on one hand, and the specificities of structured documents on the other hand, in order to retrieve information from an XML corpus. It also presents and discusses our method that relies on the structure of the document to “understand” the semantics of the request.

2. HOW CAN NLP HELP?

The applications of Natural Language Processing for Information Retrieval have been extensively studied in the case of textual (flat) collections (for overviews on this subject, see [9, 4, 10, 1, 5]). Linguistic analyses of the corpus and/or the query should carry out some decisive improvements in the retrieval process. Nevertheless, only a few linguistic methods, as phrasal term extraction or some kinds of query expansion, are now commonly used in information retrieval systems. At present, actual results are not yet up to what we could expect [13, 5].

However we think that the spread of structured corpora can bring new hopes to NLP supporters, at least for the two

following reasons:

- The benefit that can be gained from allowing requests in natural language is probably much higher in XML retrieval than in traditional IR. In the last case, a query is generally a keyword list which is quite easy to write. In XML retrieval such a list is not enough to make queries on both content and structure; for this reason, advanced structured query languages have been devised.

But one wants XML to be really widely used, and that implies that novice and casual users should be able to make requests on any XML corpus. In this perspective, two major difficulties arise, because we cannot expect such users to:

- learn a complex structured and formal query language¹;
- have a full knowledge of the DTD and its semantics.

Note that these issues already exist in the domain of databases with the Structured Query Language (SQL); but unlike databases, the XML format looks set to become used by the general public, notably through the Internet. That is why, although unambiguously machine-readable, structured and formal query languages are necessary (in order to actually extract the answers), the need for simpler interfaces will become more and more important in the future.

- In order to perform a really effective natural language-based retrieval in a flat document, a system should “understand” the semantics of the text, and this is not feasible yet. In the case of structured documents, a well-thought and semantically strong structure, because it formally marks up the meaning of the text, can make easier the query “understanding”, at least when this query refers (partly) to the structure (the VCAS task in INEX).

However, this requires a certain amount of knowledge about the corpus, that has to be integrated into a system besides the documents themselves, in order to perform a good retrieval process. We discuss this subject in section 5.4.

¹In this paper we call “*formal query language*” a language with formalized semantics and grammar, as opposed to natural language.

3. DESCRIPTION OF OUR APPROACH

Our aim is to generate a query in a formal structured language from the <description> part of INEX topics, which is written in natural English. The topics are divided into two categories:

- *Content-and-Structure* queries, which contains structural constraints.
e.g.: *Find paragraphs or figure-captions containing the definition of Godel, Lukasiewicz or other fuzzy-logic implications.* (Query 127, INEX 2004)
- *Content-only* queries that ignore the document structure.
e.g.: *Any type of coding algorithm for text and index compression.* (Query 162, INEX 2004)

To achieve the analysis of such requests, the steps that we perform are:

- a part-of-speech tagging of the query (3.1);
- a syntactic/semantic analysis of the query (3.2);
- with the help of specific rules (3.3):
 - 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 the semantic elements mapping on the structure and, respectively, mapping on the content;
- a treatment of relations existing between different elements recognized at last stage (3.4);
- the construction of a formal language query (3.5).

3.1 Part-of-speech tagging

A part-of-speech (POS), or word class, is the role played by a word in the sentence (e.g.: noun, verb, adjective...). POS tagging is the process of marking up the words in a text with their corresponding roles. To carry out this task we chose the free tool TreeTagger [8, 7]. For example, for the following query:

(1) *Find the title of articles that deal with semantics.*

... the output of TreeTagger is given in figure 1².

3.2 Syntactic/semantic analysis

This analysis is performed with a set of rules describing the grammatical constructions that are the most current in queries and questions. These rules are said *context-free*: they define which sequence of elements (on the right side) is needed to compose a single new element (on the left side). The whole set of rules is a *context-free grammar (CFG)*. As an example, we listed in figure 2 the rules that are triggered when parsing our sample query (1).

²For a clearer comprehension by a non-expert reader, we changed the names of the tags into less complete but more explicit abbreviations.

Find	VERB(IMP)	find
the	DET	the
title	NOUN	title
of	PREP	of
articles	NOUN(PLUR)	article
that	REL_PRO	that
deal	VERB	deal
with	PREP	with
semantics	NOUN	semantics

Figure 1: POS tagging of sentence (1) with TreeTagger². *Find* is an imperative verb, *of* and *with* are prepositions and *that* is a relative pronoun

NP	→	DET? NOUN
NP	→	NP PREP NP
NP	→	NP REL_PROP
REL_PROP	→	REL_PRO VP
VP	→	VERB PREP? NP
S	→	VERB(IMP) NP

Figure 2: Examples of CFG rules, with S = Sentence, NP = Noun Phrase, REL_PROP = relative proposition, VP = Verbal Phrase. The question mark “?” means that the element is optional in the sequence.

A recursive application of these CFG rules results in the *syntactic tree* represented in figure 3³.

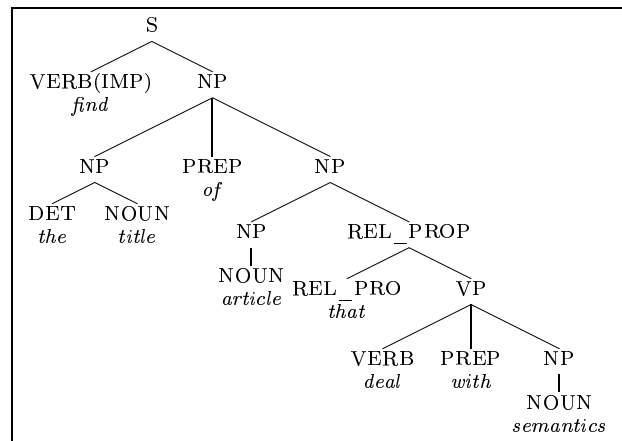


Figure 3: Syntactic tree of sentence (1), obtained with rules of figure 2

This operation gives us a syntactic structure, but we need some semantics to have an idea about the relations existing between the words. In that aim, we use a very simple implementation of Discourse Representation Theory (DRT) [6]

³Note that with this set of rules two different parsing are possible: the relative proposition can be attached to the noun “article” (as shown in the figure) or to the noun “title”. In practice both trees are explored.

(for an overview, see [2], volume II). In DRT, the semantic representation of a discourse (or a part of discourse) is described with a two-level “box” called “Discourse Representation Structure” (DRS). The upper level gives the discourse referents, which are the elements introduced by the discourse; the lower level represents the conditions concerning the referents.

Figure 4 shows a typical example of DRS.

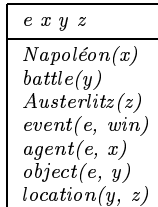


Figure 4: Semantic representation in DRT of the sentence: “Napoleon wins a battle in Austerlitz”

In this example, the terms “Napoléon”, “Austerlitz”, “battle” and the verb “to win” (event e) are discourse referents, represented by letters in the upper level and described by logical predicates in the lower level. The other conditions are about the agent and object of the event (respectively “Napoléon” and “battle” for the event “to win”) and the location of the battle.

One can note that the semantics of some predicates can differ from a domain to another. If the *location* is here understood as a geographic feature, in an XML retrieval context it would probably be a structural constraint. A task-specific choice has to be made in order to model constructions that are peculiar to XML retrieval.

To compute a DRS representing a whole sentence, we attribute a basic DRS to each word, depending on its class (POS). Let us come back to our running example of sentence (1). Figure 5 contains three examples of POS DRSs for a noun, a verb and a preposition.

The syntactic rules are enriched with semantic actions. In our example, with the rule:

$VP \rightarrow VERB\ PREP? NP$,

applied to the DRSs of figure 5, we add the following identity conditions: $e_1 = e_2$ and $x = y$. The result is the semantic tree of figure 6.

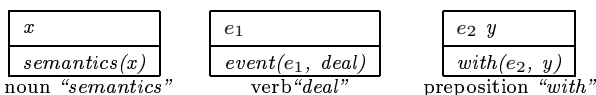


Figure 5: examples of word DRSs

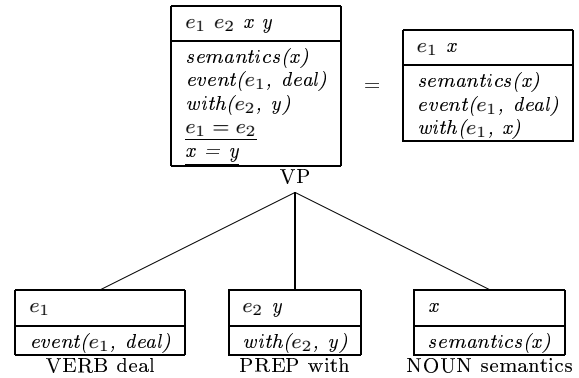


Figure 6: example of DRS for the verbal phrase “deal with semantics”

Due to a lack of space we cannot show the full semantic tree obtained for the example. The figure 7 gives the final DRS. Note that a referent has been added, which is implicit in the sentence: the *interlocutor* to which we give an order when using a verb in the imperative mood (here, “find...”).

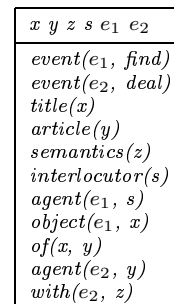


Figure 7: DRS for sentence (1)

N.B.: The set of syntactic/semantic rules that we use is made up of about 50 rules and is obviously not intended to describe the whole language. The stress has been put on noun phrases, that are often much more meaningful than verbal phrases (at least in terms of Information Retrieval). Relative propositions, prepositional phrases are also very important because they mark a structure of query that we do not want to miss. For complex demands, an entire parsing is often impossible. In that case only the noun phrases are analyzed and the verbs are left out.

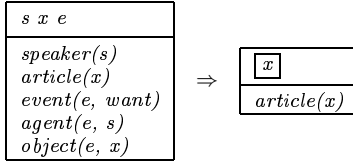
The DRS we obtain at this stage cannot be used to build a formal query yet. Some IR-specific rules have to be set up.

3.3 Specific rules

The semantic construction can be reduced by taking some special cases into account, among which:

1. The **“query verbs”** like *“to want”*, *“to find”*... With the help of a dictionary describing the semantic relation between those verbs and the queries, we set a particular flag on the concerned element. This flag means that this element should be selected as a good response to the query, as shown in the following example.

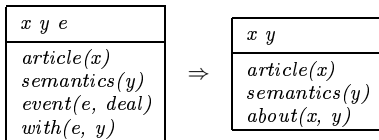
(2) I **want** an article.



Here we know that the verb *“to want”* means that its object (*“article”*) has to be selected, and this new information is represented by a framed referent. The agents of such verbs (here the *speaker*, or *“I”*), as well as the verbs themselves, are then left out.

2. The **description verbs** like *“to deal with”*, *“to concern”*... An other dictionary contains the information that allows us to add a new relation called *about*:

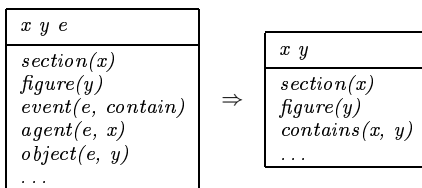
(3) an article that **deals with** semantics.



The verb referent is removed as well in this case.

3. The **verbs of topological relation** like *“to contain”*, *“to include”*... If such a verb has an agent and an object, then an appropriate relation is set up between those two elements and the verb is deleted:

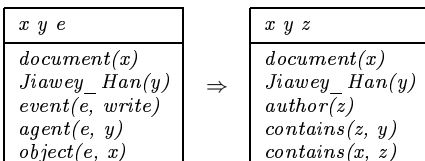
(4) a section that **contains** a figure...



4. Some corpus-specific **semantic rules** that have to be added in order to recognize some precise linguistic constructions.

(5) a document written by Jiawei Han.⁴

Here an *ad-hoc* rule imposes the following transformation:



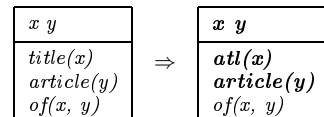
N.B.: This kind of rules no longer represents *linguistic* features, but IR-specific rules. The two *contains* predicates in this example do not have a real linguistic meaning, but express a structural constraint in the XML document.

5. The **words or phrases in quotation marks** are considered as non-separable expressions and are grouped together in a single variable.

(6) We are looking for sections in articles, whose abstracts contain "spatial join," that describe "performance evaluation." (*Topic 156*)

6. And above all, a **term recognized as a DTD-tag (or synonym)** is changed into this tag name and is marked as such (here by a bold predicate in the DRS, with *atl* standing for *“title”*).

(7) the **title** of an **article**...



A dictionary of synonyms is used. This dictionary is absolutely not intended to have a general purpose; it is corpus-specific. Indeed the DTD tag names are rarely real words, but abbreviations instead (*st* for *section title*, *p* for *paragraph* etc.).

Figure 8 shows the application of some of these specific rules on our sample DRS. Let us remind the initial request:

- (1) *Find the title of articles that deal with semantics.*

We suppose that our dictionary tells us that the words *“title”* and *“article”* respectively stand for the tag names *atl* and *article*.

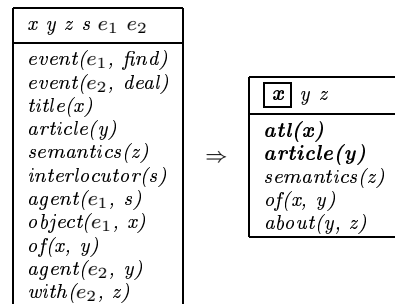


Figure 8: DRS for sentence (1) before and after application of specific rules. Let us remind that we are looking for “a title of article that deals with semantics”. The referent *x* is selected (rule 1), the article *y* is *about* “*semantics*” (rule 2) and the terms “*article*” and “*title*” is recognized as tag identifiers *atl* and *article* (rule 6).

⁴Inspired by topic 131, INEX 2004

In this new DRS, we can clearly distinguish a *tag name*, which is related to the structure of the document (in bold type), from what we will now call a *term*, as “*semantics*”, which is supposed to be a part of the textual contents of the document.

3.4 Structure analysis

At this stage we still have some binary relations between referents that have not been treated by any specific rule (in figure 8 the relation $of(x, y)$). These relations have all a particular meaning, and a system cannot have the knowledge of each of these meanings. We only implemented a semantic representation of some important relations (and particularly “temporal” relations – *after*, *included*, etc., that should be understood here as order constraints in the XML file).

Let $R(x, y)$ be a binary relation between referents x and y . To handle “known” relations as well as “unknown” ones, we apply a heuristic according to the following cases:

1. x and y refer to two tag names (representing structural elements):

- (a) if the relation R is known, no action is needed.

e.g.: A **paragraph** after a **figure**:

x y
$p(x)$ $fig(y)$ $after(x, y)$

- (b) if the relation R is unknown, the fact that a relation exists is in itself an information: the structure given by the DTD allows then to guess which relation(s) it can be. In our example the DTD will tell us that an element *atl* (title x) is *contained* by an element *article* (y).

e.g.: A **title** of **article**:

x y
$atl(x)$ $article(y)$ $of(x, y)$

 $\xrightarrow[\text{analysis}]{DTD}$

x y
$atl(x)$ $article(y)$ $contains(y, x)$

2. The relation links a tag (let us say x) and a term (y):

- (a) if R is known, we add a tag that can match with any name (called ‘*’). This tag is *about* y , and the relation is transferred to this new tag:

e.g.: A **paragraph** before the *conclusion*⁵:

x y
$p(x)$ $conclusion(y)$ $before(x, y)$

 \rightarrow

x y z
$p(x)$ $conclusion(y)$ *(z) $about(z, y)$ $before(x, z)$

The paragraph should be written in the XML file before a tag that contains the word “*conclusion*”.

- (b) if R is unknown, our first experiments show that a treatment produces more noise than useful information. The relation is suppressed⁶.

e.g.: The topic is to find *XML* **articles**⁷:

⁵We suppose in this example that the word “*conclusion*” cannot be assimilated to a tag name, as it is the case in IEEE collection.

⁶Here the referent y does not have any more relation with other elements and becomes useless.

⁷Inspired by topic 140, INEX 2004

x y
$article(x)$ $XML(y)$ $rel_adjective(x, y)$

 $\xrightarrow{\text{remove}}$

x y
$article(x)$ $XML(y)$

N.B.: For this rule we can imagine that a deeper semantic knowledge could be helpful. For example, if one asks for a *short* article, a description of what is considered to be a *short* document would allow to get more appropriate answers. But this kind of knowledge is very hard to model because of the number and the subjectivity of the relations involved.

3. R holds between two terms (*term* is here used as opposed to *DTD tag names*): in that case the relation does not apply to the structure, as x and y refer to content elements. We do not have any particular treatment to do, but the relation can be a useful linguistic information that we keep for the retrieval process (which can use it or not).

e.g.: The structural similarity between labeled trees⁸:

w x y z
$similarity(w)$ $tree(x)$ $structural(y)$ $labeled(z)$ $rel_adjective(w, y)$ $rel_adjective(x, z)$ $between(w, x)$

If the search engine is able to handle such relations (*adjectives* and *between*), it is useful to know that, for example, the whole phrase “*similarity between trees*” is preferable to the separate words “*similarity*” and “*trees*”.

Among these rules, only the rule 1b applies in our running example, and our final DRS is shown in figure 9.

x y z
$atl(x)$ $article(y)$ $semantics(z)$ $contains(x, y)$ $about(y, z)$

Figure 9: final DRS for sentence (1).

3.5 Formal language query

At the end of the linguistic phase, our aim is to obtain a formal language query that could easily be translated into an existing structured language. From those languages (and initially from SQL) we take the idea of clause pattern (SELECT-FROM-WHERE in SQL) for restructuring the request. We chose the following four-clause pattern (expressed in an XML syntax):

- the **from** clause contains the tag names and indications on the tag path (XPath[14] expressions);

⁸Inspired by topic 167, INEX 2004

- in the **select** clause we find the elements that are to be returned to the user; those elements must be referred in the **from** clause;
- the **where** clause contains the relations between tags or variables (*e.g.*: before(x, y), about(a, b));
- the **variables** are identifiers replacing terms in the other clauses.

N.B.: The **select** clause only contains element names, we do not want to provide (as in XQuery-like [15] query languages) any possibility of formatting the output. Besides this is neither the purpose of INEX nor of IR in general.

The transformation process from DRS to formal language is straightforward: the tag names are already flagged (in bold type in our representation), as well as the selected elements (framed referents in the DRS). The variables are the unary predicates that are not tag names, and the **where** clause corresponds to the other conditions.

In this manner, our system automatically generates a query in XML. Not surprisingly this form is quite verbose and hard to read, and we rather chose to show a SQL-like representation of the DRS of figure 9 that we can obtain with a simple XSL Transformation [16] (figure 10).

```
FROM y = /article,
      x = y//atl,
WHERE about(y, z)
VARIABLES z = "semantics"
SELECT x
```

Figure 10: Example of formal query obtained from DRS 9 (query (1)). *y* is an *article* tag, *x* is an *atl* tag (title) contained in *y*, and *z* is a variable representing the term “*semantics*”.

4. THE RETRIEVAL PROCESS

At this stage we have a formal query that can be translated into an existing and implemented language. But while doing that we should keep in mind that the Information Retrieval is yet to be done after the linguistic process, and a database-oriented (XQuery-like) query language does not suffice:

- In our example the *about* relation that is obtained from query (1) is to be implemented in one way or another.
- It would be great if the search engine could “understand” the linguistic constraints that we generated (see sections 3.4 and 6).
- As the request is a natural language query, the **from** clause should not necessarily be considered as strict paths. If the user’s request is “*give me a paragraph about semantics*”, a section or a figure could also be relevant (maybe not even *less* relevant). This remark corresponds to the INEX choice of assessing “*Vague Content and Structure*” (*VCAS*) queries rather than “*Strict Content and Structure*” (*SCAS*) queries.

For our participation to INEX, we chose to develop our own XML retrieval system, to make easier the connection of our

query analysis system into the retrieval module. In view of the disappointing results of this system, it proved to be an unfortunate choice. The serious lack of time before INEX deadline that we have been confronted with is one of the reasons for that; but anyway, adapting our linguistic module to an existing system or (even better) integrating it into a system developed in collaboration with a better skilled team would be a more judicious solution in the future.

5. COMMENTS

A reflection about our work for INEX gives rise to some comments about specificities and limits of our approach.

5.1 CAS versus CO topics

The strongest originality of our method is that it relies on the structure of the document in order to analyse the semantics of the request. For this reason it was predictable that results for *Content-and-Structure* task would be better than for *Content-Only* task. That is what happened, while we are ranked 3rd for *VCAS* topics and 4th for *CO* topics in *NLPX* track, and 36th for *VCAS* and 51st for *CO* in the *ad hoc* track. If a *SCAS* track had existed we could have expected better results.

5.2 Structural density

A comparison between our work on IEEE collection for INEX and our other studies can give us some indications about the relation between the density of mark-up in the corpus and the ability to analyse properly the requests on this corpus. While we are using the structure to perform the analysis, it makes sense to consider that the higher this density is, the easier our work should be. And this is right to a certain extent. But if a corpus is more data-centric, closer to a database format (as address books or flight schedules), then the users’ requests on this corpus are very strict and precise (puch more than INEX topics). Then the limits of NLP (so far) are reached, and any imperfection in the system or ambiguity could lead to an erroneous interpretation. We need in such cases to consider a “restricted” natural language, limited to unambiguous structures and pre-defined terms. As a matter of fact, The INEX collection seems to be an “ideal” in-between format to apply our method.

5.3 Topic complexity

The topics proposed by participants at INEX 2004 are very diverse. The **description** part varies from 5 to 76 words:

- 185 Find articles about gesture recognitions.
- 201 ranked retrieval in the WWW.
- 187 We are looking for articles containing description of dimension reduction methods. These methods allow us to lessen effects of the "curse of dimensionality" and make retrieval of documents faster. Examples of this methods are well-known latent semantic indexing (LSI) which improves recall of the retrieval system and random projection which does not modify distances too much. We are not interested in stoplist filtration which does reduce the dimension a bit as a byproduct of term removal.

The main problem is not so much the length of the request as the fact that the language changes when the request is long. Indeed the topics 185 and 201 are typical IR requests:

a “Find + Noun Phrase” construction and a single noun phrase. On the other hand, the last example consists of sentences of common language, containing many anaphoras and pragmatic insinuations; a proper analysis of all the niceties of language is not conceivable. This leads to a lot of noise in the results; for instance the words “effects” or “projection” will be considered by our system as terms to find in the documents as well as the others. Even if they are related to the topic their importance is quite lower. Moreover our system does not handle negations yet.

5.4 Corpus knowledge

During the stages that we detailed in the previous sections, we used several kinds of information about the explored corpus. This knowledge has to be modeled for each new set of documents⁹ to study. Even if our track is not concerned by heterogeneous documents, we tried to reduce this necessity to the minimum. But despite this, we think that the following points are the minimum knowledge to handle in order to perform an appropriate query analysis:

- The DTD
- Because the DTD tag names are not “real” words (see section 3.3) and also because several words can be used for a single concept, we need a dictionary of acceptable synonyms for the tag names (*e.g.*: paper = article = document, title = atl, etc.);
- Semantic locutions (*e.g.*: “a list of keywords” = “keywords”), in order to avoid noise generated by an erroneous detection of terms (here, *list* is neither a tag name nor a term to look for in the text);
- Some very simple ontologic structures (*e.g.*: “a novel written by Marcel Proust” = “a novel of which the author is Marcel Proust” – if we suppose that *novel* and *author* represent tag names);

6. EXAMPLES

We give here three significant examples of topic analyses. We already explained that several syntactic parsings could be possible for the same sentence. In practice a “score” is attributed to each rule release, depending on several parameters (among which the distance between the words that are linked, the length of the phrases, the type of the relations. . . Unfortunately we lack space to explain more precisely this process). In our sample topics only the best scored result is given.

6.1 Topic 127

127 Find paragraphs or figure captions containing the definition of Godel, Lukasiewicz or other fuzzy-logic implications.

The first DRS obtained before the application of IR rules is shown in figure 11. The predicate “rel_adjective(x, text)” means that the word *text* qualifies the variable x. The relation “rel_nmodifier(x, y)” stands for nominal compounds (here, “fuzzy-logic implication” and “figure caption”).

⁹We call a *new* set of documents a corpus with a different DTD and different subject.

The final query is given in figure 12. The relations beginning by “ling_” are linguistic relations that it would be interesting (but not essential) to treat (in order to take into account possible variations in multi-word terms [3, 11]).

```

c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13
event(c1, sym:find, object:c12, agent:c13)
event(c2, sym:contain, object:c11, agent:c12)
of(c3, c9 & c10)
definition(c3)
c9 = godel
c10 = lukasiewicz
'fuzzy-logic'(c4)
implication(c5)
rel_nmodifier(c5, c4)
rel_adjective(c5, other)
c11 = c3 ∨ c5
paragraph(c6)
figure(c7)
caption(c8)
id_nmodifier(c8, c7)
c12 = c6 ∨ c8
c13 = id_interlocutor

```

Figure 11: DRS for topic 127

```

from      v2 = *
          v3 = *
          c12 = *
          v4 = *
          c6 = v4//p
          v5 = *
          c8 = v5//fgc
where     c12 = v4 or v5
          c12 = v2 or v3
          ling_adj(c5, v1)
          ling_np_relation(c3, c10, of)
          ling_np_relation(c3, c9, of)
          ling_np_relation(c5, c4, nmodifier)
          about(v3, c5)
          about(v2, c3)
select   c12
var      c5 = implication
          c3 = definition
          v1 = other
          c10 = lukasiewicz
          c9 = godel
          c4 = fuzzy-logic

```

Figure 12: Final structured query for topic 127

6.2 Topic 141

141 Retrieve sections about threads from articles about java

This is a textbook case, a short and syntactically unambiguous query, that works perfectly. See figure 13.

6.3 Topic 164

164 I am surveying the area of knowledge management for a report I am writing, and am looking for knowledge management frameworks and technologies for organizational memories.

This topic raises some of the problems described in section 5.3. Most of the proposed relations (see figure 14) are

```

from      c3 = /article
          c1 = c3/bdy/sec
where     about(c3, c4)
          about(c1, c2)
select   c1
var       c4 = java
          c2 = thread

```

Figure 13: Final structured query for topic 141

interesting and may be useful, but some terms (and a erroneous analysis of the phrase “I am writing ..”) generates noise (especially terms “write” (c4) and “report” (c3)).

```

from      v1 = *,
          v2 = *
where     ling_np_relation(c1, c3, for)
          ling_np_relation(c1, c2, nmodifier)
          ling_np_relation(c7, c6, nmodifier)
          ling_np_relation(c6, c5, nmodifier)
          ling_np_relation(c4, c8, for)
          ling_np_relation(c4, c7, for)
          ling_np_relation(c8, c9, for)
          ling_np_relation(c7, c9, for)
          ling_adj(c9, v6)
          about(v2, c4)
          about(v1, c1)
select   v1
var       c4 = verb_write,
          c8 = technology,
          c7 = framework,
          c9 = memory,
          c1 = management,
          v6 = organizational,
          c3 = report,
          c2 = knowledge,
          c6 = management,
          c5 = knowledge

```

Figure 14: Final structured query for topic 164

7. CONCLUSION

The Graal of Information Retrieval, which is to “build software that will analyse, understand, and generate results in response to queries that humans express naturally”, as said in the INEX NLPX track presentation, is far from being reached. It requests a detailed syntactic, semantic and pragmatic understanding of both queries and documents; this is impossible in the present state of our knowledge and resources.

In this paper we focused on a first step, a technique to translate natural language queries into a structured formal query language. On top of the ameliorations of this system, we now intend to work on the use of the linguistic information provided by our method, in order to improve an XML retrieval system.

8. REFERENCES

- [1] A. Arampatzis, T. van der Weide, C. Koster, and P. van Bommel. Linguistically-motivated Information

Retrieval. In A. Kent, editor, *Encyclopedia of Library and Information Science*, volume 69, pages 201–222. Marcel Dekker, Inc., New York, Basel, Dec. 2000.

- [2] P. Blackburn and J. Bos. *Representation and Inference for Natural Language; A first Course in Computational Semantics*. comsem, 1999.
- [3] C. Fabre and C. Jacquemin. Boosting Variant Recognition with Light Semantics. In *Proceedings of the 18th International Conference on Computational Linguistics, COLING 2000*, pages 264–270, Saarbrücken, Aug. 2000.
- [4] S. Feldman. NLP Meets the Jabberwocky: Natural Language Processing in Information Retrieval. Online, May 1999. <http://www.onlinemag.net/OL1999/feldman5.html>.
- [5] K. S. Jones. What is the role of NLP in text retrieval? In Strzalkowski [12], pages 1–24.
- [6] H. Kamp and U. Reyle. *From discourse to logic*. Kluwer Academic Publisher, 1993.
- [7] TreeTagger - a language independent part-of-speech tagger. <http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html>.
- [8] H. Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing*, Sept. 1994.
- [9] A. F. Smeaton. Information Retrieval: Still Butting Heads with Natural Language Processing? In M. Paziienza, editor, *Information Extraction - A Multidisciplinary Approach to an Emerging Information Technology*, volume 1299 of *Lecture Notes in Computer Science*, pages 115–138. Springer-Verlag, 1997.
- [10] A. F. Smeaton. Using NLP or NLP Resources for Information Retrieval Tasks. In Strzalkowski [12], pages 99–111.
- [11] K. Sparck Jones and J. I. Tait. Automatic Search Term Variant Generation. *Journal of Documentation*, 40(1):50–66, 1984.
- [12] T. Strzalkowski, editor. *Natural Language Information Retrieval*. Kluwer Academic Publisher, Dordrecht, NL, 1999.
- [13] T. Strzalkowski. Preface. In *Natural Language Information Retrieval* [12], pages xiii–xxiii.
- [14] XML Path Language (XPath). World Wide Web Consortium (W3C) Recommendation. <http://www.w3.org/TR/xpath>.
- [15] XQuery 1.0: An XML Query Language. World Wide Web Consortium (W3C) Working Draft. <http://www.w3.org/TR/xquery>.
- [16] XSL Transformation (XSL). World Wide Web Consortium (W3C) Recommendation. <http://www.w3.org/TR/xslt/>.

Kyungpook National University at INEX 2004: Interactive Track

Heesop Kim

heesop@knu.ac.kr

Dept. of Library & Information Science, Kyungpook National University, Daegu, 702-701, South Korea

Heejung Son

sonhjung@postech.ac.kr

Abstract

At the INEX 2004 Interactive Track, we intended to understand the searcher behavior by focusing on search characteristics when interacting with structured XML documents. We tried to find out whether there exists any correlation between 6 factors of search characteristics (i.e., number of query iterations, number of query terms used, number of unique query terms used, number of documents/components viewed, number of documents/components assessed, time spent) and the searcher's satisfaction. 8 subjects were take part in and followed the experimental guidelines from the INEX Interactive Track organizer with no modification. 16 transaction logs from the search sessions and 6 responses from the questionnaires per subject were collected. We used SPSS for Windows 10.0 as a statistical analysis tool and adopted Pearson's correlation coefficient (r) for correlation analysis. No significant correlation was found between satisfaction and the chosen 6 factors in this experiment. However, a much larger sample size could support more meaningful interpretation in the next experiment.

1. Introduction

In the INEX 2004 Interactive Track experiment we tried to investigate the behavior of searchers with structured XML documents, and find out which factors of 6 search characteristics (number of query iterations, number of query terms used, number of unique query terms used, number of

documents/components viewed, number of documents/components assessed, time spent) have a significant relationship with the searcher's satisfaction. We analyzed transaction logs from the search sessions and questionnaire data from each subject, collected according to the standard experimental guidelines from the INEX 2004 Interactive Track organizers.

2. System, database and tasks

HyREX (Hyper-media Retrieval Engine for XML) on-line search engine, which was provided by the track organizers, was used to retrieve the XML documents or their components, and to collect relevance assessments from the subjects. Of the two interfaces available in the system, we used the simple 'baseline interface' rather than the additional 'graphical interface.' INEX data collection of the *ad hoc* track (version 1.4), which consists of full text from the journals published by the *IEEE Computer Society*, was used in this track. The topic area of the collection is computer science, focusing on hardware and software development. The time span covered by the collection is 1995 to 2002.

Four tasks, divided into two categories, that is, 'Background category (B)' and 'Comparison category (C)', were used. The tasks were derived from the CO topics that were extended to engage the subjects in realistic searching behavior. Among topics

from the ‘Background’ category which express information need in the form of “I’d like to find out about X,” topics 180 and 192 were selected for two tasks in this category. Topics 188 and 198 were taken for two tasks of the ‘Comparison’ category, in which the topics express “Find differences between X and Y”. Each subject was asked to choose one task from each category¹.

3. Experimental design

3.1. System interface

We used HyREX baseline interface for the system interface. Figure 1 shows the ranked list of search results. Each result of the list consists of rank, article title, author(s), retrieval status value, and a pointer to the path of the retrieved document component. Up to 100 top-ranked results were returned for each query and 10 results were seen per page. Each detailed result page can be accessed by clicking on the path from the ranked list (see Figure 2). Each one provides its own *Table of Contents* on the left which represents the structure of the document and allows the links to the corresponding components. From a drop down box on the top and bottom of the detailed result page, the subjects were asked to submit relevance assessments for the retrieved document. Two aspects of relevance were assessed according to the followings: (1) the extent to which the displayed component contains information that is useful for solving the given task (Usefulness=Exhaustiveness) and (2) the extent to which the displayed component is focused on the topic of the task (Specificity). Finally, 10 categories combined from the two aspects were provided to the subjects for relevance assessments².

3.2. Experimental procedure

We adopted the experimental guidelines from the

organizers without any modification. The order of an experimental session was as follows:



Figure 1: HyREX interface for ranked list

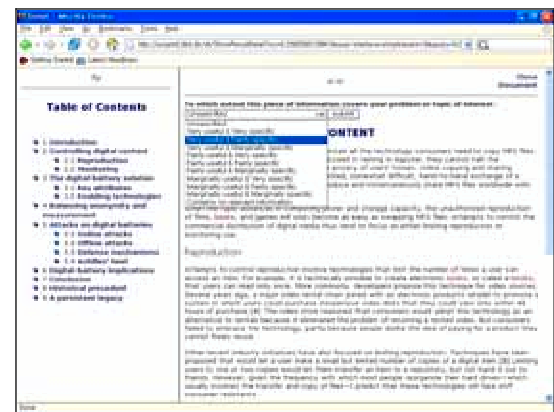


Figure 2: HyREX interface for detailed result

1. Brief information about the experiment and procedures was given to the subjects.
2. ‘Before-experiment questionnaire,’ which contains questions about the demographic information and search experience of the subjects, was filled in by the subjects.
3. ‘Introduction for Searchers’ was distributed to the subjects.
4. System tutorial was conducted. Test runs with the query ‘Information Retrieval’ were also performed by the subjects.
5. Category task selection from the first category was performed by the subjects.
6. ‘Before-each-task questionnaire’ was filled in by the subjects.
7. Conducted ‘Search’ with maximum 30 minutes.
8. ‘After-each-task questionnaire,’ which contain

¹ Four tasks are presented in the **Appendix I**

² 10 categories are found in **Figure 1**

questions about subjects' satisfaction with the results and the system interface, was filled in by the subjects.

9. Stages 5-8 were repeated for the other task category by the subjects.
10. 'Post-experiment questionnaire,' which contains questions about the whole tasks and the system, was filled in by the subjects.

Two transaction logs from each searcher session were collected, in addition to the questionnaire data.

3.3. Subjects

Eight volunteer subjects participated in this experiment. Seven of them are students of the Department of Library and Information Science at Kyungpook National University, and the other one is a researcher. Four of the subjects are male and four are female. All of them speak Korean as their first language. The ages of the subjects were between 24 and 31 (mean = 27.125) at the time of experiment. One of them has Master's degree, while half (4 subjects) have Bachelor's degree and the others (3 subjects) are undergraduate students. Two subjects have been participated in previous on-line searching studies as a test person, but none for as an experimenter. It is reported that our subjects have an average of 7.13 years of on-line searching experience (min. = 5, max. = 10, std. deviation = 2.03). Subjects' detailed levels of experience with searching are shown in Table 1.

Table 1: Subjects' Experience with Searching³

Searching experience	Mean	Std. Dev.
Computerized library catalogues	3.25	1.282
Digital libraries of scientific articles	3	0.926
WWW search engines	4.625	0.518
IEEE journal and magazines	2	1.069

As you can see from the table, our subjects are more

³ Based upon a 5 point scale in which 1= No, 3= Some, 5= A great deal

familiar with the WWW search engines than other kinds of library information systems or databases. The average frequency of performing a search on any kind of system was 4.875 based upon a 5 point scale in which 1=Never and 5=One or more times a day.

4. Results

4.1. Search characteristics

From the transaction logs we identified 9 possible factors of search characteristics that could be regarded as reflecting the searcher's behavior. Those factors were: (1) the total number of queries issued by the subject per session (Query iterations), (2) the total number of query terms used per session (Query terms used), (3) the average number of query terms used per iteration (Average query terms used), (4) the number of unique query terms used per session (Unique query terms), (5) the number of unique query terms derived from the content of the task (Unique query terms in task), (6) the number of documents (Documents/components viewed), (7) document components assessed (Documents/components assessed), (8) the time spent for each session (Time spent), (9) and the time spent until the 1st relevance assessment was submitted (Time of 1st relevance assessment). The most distinguishing 6 factors out of these 9 were selected according to the purpose of our research. Table 2 shows the overall search characteristics with all 9 candidate factors.

Table 3 and Figure 1 present the results for each task as the result of the relevance assessments from the subjects which obtained from the transaction logs.

Table 2: Overall Search Characteristics

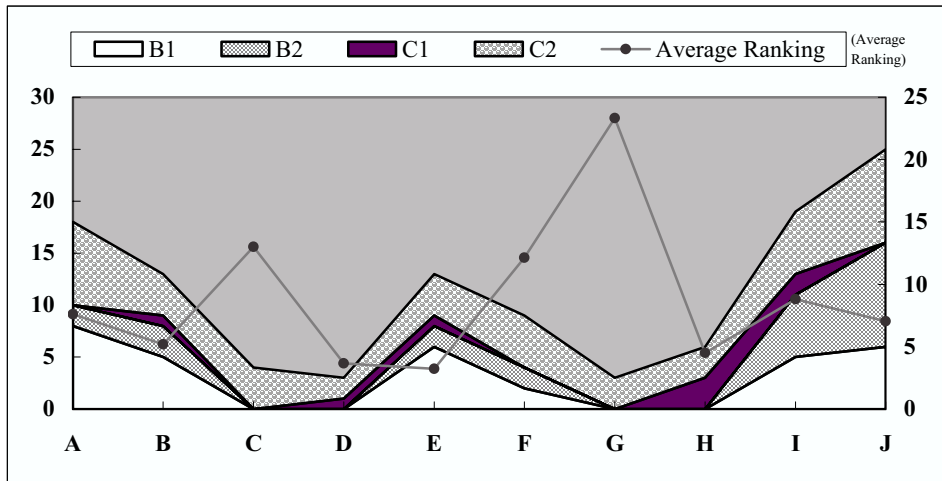
Subjects	Tasks	* Query iterations	* Query terms used	Average query terms used	* Unique query terms	Unique query terms in task	* Documents/components viewed	* Documents/components assessed	* Time spent	Time of 1 st relevance assessment
001	B1	7	32	4.571	11	11	8	2	0:29:16	0:17:30
	C2	2	4	2	4	4	5	2	0:07:10	0:02:32
002	C2	7	23	3.286	9	9	23	8	0:32:23	0:03:48
	B1	5	15	3	8	7	16	5	0:15:18	0:02:47
003	B1	3	8	2.667	7	7	7	4	0:14:30	0:02:38
	C2	8	33	4.125	18	17	9	7	0:19:16	0:03:15
004	C2	7	28	4	7	7	18	2	0:22:29	0:15:47
	B1	1	3	3	3	3	2	1	0:15:26	0:05:31
005	B2	6	11	1.833	5	5	16	7	0:23:50	0:01:44
	C2	8	18	2.25	7	6	17	12	0:26:20	0:04:38
006	C2	3	10	3.333	8	8	18	13	0:31:46	0:03:41
	B1	6	20	3.333	10	10	22	20	0:24:42	0:03:51
007	B2	4	6	1.5	4	4	56	8	0:27:03	0:07:26
	C2	3	6	2	5	5	21	4	0:24:40	0:04:04
008	C1	3	12	4	6	6	17	8	0:29:59	0:03:47
	B2	8	25	3.125	13	10	18	10	0:27:52	0:01:51
N		16	16	16	16	16	16	16	16	16
Minimum		1	3	1.5	3	3	2	1	0:07:10	0:01:44
Maximum		8	33	4.571	18	17	56	20	0:32:23	0:17:30
Mean		5.063	15.875	3.001	7.813	7.438	17.063	7.063	0:23:15	0:05:18
Std. Deviation		2.351	9.979	0.907	3.834	3.464	12.157	5	0:07:08	0:04:39

(*: Selected 6 factors for correlation analysis with subjects' satisfaction)

Table 3: Results of Relevance Assessments per Task

	B1	B2	C1	C2	Total	Percentage	Average Ranking ⁴
A	8	2	0	8	18	15.929	7.6
B	5	3	1	4	13	11.504	5.2
C	0	0	0	4	4	3.54	13
D	0	0	1	2	3	2.655	3.667
E	6	2	1	4	13	11.504	3.222
F	2	2	0	5	9	7.965	12.143
G	0	0	0	3	3	2.655	23.333
H	0	0	3	3	6	5.31	4.5
I	5	6	2	6	19	16.814	8.824
J	6	10	0	9	25	22.124	7.042
Total	32	25	8	48	113	100	

⁴ Average ranking of the documents or document components which were assessed with corresponding category for relevance assessment (A-J)



A: Very useful & Very specific, B: Very useful & Fairly specific, C: Very useful & Marginally specific
D: Fairly useful & Very specific, E: Fairly useful & Fairly specific, F: Fairly useful & Marginally specific
G: Marginally useful & Very specific, H: Marginally useful & Fairly specific, I: Marginally useful & Marginally Specific, J: Contains no relevant information

Figure 1: Results of Relevance Assessments per Task

We observed that ‘A’ (Very useful & Very specific, 15.929%), ‘E’ (Fairly useful & Fairly specific, 11.504%), and ‘I’ (Marginally useful & Marginally specific, 16.814%) take approximately 50% of the total relevance assessments; while ‘C’ (Very useful & Marginally specific, 3.54%) and ‘G’ (Marginally useful & Very specific, 2.655%) have only about 7%. The difference between the assessments from the subjects and the ranking from the system can be explained by ‘J’ (Contains no relevant information). ‘J’ account for 22.124%, the highest percentage, of the total assessments, but the average ranking (7.042) for the document components whose relevance was assessed with ‘J’ is higher than that whose relevance was assessed with ‘A’ (7.6).

4.2. Correlation analysis between search characteristics and subjects’ satisfaction

We examined the correlation between the 6 factors of search characteristics and the subjects’ satisfaction. Two questions (AQ3: “Are you satisfied with your search results?” and AQ4: “Do you feel that the task has been fulfilled?”) are adopted to measure the subjects’ satisfaction. As shown in Table 4, there was no significant correlation between the subjects’

satisfaction and the chosen 6 factors of search characteristics in $p < 0.05$. However, it is interesting to note that the level of system support (AQ9: “How well did the system support you in this task?”) and the average relevance of the information presented (AQ10: “On average, how relevant to the search task was the information presented to you?”) have a strong positive correlation ($p < 0.01$, $r = 0.769$) with the subjects’ satisfaction (AQ3: “Are you satisfied with your search results?”)⁵.

4.3. Subjects’ comments about the experiment and the system

After the experiment, all of the subjects were asked to answer the ‘Post-experiment questionnaire’ which was intended to find out how understandable the tasks were and how easy it was to learn and use the system. General comments about the system were also requested. The subjects’ responses about understandability of tasks (PQ2: “How understandable were the tasks?”), similarity to other searching tasks (PQ3: “To what extent did you find

⁵ Questions (AQ3-AQ4, AQ9-AQ10) are listed in the Appendix II

the tasks similar to other searching tasks that you typically perform?”), and easy to learn and using the system (PQ4: “How easy was it to learn to use the system?”; PQ5: “How easy was it to use the system?”; PQ6: “How well did you understand how to use the system?”) are shown in Table 5. It is noteworthy that

Table 4: Correlation Coefficients between 6 Factors of Search Characteristics and Subjects’ Satisfaction

		Query iterations	Query terms used	Unique query terms	Documents/ components viewed	Documents/ components assessed	Time spent
AQ3	coefficient	-.431	-.307	-.105	-.131	.062	-.406
	p-value	.096	.248	.698	.628	.819	.118
AQ4	coefficient	-.337	-.275	-.080	.188	.150	-.229
	p-value	.202	.303	.768	.484	.580	.395

Table 5: Selected Questions from the Post-experiment Questionnaire⁶

	PQ2	PQ3	PQ4	PQ5	PQ6
Mean	3.25	2.125	3.75	3.875	3.625
Std. Dev.	0.707	0.835	0.707	0.835	0.916

Table 6: Subjects’ Likes and Dislikes about the Search System and Interface

Likes about the search system and interface
- Easy to browse the search results (1)
- Easy to understand or learn the system (3)
- Display by the structured method (Provision of the “Table of Contents”) (4)
- Simple interface design (5)
- Search capabilities (“Keyword Highlighting,” rsv etc.) (3)
Dislikes about the search system and interface
- Delay of the response time (3)
- Lack of search functions and Boolean operators (7)
- Unstableness of the ranked results (2)
- Duplication of the same article in the same result (3)
- Too short information on the ranked result list (1)
- Display of the complex document by the tree structure (1)

(N): Number of subjects

⁶ Based upon a 5 point scale in which 1=Not at all, 3=Somewhat, 5=Extremely; Questions are listed in the Appendix III

the subjects’ are satisfied with the system’s user interface but they seem unfamiliar with searching XML documents which showed poorest result. Table 6 presents the subjects’ likes and dislikes about the search system and its interface.

5. Conclusion and Discussion

In this experiment, we tried to discover which factors of search characteristics influence the searcher’s satisfaction. No significant correlation was found between the factors of search characteristics and searcher’s satisfaction, but we expect a much larger sample size to produce more statistically meaningful interpretation in the next experiment. Also, it would be interest to compare the subjects’ demographic factors as many previous user studies carried out. In addition, we would like to make comparison between the graphical interface and the simple baseline interface in the future experiment.

It also seems that the nominal scale of relevance assessment is not that helpful in analyzing the results. Therefore, a different kind of scale for relevance assessment could be considered in the next experiment.

References

- Belkin, N. J., Cool, C., Kelly, D., Kim, G., Kim, J. Y., Lee, H. J., Muresan, G., Tang, M. C., Yuan X. J. (2002) “Rutgers Interactive Track at TREC 2002.” In E. M. Voorhees and L. P. Buckland (Eds.). *The Eleventh Text REtrieval Conference, TREC 2002* (pp.

539-548). Washington, D.C.: GPO.

Craswell, N., Hawking, D., Thom, J., Upstill, T., Wilkinson, R., Wu, Mingfang. (2002) "TREC11 Web and Interactive Tracks at CSIRO." In E. M. Voorhees and L. P. Buckland (Eds.). *The Eleventh Text REtrieval Conference, TREC 2002* (pp. 197-206). Washington, D.C.: GPO.

Fuhr, N., Malik, S., Lalmas, M. (2004) "Overview of the INitiative for the Evaluation of XML Retrieval (INEX) 2003." In N. Fuhr, M. Lalmas and S. Malik (Eds.). *INEX 2003 Workshop Proceedings* (pp. 1-11). INEX 2004 Interactive track guidelines: [online available] at: [http://inex.is.informatik.uni-
duisburg.de:2004/tracks/int/](http://inex.is.informatik.uni-duisburg.de:2004/tracks/int/)

Appendix

I. Four tasks per category

- Background(B)

Task ID: B1

You are writing a large article discussing virtual reality (VR) applications and you need to discuss their negative side effects. What you want to know is the symptoms associated with cybersickness, the amount of users who get them, and the VR situations where they occur. You are not interested in the use of VR in therapeutic treatments unless they discuss VR side effects.

Task ID: B2

You have tried to buy & download electronic books (ebooks) just to discover that problems arise when you use the ebooks on different PC's, or when you want to copy the ebooks to Personal Digital Assistants. The worst disturbance factor is that the content is not accessible after a few tries, because an invisible counter reaches a maximum number of attempts. As ebooks exist in various formats and with different copy protection schemes, you would like to find articles, or parts of articles, which discuss various proprietary and covert methods of protection. You would also be interested in articles, or parts of articles, with a special focus on various disturbance factors surrounding ebook copyrights.

- Comparison(C)

Task ID: C1

You have been asked to make your Fortran compiler compatible with Fortran 90, and so you are interested in the

features Fortran 90 added to the Fortran standard before it. You would like to know about compilers, especially compilers whose source code might be available. Discussion of people's experience with these features when they were new to them is also of interest.

Task ID: C2

You are working on a project to develop a next generation version of a software system. You are trying to decide on the benefits and problems of implementation in a number of programming languages, but particularly Java and Python. You would like a good comparison of these for application development. You would like to see comparisons of Python and Java for developing large applications. You want to see articles, or parts of articles, that discuss the positive and negative aspects of the languages. Things that discuss either language with respect to application development may be also partially useful to you. Ideally, you would be looking for items that are discussing both efficiency of development and efficiency of execution time for applications.

II. Selected questions from the After-each-task questionnaire

AQ3: Are you satisfied with your search results?

AQ4: Do you feel that the task has been fulfilled?

AQ9: How well did the system support you in this task?

AQ10: On average, how relevant to the search task was the information presented to you?

III. Selected questions from the Post-experiment questionnaire

PQ2: How understandable were the tasks?

PQ3: To what extent did you find the tasks similar to other searching tasks that you typically perform?

PQ4: How easy was it to learn to use the system?

PQ5: How easy was it to use the system?

PQ6: How well did you understand how to use the system?

Monday, December 6, 2004	Tuesday, December 7, 2004	Wednesday, December 8, 2004
09.00-09.20 Opening Norbert Fuhr	09.00-10.20 Session 4: Ad hoc retrieval II Chair: Patrick Gallinari Component ranking and Automatic Query Refinement for XML Retrieval. <i>Yosi Mass and Matan Mandelbrod</i>	09.00-10.20 Session 5: Ad hoc retrieval III Chair: Anastasios Tombros An algebra for Structured Queries in Bayesian Networks. <i>Jean-Noël Vittaut, Benjamin Piwowarski and Patrick Gallinari</i>
09.20-10.40 Session 1: Ad hoc retrieval I Chair: Ray Larson The University of Amsterdam at INEX 2004. <i>Börkur Sigurbjörnsson, Jaap Kamps and Maarten de Rijke</i> TRIX 2004 struggling with the overlap. <i>Jaana Kekäläinen, Marko Junkkari, Paavo Arvola and Timo Aalto</i> Hybrid XML Retrieval Revisited. <i>Jovan Peheveski, James A. Thom and Anne-Marie Vercoustre</i> TIJAH at INEX 2004 Modeling Phrases. <i>Vojkan Mihajlovic, Georgina Ramirez, Arjen P. de Vries, Djoerd Hiemstra and Henk Ernst Blok</i>	The Utrecht Blend: Basic Ingredients for an XML Retrieval System. <i>Roelof van Zwol, Frans Wiering and Virginia Dignum</i> Hierarchical Language Models for XML Component Retrieval. <i>Paul Ogilvie and Jamie Callan</i> Analyzing the Properties of XML Fragments decomposed from the INEX Document Collection. <i>Kenji Hatano, Hiroko Kinutani, Toshiyuki Amagasa, Yasuhiro Mori, Masatoshi Yoshikawa and Shunsuke Uemura</i>	GPX - Gardens Point XML Information Retrieval at INEX 2004. <i>Shlomo Geva</i> Merging XML Indices. <i>Giambattista Amati, Claudio Carpineto and Giovanni Romano</i> Ranked Retrieval of Structured Documents with the STerm Vector Space Model. <i>Felix Weigel, Klaus U. Schulz and Holger Meuss</i>
10.40-11.00 Coffee	10.20-10.40 Coffee	10.20-10.40 Coffee
11.00-12.20 Session 2: Heterogeneous collection track Chair: Arjen de Vries Building and Experimenting with a Heterogeneous Collection. <i>Zoltan Szlavik and Thomas Rölleke</i> Cheshire II at INEX 04: Fusion and Feedback for the Ad hoc and Heterogeneous Tracks. <i>Ray R. Larson</i> Using a relevance propagation method for Ad hoc and Heterogeneous tracks in INEX 2004. <i>Karen Sauvagnat and Mohand Boughanem</i> A Test Platform for the INEX Heterogeneous Track. <i>Serge Abiteboul, Ioana Manolescu, Benjamin Nguyen and Nicoleta Preda</i>	10.40-11.20 Session 6: Natural language track Chair: Trond Aalberg NLPX at INEX 2004. <i>Alan Woodley and Shlomo Geva</i> Analysing Natural Language Queries at INEX 2004. <i>Xavier Tannier, Jean-Jacques Girardot and Mihaela Mathieu</i>	10.40-11.30 Workshop Reports II Interactive track (<i>Birger Larsen</i>), Heterogeneous collection and topic format (<i>Thomas Rölleke and Börkur Sigurbjörnsson</i>)
	11.20-12.00 Relevance feedback track Chair: Anja Theobald TIJAH at INEX 2004 Modeling Relevance Feedback. <i>Vojkan Mihajlovic, Georgina Ramirez, Arjen P. de Vries, Djoerd Hiemstra and Henk Ernst Blok</i> Relevance feedback for XML retrieval. <i>Yosi Mass and Matan Mandelbrod</i>	11.30-12.00 INEX 2005 Track proposals Gabriella Kazai
12.20-14.00 Lunch	12.00-12.20 Interactive track Chair: Jaana Kekäläinen The Interactive Track at INEX 2004. <i>Anastasios Tombros, Birger Larsen and Saadia Malik</i>	12.00-12.15 Any other business and Closing Mounia Lalmas
14.00-15.00 Session 3: Methodology Chair: Norbert Fuhr NEXI, Now and Next. <i>Andrew Trotman and Börkur Sigurbjörnsson</i> Reliability Tests for the XCG and inex-2002 Metrics. <i>Gabriella Kazai, Mounia Lalmas and Arjen de Vries</i> Some statistics about INEX 2004. <i>Mounia Lalmas, Norbert Fuhr, Saadia Malik, Zoltan Szlavik and Vu huyen Trang</i>	12.20-14.00 Lunch	12.15-14.00 Lunch
15.00-15.30 Workshop descriptions and grouping Mounia Lalmas	14.00-15.30 Workshop sessions II (methodology and tracks)	14.00-
15.30-16.00 Coffee	15.30-16.00 Coffee	15.30-16.00 Coffee
16.00-17.30 Workshop sessions I (methodology and tracks)	16.00-17.45 Workshop reports I Metrics (<i>Norbert Fuhr</i>), Relevance (<i>Mounia Lalmas</i>), Relevance feedback track (<i>Yosi Mass</i>), Natural language track (<i>Shlomo Geva</i>)	
18.00-20.00 Dinner	18.00-20.00 Dinner	18.00-20.00 Dinner



INEX'04 Guidelines for Topic Development

The aim of the INEX initiative is to provide means, in the form of a large test collection and appropriate scoring methods, for the evaluation of content-oriented XML 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.

1 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, is a retrievable unit. Regarding the topics, with the use of XML query languages, users of an XML retrieval system are able to combine both content and structural conditions within their information need and restrict their search to specific structural elements within an XML collection. Finally, the relevance assessments for an XML collection must also consider the structural nature of the documents and provide assessments at different structural levels.

This guide deals only with the topics of the test collection and provides detailed guidelines for their creation for INEX 2004. The guide is organized as follows. Section 2 describes the topic creation criteria; Section 3 explains the topic format; and Section 4 describes a procedure for topic development. Appendix A provides example topics.

2 Topic creation criteria

Creating a set of topics for a test collection requires a balance between competing interests. It is a well-known fact that 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, 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 account.

1. **The author of a topic should be either an expert or the very least be familiar with the subject area covered by the collection!** (Note that the author of a topic should also be the assessor of relevance!).
2. Topics should reflect what real users of operational systems might ask.
3. Topics should be representative of the type of service that operational systems might provide.
4. Topics should be diverse.
5. Topics may also differ in their coverage, e.g. broad or narrow topic queries.

2.1 Topic types

As in previous years, in INEX 2004 we distinguish two types of topics, the topic types reflect two user profiles, where the users differ in the amount of knowledge they have about the structure of the collection:

Content-only (CO) topics: are traditional IR topics written in natural language and constrain the content of the desired results.

The CO topics simulate users who do not (want to) know, or do not want to use, the actual structure of the XML documents. This profile is likely to fit most users searching XML digital libraries.

Content-and-structure (CAS) topics: are topic statements, that not only restrict content of interest but also contain explicit references to the XML structure.

The CAS topics simulate users that have some knowledge of the structure of the XML. Those users might want to use this knowledge to try to make their topics more concrete, by adding structural constraints. This user profile could fit librarians that have some knowledge of the collection structure.

Both the CO and the CAS topics are IR topics, in the sense that they do not have a strict semantics. Both topic types should be thought of as hints to the retrieval system. The final judgment of relevance will necessary always be in the hands of a human assessor.

3 Topic format

Both CO and CAS topics are made up of four parts. The parts explain the *same* information need, but for different purposes.

Title: a short explanation of the information need. It serves as a summary of both the content and, in the case of CAS topics, also the structural requirements of the user's information need. The exact format of the topic title is discussed in more detail later in this section.

Description: a one or two sentence natural language definition of the information need.

Narrative: a detailed explanation of the information need and the description of what makes a document/component relevant or not. The narrative should explain not only what information is being sought for, but also the context and motivation of the information need is, i.e., *why* the information is being sought and what work task it might help to solve. Note that the assessments will be made on the basis of the narrative alone. It is therefore important that this description is clear.

Keywords: a set of comma-separated scan terms that are used in the collection exploration phase of the topic development process (see Section 4) to retrieve relevant documents/components. Scan terms may be single words or phrases and may include synonyms, and terms that are broader or narrower terms than those listed in the topic description or title.

Note that the *title* and the *description* must be interchangeable. In addition, the *topic title* and the *topic narrative* can be viewed respectively as a formal expression and an extension of the *topic description*. In the next section we will explain the format of the topic title. The other fields will not get their own sections, but will be referred to in Section 4.

3.1 Topic title

In this section we will give a high level description of the format of the title part of the topics. A more formal specification of both the CO and CAS topic can be found on the INEX web-page.

<http://inex.is.informatik.uni-duisburg.de:2004/internal/pdf/NEXI.pdf>

To make sure that your topics are syntactically correct, parsers have been implemented in Flex and Bison (the GNU tools compatible with LEX and YACC). Online version of the parsers is available:

<http://metis.otago.ac.nz/abin/nexi.cgi>

3.1.1 CO Topics

The topic title of a CO topic is a short, usually a 2-5 terms representation of the topic statement. 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. Let's look at two examples.

Example: A user wants to retrieve documents/components about computer science degrees that are not master degrees:

"computer science" +degree -master

Note that 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, although 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 document/components about information retrieval from semi-structured documents:

"information retrieval" +semi-structured documents

As in the previous example the following text might be judged relevant, even if 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. ...

Note that, although the semantics of phrases and the +/- tokens is not strict, they can potentially be of use to the retrieval engine. A full example of a CO topic is given in Appendix A.1.

3.1.2 CAS Topics

The title of CAS topics is written in an XPath dialect (<http://www.w3c.org/TR/xpath>). In short, the CAS titles are 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 **contains** function in XPath. We will restrict the usage 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. The **about** function is used to say that the content of the element located by the path is about the information need expressed in the query. Note that, as with the CO topics, the title is meant to be a hint for the search engine and does not have definite semantics.

Example: Suppose a user wants to know what the INEX participants said about native XML databases last year. The user assumes that the conference proceedings are mentioned somewhere in the front matter.

```
//article[.//fm//yr = 2003 and about(.//fm, INEX proceedings)]/*[about(. ,
native XML databases)]
```

Note that 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 proceedings)
C = /*
D = about(. , native XML databases)
```

Example: Suppose a user want 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 she might write the query:

```
//article[about(.//abs, flight simulator)
and about(.//p, implementation java)
and about(. , flight simulator java)]
```

When looking at the results the user is not likely to be picky whether the results fit her query exactly. The user might for example be happy with articles which do not have an abstract, as long as they are about Java implementation of flight simulators.

Note that 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 are the human made relevance assessments. Thus, each query **MUST** have at least one about function in the rightmost predicate.

Equivalent tags In the current INEX collection there are several tags used interchangeable (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: lrj, 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

A full example of a CAS topic is given in Appendix A.2.

4 Procedure for topic development

Each participating group will have to submit **3 CO** and **3 CAS** topics by the **May 3rd 2004** by filling in the Candidate Topic Form (one per topic) which is on the INEX web-site.

<http://inex.is.informatik.uni-duisburg.de:2004/internal/TopicSubmission.html>

This section outlines the procedures involved in the development of candidate topics. The topic creation process is divided up into several steps. When developing a topic, use the on-line version of the Candidate Topic Form (or a printout of it) to record all information on the topic you are creating.

Step 1: Initial topic statements 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 is, i.e., *why* the information is being sought and what work task it might help to solve.

Step 2: Collection exploration In this step the initial topic statement is used to explore the document collection in order to obtain an estimate of the number of relevant documents/elements in the collection and to evaluate whether this topic can be judged consistently in the assessment phase. You may use any retrieval engine for this task, including your own or HyRex (HyRex can be accessed via the INEX website).

<http://inex.is.informatik.uni-duisburg.de:2004/hyrex/>

While exploring the collection record the set of keywords that you used for retrieval (make sure to record all the keywords from all the iterations of your search or, if you use query expansion strategies, the query terms generated by the process). Think of words that would make good scan words when assessing. List those words in the keywords field of the Candidate Topic Form.

Step 2a: Assess top 25 results Judge the top 25 documents/components of your retrieval result. To assess the relevance of a retrieved document/component use the following working definition: mark a document/component 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 document/component should be judged on its own merits. That is, a document/component is still relevant even if it is the thirtieth document/component you have seen with the same information. It is crucial to obtain exhaustive relevance judgments. It is also very important that your judgment of relevance is consistent throughout this task. Using the Candidate Topic Form record the number of found relevant components and the XPath path representing each relevant element (see the form for details).

- If you find *less than 2* or *more than 20* relevant components within the top 25 results, you should abandon the topic and start with a new one.
- If you have found *at least 2* relevant components and *no more than 20*, perform a feedback search (explained below).

Step 2b: Feedback search After assessing the top 25 documents/components, you should have gotten an idea about which words (if any) could be added to the query to make the query as expressive as possible for the kind of documents you wish to retrieve. Add those words in the keywords field of the Candidate Topic Form. Use the expanded query, to retrieve a new list of relevant document/components.

Step 2c: Assess top 100 results Judge the top 100 documents/components (some of them you will have judged already), and record the number of relevant documents/components in Candidate Topic Form.

Step 2d: Write narrative Having judged the top 100 documents/components you should have gained a clear idea about what makes a component relevant or irrelevant, and thus you should know how you will judge the topic in the assessment phase. Write this idea in the narrative field. The narrative should be a detailed explanation of the information need and what makes a document/component relevant. Record not only what information is being sought for, but also the context and motivation of the information need. Make sure your description is as exhaustive as possible as there will be a couple of months gap before you will return to the topic for relevance assessments.

Step 3: Refining topic statements Refining the topic statement means finalizing the topic title, description, keywords and narrative. Note that it is important that the four parts all express the same information need; it should be possible to use each of the four parts of a topic in a stand-alone fashion (e.g. using only the title for retrieval, or only the description for filtering etc.).

Topic Submission Once you finished, submit the on-line Candidate Topic Form on the INEX website.

<http://inex.is.informatik.uni-duisburg.de:2004/internal/TopicSubmission.html>

Make sure you submit all **6** candidate topics no later than the *May 3rd 2004*.

Topic selection

From the received candidate topics, we (the clearinghouse) will then decide which topics to use such that a wide range of likely number of relevant documents is included. The data obtained from the collection exploration phase will be used as input to the topic selection process. We will then distribute final set of topics back to you to be used for the retrieval and evaluation.

We would like to thank you for your contribution.

Acknowledgments

The topic format proposed in this document is based on the outcome of working groups set up during the INEX 2002 and 2003 workshops in Dagstuhl. We are very grateful for their contribution. This document is a modified version of the topic development guides for INEX 2002 and 2003.

Authors:

Börkur Sigurbjörnsson, Birger Larsen, Mounia Lalmas and Saadia Malik

April 13th 2004

A Examples

A.1 Content-Only topic

```
<inex-topic query_type="CO">
<title>+"open standards" +"digital video" +"distance learning"</title>
<description>We are looking for articles that discuss open standards behind
media streaming and distribution of digital video that may be useful for
distance learning projects.</description>
<narrative>Our aim is to use our acquired knowledge and experience with
digital video in a distance learning project. We prefer, however, to use
software and methods that are based on open standards only. We are looking
for articles/components discussing methodologies of digital video production
and distribution that respect free access to media content through internet
or via CD-ROMs or DVDs in connection to the learning process. Discussions of
open versus proprietary standards of storing and sending digital video will
be appreciated.</narrative>
<keywords>media streaming, video streaming, digital video, distance
learning, open standards, open technologies, free access</keywords>
</inex-topic>
```

A.2 Content-and-Structure topic

```
<inex_topic query_type="CAS">
<title>//article[about(../bb, Rumbaugh Jacobson Booch) and
about(../abs, formal methods logic)]//sec[about(., UML formal
logic)]</title>
<description>Find information on the use of formal logics to model or
reason about UML diagrams.</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>
<keywords>unified modeling language, first order logic, temporal
logic, description logic, formal verification, theorem proving, model
checking, Kripke structures, Spin, Promela, SMV</keywords>
</inex_topic>
```

Narrowed Extended XPath I (NEXI)

Andrew Trotman
Department of Computer Science
University of Otago
Dunedin, New Zealand
andrew@cs.otago.ac.nz

Börkur Sigurbjörnsson
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands
borkur@science.uva.nl

ABSTRACT

INEX has through the years provided two types of queries: Content-Only queries (CO) and Content-And-Structure queries (CAS). The CO language has not changed much, but the CAS language has been more problematic. For the CAS queries, the INEX 02 query language proved insufficient for specifying problems for INEX 03. This was addressed by using an extended version of XPath, which, in turn, proved too complex to use correctly. Recently, an INEX working group identified the minimal set of requirements for a suitable query language for future workshops. From this analysis a new IR query language NEXI is introduced for upcoming workshops.

1. Introduction

The INEX [4] query working-group recently identified the query language requirements for future workshops. While no changes were suggested for the CO queries, several amendments were suggested for the CAS queries. The most over-riding requirement was a language continuing to look like XPath [2], but not XPath. An alternative syntax was proposed at the workshop [6].

The working group identified many aspects of XPath to be dropped (e.g. functions), aspects to be severely limited (e.g. the only operator to be allowed in a tag path is the descendant operator). New features were also added (e.g. the about() filter). The shape of XPath was considered appropriate while the verbosity was considered inappropriate. The complete list of changes is outlined in the working group report [8]. Amendments were considered sufficient to warrant an XPath derivative language. NEXI is now introduced as that language. Extra to the working group list, the use of wildcards in search terms has been dropped.

The most significant diversion from XPath is semantics. Whereas in XPath the semantics are defined, in NEXI the retrieval engine must deduce the semantics from the query. This is the information retrieval problem – and to do otherwise is to make it a database language. For clarity, strict and loose interpretations of the syntax are included herein, however these should not be considered the only interpretations of the language.

A NEXI parser has been implemented in Flex [7] and Bison [3] (the GNU tools compatible with LEX and YACC). The parser is made available for public use (and is included in the appendices). The existing INEX queries (queries 1-126) have been translated into NEXI (where possible) and are also included.

2. Query types

There are currently two query types in INEX, the content only (CO) query and the content and structure (CAS) query [5].

2.1. The Content Only (CO) query

This is the traditional information retrieval query containing words and phrases. No XML [1] element restrictions are allowed, and no target element is specified. This kind of query occurs when a user is unfamiliar with the tagging structure of the document collection, or does not know where the result will

be found. To answer a CO query a retrieval engine must deduce the information need from the query, identify relevant elements (of relevant documents) in the corpus, and return those sorted most to least relevant.

Deduction of the information need from the query is to determine semantics from syntax. This is the information retrieval problem, the problem being examined at INEX. As such, the queries must be considered as “hints” as to how to find relevant documents. Some relevant documents may not satisfy a strict interpretation of the query. Equally, some documents that do satisfy a strict interpretation of the query may not be relevant.

2.2. The Content And Structure (CAS) query

Content and structure queries may contain either explicit or implicit structural requirements. Such a query might arise if a user is aware of the document structure. To answer a CAS query a retrieval engine must deduce the information need from the query, identify elements that match structural requirements, and return those sorted most to least relevant. CAS queries can be interpreted in two ways, either strictly (SCAS) or loosely (VCAS).

2.2.1. The SCAS interpretation

The target structure of the information need can be deduced exactly from the query. All target-path constraints must be upheld for a result to be relevant. If a user asks for <sec> tags to be returned, these must be returned. All other aspects of the query are interpreted from the IR perspective, i.e. loosely.

2.2.2. The VCAS interpretation

Specifying an information need is not an easy task, in particular for semi-structured data with a wide variety of tag-names. Although the user may think they have a clear idea of the structural properties of the collection, there are likely to be aspects to which they are unaware. Thus we introduce a vague interpretation where target-path requirements need not be fulfilled. Relevance of a result will be based on whether or not it satisfies the information need. It will not be judged based on strict conformance to the target-path of the query

3. The INEX Topic Format

This discussion of the INEX topic format is included for context. As the topic format is likely to change from year to year readers are advised to consult the latest edition of the guidelines for topic development for complete details.

3.1. Restrictions on Queries

For an individual query to be useful for evaluation purposes it must satisfy several requirements (the details of which are explained below):

- It must be interpretable loosely. To satisfy this requirement, every query must contain at least one about() clause requiring an IR interpretation (i.e. non-numerical). That clause must occur in the final filter. In //A[B] queries, this is B. In //A[B]/C[D], this is D.
- It must not be a simple mechanical process to resolve the path. To satisfy this requirement, every query must be in the form //A[B] or //A[B]/C[D]. The form //A[B]/C is not allowed at INEX as the resolution of //C from //A[B] is a simple mechanical process.
- It must have more than 5 known results. If this cannot be satisfied, abandon the query and choose another.
- It must be “middle” complex. Perform the search and examine the top 25 results. If there are less than 2 or more than 20 relevant results, the query is not middle-complex.
- Queries should reflect a real information need. Contrived queries are unlikely to be accepted.
- Queries should be diverse. If submitting more than one query, please make each different.

3.2. Equivalence Tags

In the current INEX collection there are several tags used interchangeable (for historical paper-publishing reasons). Tags belonging to the following groups are considered equivalent and interchangeable in a query:

Paragraphs:

ilrj, ip1, ip2, ip3, ip4, ip5, item-none, p, p1, p2, p3

Sections:

sec, ssl, 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

Due to tag equivalence, the query

```
//article//sec[about(../p, Computer)]
```

and

```
//article//ss2[about(../item-none, Computer)]
```

are identical.

3.3. Submission format

Topics are submitted in the INEX topic format detailed each year in the annual guidelines for topic development [5]. Detailed here is the 2003 format, which to date has not changed for subsequent workshops.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT inex_topic (title, description, narrative, keywords)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT narrative (#PCDATA)>
<!ELEMENT keywords (#PCDATA)>

<!ATTLIST inex_topic
  topic_id CDATA #REQUIRED
  query_type CDATA #REQUIRED
>
```

<inex_topic topic_id=""> – Supplied by INEX once all topics have been collected. This and other attributes may be present in the final topics selected by INEX.

<inex_topic query_type=""> – either “CO” or “CAS”. This attribute determines whether the topic is a content only (CO) or content and structure (CAS) topic. It consequently determines the query type used in the <title> tag.

<title> – a NEXI query (either CO or CAS, depending in the query_type attribute of the inex_topic tag). It should be noted the usual XML character encoding will be necessary, this includes substituting ‘<’ with ‘<’. See sections 4 and 5 for details.

<description> – a short (one or two sentence) natural language translation of the title. Although this can be used by any track, it is also used by the Natural Language track as the query specification.

<narrative> – a detailed explanation of the information need including a description of what makes a result relevant. It should be possible for someone other than the author to read the narrative and a result and determine unambiguously if the result is relevant or not.

<keywords> – a comma separated list of terms and phrases used during the topic formulation.

It is important that the title, description, and narrative all describe the same information need.

3.4. Example of an INEX topic

```
<inex_topic query_type="CAS">
  <title>
    //article[./yr = 2001 or ./yr = 2002]//sec[about(.,summer holidays)]
  </title>
  <description>
    Summer holidays either of 2001 or of 2002.
  </description>
  <narrative>
    Return section elements, which are about summer holidays, where the
    sections is descendent of article element, and the article is from 2001
    or 2002.
  </narrative>
  <keywords>
    summer, holiday, 2001,2002
  </keywords>
</inex_topic>
```

3.5. Topic Titles

The topic title contains the information retrieval query expressed in NEXI. The syntax of such queries is precisely defined below and a parser written in FLEX and BISON is included in the appendices. It is the information retrieval problem to deduce the semantics from the information need, however no meaningful language can exist without semantics. This duality can only be resolved by strictly defining the semantics to be loose.

4. The Content Only (CO) query

4.1. Searching for words and numbers

The smallest searchable unit in a CO query is the word:

```
word: NUMBER | ALPHANUMERIC

ALPHANUMERIC: {LETTER}{LETTERDIGITEXTRAS}*
NUMBER: "-"?{DIGIT}+

LETTER: [a-zA-Z]
DIGIT: [0-9]
LETTERDIGIT: [a-zA-Z0-9]
LETTERDIGITEXTRAS [a-zA-Z0-9'-]
```

Positive numbers, negative numbers and sequences of alphanumerics preceded by an alphabetic character are all valid search words. Alphanumerics have already been used in query 41 so must be included. Hyphens are allowed after the first character of an alphanumeric (to avoid confusion with term restrictions, see section 4.3). The apostrophe can only occur after the first character of an alphanumeric.

Example: To search for the single word Apple, the CO query is

```
Apple
```

Loose interpretation: It is anticipated that using the word Apple will help locate relevant documents. I won't tell you if I mean "Macintosh Computer", "Granny Smith", or "Mr Apple" but find what I want anyway.

4.2. Searching for phrases

A phrase is a double quoted sequence of words:

```
phrase: '"' word_list '"'
word_list: word word | word_list word
```

A phrase must contain two or more words. A phrase containing only one word is erroneous and the quotes should be removed to make a single word query.

Example: To search for Charles Babbage, the CO query will be

```
"Charles Babbage"
```

Loose interpretation: Relevant documents are anticipated to contain these two words adjacent to each other, but need not. They may contain both words non-adjacent. For that matter they might not contain both words. A relevant document might not even contain either word.

4.3. Term restrictions

Terms can be preceded by either a plus (+) or minus (-) sign

```
term: term_restriction unrestricted_term
term_restriction: EMPTY | '+' | '-'
unrestricted_term: word | phrase
```

Loose interpretation: The '+' signifies the user expects the word will appear in a relevant element. The user will be surprised if a '-' word is found, but this will not prevent the document from being relevant. Words without a sign are specified because the user anticipates such terms will help the search engine to find relevant elements. As restrictions are only hints, it is entirely possible for the most relevant element to contain none of the query terms, or for that matter only the '-' terms.

4.4. CO queries

A CO query is a sequence of one or more searchable terms.

```
co : term | co term
```

Example:

```
+"face recognition" approach
```

Loose interpretation: "I expect the phrase 'face recognition' will appear in a relevant document, I also anticipate the word 'approach' will help you find the documents I want".

4.5. Bag of Words

Term ordering in IR queries is often assumed to be irrelevant. In the "bag of words" interpretation, a query is an unordered set of search terms (and phrases). The assumption does not hold true for some queries. For example,

```
computer history
```

and

```
history computer
```

express different information needs even though the "bag of words" is identical.

Additionally, if a term occurs multiple times, the occurrence count is lost when the term is added to the "bag of words". For some queries, multiple term occurrences are needed to adequately specify the information need. For example, the query

```
The The
```

should search for documents about the well known rock band of the same name, and cannot be specified without the use of the multiple occurring term. Further, some search engines "stop" common words not considered useful for searching (such as the, and, of, etc). This query requires the use of such a term.

Loose interpretation: There may or may not be an implied order to the terms in a query. If a term occurs multiple times this may or may not imply meaning. Stopping common words may or may not alter the meaning of the query.

4.6. The pitfalls of queries

The minus sign (-) maintains two meanings; it is used for both exclusionary terms and negative numbers. For the purpose of clarity, 12 and -12 are numbers. By inserting a space (represented as '␣' in this paragraph) between the - and the 12 (-␣12), the meaning is changed to exclusionary. "Don't search for the number -12" can be expressed as --12 or -␣12. Equally, --␣12 is an error.

5. The Content and Structure (CAS) query

CAS queries can take three possible forms:

//A[B]	Return A tags about B
//A[B]//C	Return C descendants of A where A is about B (used in INEX'02)
//A[B]//C[D]	Return C descendants of A where A is about B and C is about D

A and C are paths whereas B and D are filters. The syntax is defined as:

```
cas: path cas_filter | path cas_filter path | path cas_filter path cas_filter
cas_filter: '[' filtered_clause ']'
```

Use of the form //A[B]//C is not useful for information retrieval evaluation purposes. Once the result of //A[B] has been determined, it is a mechanical process to extract the //C descendants. Use of this form was deprecated in INEX'03.

5.1. Path specification

Tag and attribute names follow the XML 1.1 [1] specification

```
XMLTAG: {XML_NAME}{XML_NAMECHAR}*
XML_NAMECHAR: [-_.:a-zA-Z0-9]
XML_NAME: [_:a-zA-Z]
```

Element nodes in the XML tree are identified as “//tag” and attribute nodes as “//@attribute”. The wildcard “//*” is included to identify first or subsequent descendant (tag or attribute). Convoluted use of attributes and wildcards is discouraged.

```
node: named_node | any_node | tag_list_node
NODE_QUALIFIER: " //"
named_node: NODE_QUALIFIER tag
attribute_node: NODE_QUALIFIER '@' tag
any_node: NODE_QUALIFIER '*'
```

In cases where either tag A or tag B is required, it is written “//(A|B)”.

```
tag_list: tag '|' tag | tag_list '|' tag
tag_list_node: NODE_QUALIFIER '(' tag_list ')'
```

A path through the XML tree is specified as a sequence of nodes. The only relationship between nodes in a path is descendant. There is no way to specify the child relationship or other XPath axes. Attributes cannot have descendant nodes so may only be specified at the end of a path.

```
path: node_sequence | node_sequence attribute_node
node_sequence: node | node_sequence node
```

Strict interpretation: “//A” is any A tag in the tree. “//A//B”, any B descendant of an A tag in the tree. “//@C” is the C attribute of any tag. “//A//@C” is any C attribute anywhere in the tree beneath an A tag in the tree.

For any descendant of A use “//A//*”. Any descendant of the root, “//*”, is also any tag in the tree. “//**/**/**” is any tag at least three levels deep in the tree. “//**//A” is an A that is not the root of the tree, while “//**//A//*” means any descendant of A so long as A is not the root.

The path “//(A|B)” means any A tag in the tree or any B tag in the tree. “//(A|B) //(C|D)” is any C or D descendant of either an A or B tag. This includes “//A//C”, “//A//D”, “//B//C” and “//B//D”. Convoluted use of this syntax is discouraged.

The path //T₁...//T_n is an ordered sequence of nodes in the tree starting with T₁ and terminating at T_n such that for all p ∈ n, T_{p+1} is a descendant of T_p.

Loose interpretation: There is likely to be relevant information in the document in places not specified in a user query. The path specifications should therefore be considered hints as to where to look.

5.1.3. A Note on Attributes

No real query using attributes on the INEX collection is believed to exist. Query authors are discouraged from using attributes simply because they can.

5.2. Path filters

At present paths can be filtered either with search strings, or numerically. In future versions, filtering based on proper nouns (e.g. Author Names), and other data types is anticipated.

5.2.4. String filtering

Documents can be filtered to only those that satisfy a given textural (CO) query in the given path (or relative to the given path).

```
about_clause : ABOUT '(' relative_path ',' co ')'
relative_path: '.' | '.' path

ABOUT: "about"
```

Relative paths are specified relative to a context path. At B in //A[B] the context path is //A. At B in //A[B]/C[D] the context path is //A. At D in //A[B]/C[D] the context path is //A/C. The relative path “.” is interpreted as “the context path”. The relative path “./p” is interpreted as “a p descendant of the context path”.

Example:

```
//article[about(./p, "information retrieval")]
```

Strict interpretation: “What ever you do, you must return article tags. Now, as a suggestion, look for //article//p elements about information retrieval.”

Loose interpretation: “What I want is most likely a whole article that mentions information retrieval in a p tag. Relevant results are not limited to this, but I’m pretty sure it’ll help you find what I want.”

5.2.5. Arithmetic filtering

Documents can also be filtered to only those that satisfy a numeric query. As with string filtering, this is specified with a relative path.

```
arithmetic_clause: relative_path arithmetic_operator NUMBER
arithmetic_operator: '>' | '<' | '=' | '>=' | '<='
```

Example:

```
//article[./pdt/yr = 2003]
```

Strict interpretation: Retrieve article elements from documents that loosely “contain the value 2003 in an //article//pdt//yr element”.

Loose interpretation: A loose interpretation could be to look at a year range (2002, 2003, and 2004). This might be useful if, for example, a workshop held in December 2003, published the formal proceedings in 2004. Alternatively, a paper published electronically in December 2002 might finally appear in print in January 2004 leading to confusion over the publication date.

The above example could also be described using string filtering

```
//article[about(./pdt/yr, 2003)]
```

however, the arithmetic syntax is preferred.

Both positive and negative numbers are supported by CO and CAS queries. The ambiguity arising from the multiple meaning of the minus (-) was discussed in section 4.6.

5.2.6. Boolean Operators

Path filters can be joined with Boolean operators AND and OR. They can also be bracketed.

```
filter: about_clause | arithmetic_clause;

filtered_clause:      filter
                    | filtered_clause AND filtered_clause
                    | filtered_clause OR filtered_clause
                    | '(' filtered_clause ')';

AND: "AND" | "and"
OR:  "OR"  | "or"
```

Examples:

```
//article[about(.., apple) and about(.., computer)]
//article[about(.., apple) or about(.., computer)]
```

Strict interpretation: The first example will return article elements from documents about apple and about computer, the second about apple or about computer (remember: these are only hints). This introduces a subtle difference in query meaning between the two queries:

```
//article[about(..//sec, apple computer)]
and
//article[about(..//sec, apple) and about(..//sec, computer)]
```

The first query asks for articles that have a section discussing ‘apple computer’. The second asks for articles that have a section discussing ‘apple’ and a section discussing ‘computer’ (even if they are not the same section). In the first query, the topics must co-occur. In the second they may co-occur.

Loose interpretation: AND is interpreted as ANDish, OR as ORish. The query contains the Boolean operators strictly as hints on how to resolve the information need. CO, SCAS and VCAS all interpret Boolean operators loosely.

5.2.7. Examples

Examples of some CAS queries are given here along with strict interpretations. Loose interpretation of each is the same “I’m sure this’ll help find what I want”.

```
//sec[about(.., mobile electronic payment system)]
```

Return sec tags where the sec tag mentions mobile electronic payment systems.

```
//*[about(.., singular value decomposition)]
```

Return elements about singular value decomposition. The retrieval engine must deduce the most appropriate element to return.

```
//article[../fm//yr >= 1998]//sec[about(..//p, "virtual reality")]
```

Return sec tags of documents about virtual reality and published on or after 1998.

```
//article[(../fm//yr = 2000 OR ../fm//yr = 1999) AND about(.., "intelligent
transportation system")]//sec[about(.., automation +vehicle)]
```

Return sec elements about vehicle automation from documents published in 1999 or 2000 that are about intelligent transportation systems.

6. Conclusions

The INEX query working-group at the INEX workshop outlined a set of requirements necessary for a query language to be used for future workshops. The language was to be similar in form to XPath, while at the same time being both severely reduced, and expanded. The language, NEXI, is defined herein and satisfies these needs.

A parser written in Flex and Bison and is included. The existing INEX topics have been translated into NEXI and checked against the parser. Only those queries using features deprecated by the working-group could not be translated – in these cases a near translation is included.

7. Acknowledgements

Richard A. O’Keefe read several drafts and commented on many aspects of this language.

References

- [1] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., & Cowan, J. (2003). Extensible markup language (XML) 1.1 W3C proposed recommendation. The World Wide Web Consortium. Available: <http://www.w3.org/TR/2003/PR-xml11-20031105/> [2003].
- [2] Clark, J., & DeRose, S. (1999). XML path language (xpath) 1.0, W3C recommendation. The World Wide Web Consortium. Available: <http://www.w3.org/TR/xpath> [2004].
- [3] Donnelly, C., & Stallman, R. (1995). Bison - the yacc-compatible parser generator. Available: <http://www.gnu.org/directory/bison.html>.
- [4] Fuhr, N., Gövert, N., Kazai, G., & Lalmas, M. (2002). INEX: Initiative for the evaluation of XML retrieval. In *Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*.
- [5] Kazai, G., Lalmas, M., & Malik, S. (2003). INEX’03 guidelines for topic development.
- [6] O’Keefe, R. A., & Trotman, A. (2003). The simplest query language that could possibly work. In *Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX)*.
- [7] Paxson, V. (1995). Flex, version 2.5, a fast scanner generator. Available: <http://www.gnu.org/directory/flex.html>.
- [8] Sigurbjörnsson, B., & Trotman, A. (2003). Queries: INEX 2003 working group report. In *Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX)*.

A1. Makefile

```
#
#   Makefile
#   -----
#   Andrew Trotman
#   University of Otago 2004
#
#   Script to build the NEXI parser
#
tokenizer : parser.tab.c lex.yy.c
            gcc lex.yy.c parser.tab.c -lm -o tokenizer

lex.yy.c : tokenizer.l parser.tab.h
            flex tokenizer.l

parser.tab.c : parser.y
              bison parser.y -d

clean :
        rm tokenizer parser.tab.h parser.tab.c lex.yy.c
```

A2. Flex script

```
%{
/*
    TOKENIZER.L
    -----
    Andrew Trotman
    University of Otago 2004

    FLEX script to tokenize INEX NEXI queries and check for syntax errors
*/
#include <stdio.h>
#include "parser.tab.h"
int c;
extern int yylval;
extern int line_number;
extern int char_number;

%}

LETTER [a-zA-Z]
DIGIT [0-9]
LETTERDIGIT [a-zA-Z0-9]
LETTERDIGITEXTRAS [a-zA-Z0-9'\-]
XML_NAMECHAR [a-zA-Z0-9_:\.-]
XML_NAME [a-zA-Z:_]

%%

" " { char_number++; }

"\r" { char_number++; }

"\n" {
    line_number++;
    char_number = 1;
    return yytext[0];
}

"about" {
    char_number += 5;
    yylval = yytext[0];
    return ABOUT;
}

"AND" {
    char_number += 3;
    yylval = yytext[0];
    return AND;
}

"and" {
    char_number += 3;
    yylval = yytext[0];
    return AND;
}

"OR" {
    char_number += 2;
    yylval = yytext[0];
    return OR;
}

"or" {
    char_number += 2;
    yylval = yytext[0];
    return OR;
}

">" {
    char_number++;
    yylval = yytext[0];
    return GREATER;
}

"<" {
    char_number++;
    yylval = yytext[0];
    return LESS;
}

"=" {
    char_number++;
    yylval = yytext[0];
    return EQUAL;
}
```

```

{LETTER}{LETTERDIGITEXTRAS}* {
    char_number += strlen(yytext);
    yylval = yytext[0];
    return ALPHANUMERIC;
}

"-"?{DIGIT}+ {
    char_number += strlen(yytext);
    yylval = yytext[0];
    return NUMBER;
}

"//" {
    char_number += 2;
    yylval = yytext[0];
    return NODE_QUALIFIER;
}

{XML_NAME}{XML_NAMECHAR}* {
    char_number += strlen(yytext);
    yylval = yytext[0];
    return XMLTAG;
}

. {
    char_number++;
    return yytext[0];
}

%%

/*
    YYWRAP()
    -----
*/
int yywrap(void)
{
    return 1;
}

```

A3. Bison script

```
%{
/*
    PARSER.Y
    -----
    Andrew Trotman
    University of Otago 2004

    BISON script to tokenize INEX NEXI queries and check for syntax errors
*/

#define YYDEBUG 1
#include <math.h>
#include <stdio.h>
#include <ctype.h>

int line_number = 1;
int char_number = 1;
extern char *yytext;

void yyerror(char *err) /* Called by yyparse on error */
{
    printf ("Line %d (char %d): %s at '%s'\n", line_number, char_number, err, yytext);
}

/*
    NOTES:
        INEX topics 10, 14, 19, 20 are not strict translations as they cannot
    be expressed (multiple specified target elements)
        INEX topic 13 is not a strict translation due to instance (au[1]) usage
*/

%}

%token NUMBER ALPHANUMERIC XMLTAG
%token ABOUT NODE_QUALIFIER
%token AND OR
%token GREATER LESS EQUAL

%left AND OR

%%/* Grammar rules and actions follow */
input: /* empty */
    | input line
    ;

line: '\n'
    | co '\n' { printf("CO Passed\n"); }
    | cas '\n' { printf("CAS Passed\n"); }
    ;

/*
    in a CAS query:
        the initial can be the terminal "/*" to specify "a descendant of"
    with INEX 2002) the final part can be an unrestricted target path (for compatibility
*/
cas: path cas_filter | path cas_filter path | path cas_filter path cas_filter;
cas_filter: '[' filtered_clause ']';
filtered_clause : filter
                | filtered_clause AND filtered_clause
                | filtered_clause OR filtered_clause
                | '(' filtered_clause ')';
filter: about_clause | arithmetic_clause;
about_clause : ABOUT '(' relative_path ',' co ')';
arithmetic_clause: relative_path arithmetic_operator NUMBER;
arithmetic_operator: GREATER | LESS | EQUAL | greater_equal | less_equal;
greater_equal: GREATER EQUAL;
less_equal: LESS EQUAL;

/*
    child has been eliminated and replaced with descendant. In the
    unlikely event child is ever needed, it can (most likely) be
    specified as those descendants enough to make the specification
    unambiguous.

    now, a PATH is either:
        "/" for root
*/
```

```

        "//A" for tag A
        "//A//B" for tag B within tag A
        "//*" for any tag
        "//A//*" for any descendant of A
        "//@A" for attribute A
        "//A//@B" for attribute B descendant of node A
*/
path: node_sequence | node_sequence attribute_node;

relative_path: '.' | '.' path;

node_sequence: node | node_sequence node;

any_node: NODE_QUALIFIER '*';

attribute_node: NODE_QUALIFIER '@' tag;

named_node: NODE_QUALIFIER tag;

tag_list: tag '|' tag | tag_list '|' tag;

tag_list_node: NODE_QUALIFIER '(' tag_list ')';

node: named_node | any_node | tag_list_node;

tag: alphanumeric | XMLTAG;

/*
    CO topics are sequences of numbers, terms and phrases
    with optional specifiers mandatory (+) and unwanted (-)
    note:
        "12" is a number
        "-12" is number
        "- 12" is don't search for number 12
        "--12" | "- -12" is don't search for number -12
        "-- 12" is an error
        "content-based" is an error
*/
co : term | co term;

term: term_restriction unrestricted_term;

term_restriction: /* empty */ | '+' | '-';

unrestricted_term: word | phrase;

/*
    A phrase is a sequence of two or more words surrounded by double quotes
*/
phrase: '"' word_list '"';

word_list: word word | word_list word;

/*
    a word is a sequence:
        of alphabetic
        of digits
        of digits preceded by a negative (-) sign (a negative number)
        alphanumerics starting with an alpha (for both ipl tags and Y2K
queries)
    As the operators are also valid search terms, a word is
    operator or a sequence of alphabetic characters
*/

word: NUMBER | alphanumeric;

alphanumeric : ALPHANUMERIC | ABOUT | AND | OR;

%%

/*
    MAIN ()
    -----
*/
int main(void)
{
    //yydebug = 1;
    yyparse();

    return 0;
}

```

A4. INEX queries 1-126

The pre-existing INEX queries have all been converted and checked against the parser. Topics 10, 14, 19 and 20 originally specified a set of target elements. This practice was banned for INEX'03 and is not supported here either. Topic 13 specifies a particular instance of an element as the target, again outlawed for INEX'03 and not supported here. Topic 44 used wildcards. As such, these 6 queries are not accurately translated.

1. //article[about(./(abs|kwd), description logics)]//fm//au
2. //ack[about(., research funded america)]
3. /**[about(./kwd, information data visualization) and about(., large information hierarchies spaces multidimensional data databases)]
4. /**[about(./at1|abs|st), experience results problems) and about(., extreme programming)]
5. //article[about(./bib1, QBIC) and about(., image retrieval)]//tig
6. //article[about(., Survey on Software Engineering) and about(./sec, programming languages)]//tig[about(., software engineering survey programming survey programming tutorial software engineering tutorial)]
7. //article[about(., Content-based retrieval of video databases)]//sec
8. //article[about(./fm//aff, ibm) and about(./bdy//sec, certificates)]
9. //article[about(./bdy//sec, nonmonotonic reasoning) and (./hdr//yr = 1999 or ./hdr//yr = 2000) and about(./tig//at1, -calendar) and about(., belief revision)]
10. /**[about(./at1|st|title), book review) and about(./st|p), machine learning adaptative algorithm probabilistic model neural network support vector machine kernel methods numerical computation)]
11. /**[about(./p, wireless) and about(./abs|kwd), security) and about(., security applications)]
12. //article[./pdt//yr = 2001 or ./pdt//yr = 2002]//bdy//sec[about(., internet search engine)]
13. //article[about(./fm//au//@sequence, additional) and about(./fm//abs, review) and about(., AR VR virtual augmented reality system)]//fm//au
14. /**[about(./fgc, Corba architecture) and about(./p, Figure Corba Architecture)]
15. //article[./fm//hdr//hdr2//pdt = 1996 or ./fm//hdr//hdr2//pdt = 1997]//bm//bib//bib1//bb[about(., hypercube mesh torus toroidal non-numerical database)]
16. //article[about(./bm//bib//bib1//bb//at1, concurrency control)]//fm//tig//at1
17. //article[about(./fm//au, -W -Bruce -Croft)]//bb[about(./au, W Bruce Croft)]
18. //article[about(., Hypertext Information Retrieval) and about(./bib//bib1//bb//at1, Hypertext Information Retrieval)]
19. /**[about(., singular value decomposition svd formula)]
20. //article[about(./at1, Concurrency Control) and about(./fm//hdr//hdr1//ti, data) and about(., Concurrency Control in real-time databases)]//sec
21. /**[about(./p|st|it|bb), recommender system recommender agent)]
22. //article[about(./bb//au//snm, Mannila) and (about(./bb//au//fnm, Heikki) or about(./bb//au//fnm, H)) and about(., Mannila)]//fm//au
23. //article[./yr = 1995 or ./yr = 1996 or ./yr = 1997 or ./yr = 1998 or ./yr = 1999) and about(./bdy, XML electronic commerce)]
24. //article[about(./au, Smith Jones) and about(./bdy, software engineering and process improvement)]
25. //article[about(./fm//hdr//hdr1//ti, IEEE MultiMedia) and about(., QoS Quality of Service)]
26. //article[about(./st, XML) and about(., data processing system)]//fm//tig//at1
27. //article[about(./at1, 1999 Reviewers List) and about(./ti, IEEE Transactions Visualization and Computer Graphics) and ./yr = 2000]//reviewer//name
28. //article[about(./sec1//title, Special Feature) and about(./ti, IEEE Micro)]//at1
29. /**[about(./at1, image retrieval) and about(., image retrieval colour shape texture)]
30. //article[./yr >= 1996 and about(., parallelism)]//au

31. computational biology
32. semantic web
33. software patents
34. Efficient database search structures and techniques
35. Parallel query optimization
36. Heat dissipation of microcomputer chips
37. Temporal database queries and query processing
38. multidimensional indices
39. Video on demand
40. Content-based retrieval
41. Y2K spending
42. Decryption of the Enigma code
43. approximate string matching algorithm
44. internet society communication netizen social sociology web usenet mail network culture
45. augmented reality and medicine
46. Firewalls in internet security
47. concurrency control semantic transaction management application performance benefit
48. active database rule specification
49. Query relaxation approximate and intelligent query answering
50. XML editors or parsers
51. Text Data Mining
52. History of Computing of USSR
53. information retrieval xml
54. knowledge building acquisition and sharing
55. Digital Divide city planning neighbourhood planning
56. open hypermedia systems and agents
57. public key cryptography RSA EC DSA algebraic number field
58. Location management scheme
59. schema integration methods
60. Internet speed
61. //article[about(.,clustering +distributed) and about(./sec,java)]
62. //article[about(.,security +biometrics) AND about(./sec,"facial recognition")]
63. //article[about(.,"digital library") AND about(./p, +authorization + "access control" +security)]
64. //article[about(., hollerith)]//sec[about(., DEHOMAG)]
65. //article[./fm//yr > 1998 AND about(., "image retrieval")]
66. //article[./fm//yr < 2000]//sec[about(., "search engines")]
67. //article//fm[about(./((tig|abs), +software +architecture) and about(., -distributed -Web)]
68. //article[about(., +Smalltalk) or about(., +Lisp) or about(.,+Erlang) or about(., +Java)]//bdy//sec[about(., +"garbage collection" +algorithm)]
69. //article//bdy//sec[about(./st,"information retrieval")]
70. //article[about(./fm//abs, "information retrieval" "digital libraries")]
71. //article[about(.,formal methods verify correctness aviation systems)]//bdy//*[about(.,case study application model checking theorem proving)]
72. //article[about(./fm//au//aff,United States of America)]//bdy//*[about(.,weather forecasting systems)]
73. //article[about(./st,+comparison) and about(./bib,"machine learning")]
74. //article[about(., video streaming applications)]//sec[about(., media stream synchronization) OR about(., stream delivery protocol)]

75. //article[about(., Petri net) AND about(./sec, formal definition) AND about(./sec, algorithm efficiency computation approximation)]

76. //article[(./fm//yr = 2000 OR ./fm//yr = 1999) AND about(., "intelligent transportation system")]//sec[about(.,automation +vehicle)]

77. //article[about(./sec,"reverse engineering")]//sec[about(., legal) OR about(.,legislation)]

78. //vt[about(., "Information Retrieval" student)]

79. //article[about(.,XML) AND about(.,database)]

80. //article//bdy//sec[about(., "clock synchronization" "distributed systems")]

81. //article[about(./p,"multi concurrency control") AND about(./p, algorithm) AND about(./fm//at1, databases)]

82. //article[about(.,handwriting recognition) AND about(./fm//au,kim)]

83. //article//fm//abs[about(., "data mining" "frequent itemset")]

84. //p[about(.,overview "distributed query processing" join)]

85. //article[./fm//yr >= 1998 and ./fig//no > 9]//sec[about(./p,VR "virtual reality" "virtual environment" cyberspace "augmented reality")]

86. //sec[about(.,mobile electronic payment system)]

87. //article[(./fm//yr = 1998 OR ./fm//yr = 1999 OR ./fm//yr = 2000 OR ./fm//yr = 2001 OR ./fm//yr = 2002) AND about(., "support vector machines")]

88. //article[(./fm//yr = 1998 OR ./fm//yr = 1999 OR ./fm//yr = 2000 OR ./fm//yr = 2001) AND about(., "web crawler")]

89. //article[about(./bdy,clustering "vector quantization" +fuzzy +k-means +c-means -SOFM -SOM)]//bm//bb[about(., "vector quantization" +fuzzy clustering +k-means +c-means) AND about(./pdt,1999) AND about(./au//snm, -kohonen)]

90. //article[about(./sec,+trust authentication "electronic commerce" e-commerce e-business marketplace)]//abs[about(., trust authentication)]

91. Internet traffic

92. "query tightening" "narrow the search" "incremental query answering"

93. "Charles Babbage" -institute -inst

94. "hyperlink analysis" +"topic distillation"

95. +"face recognition" approach

96. +"software cost estimation"

97. Converting Fortran source code

98. "Information Exchange" +XML "Information Integration"

99. perl features

100. +association +mining +rule +medical

101. +"t test" +information

102. distributed storage systems for grid computing

103. UML formal logic

104. Toy Story

105. +categorization "textual document" learning evaluation

106. Content protection schemes

107. "artificial intelligence" AI practical application industry "real world"

108. ontology ontologies overview "how to" practical example

109. "CPU cooling" "cooling fan design" "heatsink design" "heat dissipation" airflow casing

110. "stream delivery" "stream synchronization" audio video streaming applications

111. "natural language processing" -"programming language" -"modeling language" +"human language"

112. +"Cascading Style Sheets" -"Content Scrambling System"

113. "Markov models" "user behaviour"

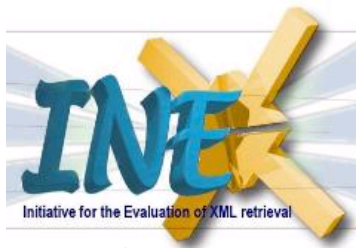
114. +women "history of computing"

115. +"IP telephony" +challenges

116. "computer assisted art" "computer generated art"

117. Patricia Tries

118. "shared nothing" database
119. Optimizing joins in relational databases
120. information retrieval models
121. Real Time Operating Systems
122. Lossy Compression Algorithm
123. multidimensional index "nearest neighbour search"
124. application algorithm +clustering +k-means +c-means "vector quantization"
"speech compression" "image compression" "video compression"
125. +wearable ubiquitous mobile computing devices
126. Open standards for digital video in distance learning



INEX 2004 Retrieval Task and Result Submission Specification

01 June 2004

Retrieval Task

The retrieval task to be performed by the participating groups of INEX 2004 is defined as the ad-hoc retrieval of XML documents. 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 two sub-tasks:

CO: Content-oriented XML retrieval using content-only (CO) queries. As described in the INEX 2004 Topic Development Guide, CO queries are requests that ignore the document structure and contain only content related conditions, e.g. only specify what a document/component 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. In this task, it is left to the retrieval system to identify the most appropriate XML elements to return to the user. These elements are components that are most specific and most exhaustive with respect to the topic of request. Most specific here means that the component is highly focused on the topic, while exhaustive reflects that the topic is exhaustively discussed within the component.

VCAS: Content-oriented XML retrieval based on content-and-structure (CAS) queries, where the structural constraints of a query can be treated as *vague* conditions. CAS queries are topic statements, which contain explicit references to the XML structure, and explicitly specify the contexts of the user's interest (e.g. target elements) and/or the contexts of certain search concepts (e.g. containment conditions). Specifying an information need is not an easy task, in particular for semi-structured data with a wide variety of tag names. Although users may think they have a clear idea of the structural properties of the collection, there are likely to be aspects to which they are unaware. The idea behind the VCAS sub-task is to allow the evaluation of XML retrieval systems that aim to implement approaches, where not only the content conditions within a user query are treated with uncertainty but also the expressed structural conditions. These systems aim to return components that contain the information sought after by the user even if the result elements do not exactly meet the structural conditions expressed in the query. The path specifications should therefore be considered hints as to where to look.

Note: INEX 2003 had an additional task, the SCAS sub-task. There, the structural constraints of a CAS query had to be strictly matched. INEX can provide relevance assessments to participants interested in the SCAS sub-task. The relevance assessments will be based on the VCAS criterion. The obtained set of assessments will be filtered to derive the relevance assessments for SCAS. This was the approach adopted last year, and was sufficient to indicate how effective were the approaches. The filtered set however does not correspond to assessments that fully satisfy the SCAS criterion, and should be considered as an "approximate set". If enough participants are interested and depending on time and resources, additional assessments could be performed to derive the "correct" relevance assessments for SCAS based on the approximate set.

Result Submission

For each sub-task up to 3 runs may be submitted. The results of one run must be contained in one submission file (e.g. up to 6 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 CO sub-task only submit the search results obtained for the CO topics).

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 overall submission format is defined in the following DTD:

```
<!ELEMENT inex-submission      (description, topic+)>
<!ATTLIST inex-submission
  participant-id      CDATA      #REQUIRED
  run-id             CDATA      #REQUIRED
  task      ( CO | VCAS )      #REQUIRED
  query      (automatic | manual)  #REQUIRED
>
<!ELEMENT description      (#PCDATA)>
<!ELEMENT topic            (result*)>
<!ATTLIST topic
  topic-id           CDATA      #REQUIRED
>
<!ELEMENT result          (file, path, rank?, rsv?)>
<!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:2004/inex04/servlet/ShowParticipants>), 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 or VCAS), and the identification 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 each topic a maximum of **1500** result elements may be included. 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. Before detailing the various elements of the above DTD, here is a sample submission file:

```
<inex-submission participant-id="12" run-id="VSM_Aggr_06" task="CO"
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>
  <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>
```

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 only article files (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>
  <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:2004/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

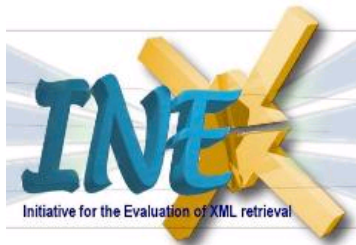
To submit a run, please use the following link:

http://inex.is.informatik.uni-duisburg.de:2004/cgi-bin/inex?mode=browse_submissions

This online submission tool will be available soon.

June 1, 2004.

Mounia Lalmas and Saadia Malik



INEX 2004 Relevance Assessment Guide

1. Introduction

During the retrieval runs, participating organisations evaluated the 74 INEX 2004 topics (40 content-only (CO) and 34 content-and-structure (CAS) queries) against the IEEE Computer Society document collection and produced a list (or set) of document components (XML elements¹) 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 2004 test collection. This requires first a definition of relevance for XML retrieval (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 dimensions: exhaustivity and specificity

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.

Exhaustivity is measured on the following 4-point scale:

Not exhaustive (E0): the document component does not discuss the topic of request at all.

Marginally exhaustive (E1): the document component discusses only few aspects of the topic of request.

Fairly exhaustive (E2): the document component discusses many aspects of the topic of request.

Highly exhaustive (E3): the document component discusses most or all aspects of the topic of request.

Specificity is assessed on the following 4-point scale:

Not specific (S0): the topic of request is not a theme of the document component.

Marginally specific (S1): the topic of request is a minor theme of the document component (i.e. the component focuses on other, non-relevant topic(s), but contains some relevant information).

Fairly specific (S2): the topic of request is a major theme of the document component (i.e. the component contains mostly relevant content and only some irrelevant content).

Highly specific (S3): the topic of request is the only theme of the document component.

Although the two dimensions are largely independent of each other, a not-exhaustive (E0) component can only be not specific (S0) and vice versa. Other than this rule, a component may be assigned any other combination of exhaustivity and specificity, i.e. E3S3, E3S2, E3S1, E2S3, E2S2, E2S1, E1S3, E1S2, and E1S1. For example, a component assessed as E1S1 is one that contains only marginally exhaustive relevant information (E1) where this relevant content is only a minor theme of the component, i.e. most of the content is irrelevant to the topic of request (S1).


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 between 500 and 1,500 document components of 500 - 510 articles, where a component may be a title, paragraph, subsection, section, or whole article, etc.

Traditionally, in evaluation initiatives for information retrieval, like TREC, 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, you will then need to provide assessment –(both with regards to exhaustivity and specificity) for the found relevant component, for all its ascendant elements until you reach the article component, and for all its descendant elements 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, which encapsulates all text fragments relevant to the topic, as highly exhaustive and fairly specific (E3S2, denoted by , see Table 1). Given this single assessment, it would not be possible to deduce the exhaustivity and specificity levels of the ascending or descending elements. For example, both `bdy` and `article` may be judged either fairly (S2) or marginally specific (S1) depending on the volume of additional, irrelevant 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 or specificity levels of its sub-components; i.e., any of the `ss1`, `ss2` subsections, and `p` paragraphs (etc.) may be highly (E3), fairly (E2) or only marginally (E1) exhaustive, and could be highly (S3), fairly (S2) or only marginally (S1) specific, or could even be irrelevant (E0S0). For example, one of the paragraphs of the first `ss2` element may be irrelevant (E0S0), while the other may be fairly exhaustive and highly specific (E2S3).


```
[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 (E3) element will always be highly exhaustive (E3), since the child element already discusses all or most aspects of the topic. Another rule for the exhaustivity dimension is that a component whose child elements are all not exhaustive (E0)

will also be not exhaustive (E0). A rule regarding specificity is that the parent of an element whose specificity degree is greater than 0, must also have a specificity level greater than 0, but less or equal to the maximum S value of all its child elements. For instance, suppose that a parent element has a small child element with S1 and a large child element with S2, then the S value of that parent can only be either 1 or 2. However, besides these general rules, no specific rules exist that would automate all the assessment of ascendant and descendant elements of relevant components. Therefore, **you will need to explicitly judge all elements that contain relevant information.** This is the only way to ensure both comprehensive and consistent relevance assessments.

4. How to judge

To assess the exhaustivity and specificity of document components, we recommend a three-phase approach.

- During the first phase, 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 will assist you in this task by highlighting keywords within the article (see Section 5).
- In the second phase, you should assess the exhaustivity and specificity of the relevant components (i.e. identified in the first phase), and that of their ascendant and descendant XML elements.
- To ensure comprehensive assessments, in the third phase, you should assess the exhaustivity and specificity of the descendant XML elements of all elements that have been assessed as relevant during the second phase.

The on-line assessment system (see Section 5) will identify for you all elements that have to be assessed for phases 2 and 3.

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, topic title and keywords. 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. The keywords should be used strictly as a source of possibly relevant cue words and hence only as a means of aiding your assessment. You should **not rely only on the presence or absence of these keywords** in document components to judge their relevance. It may be that a component contains some or maybe all the keywords, but is irrelevant to the topic of the request. Also, there may be components that contain none of the keywords yet are relevant to the topic. The same applies to the terms listed within the topic title!

In the case of CAS topics, the topic titles contain structural constraints in the form of XPath expressions. Since these structural conditions are there to provide hints for the search engines, they should be ignored during your assessment. This means that you should assess the elements returned for a CAS topic as whether they satisfy your information need (as specified by the topic) **with respect to the content criterion only**. Therefore, you should not assess an element as “irrelevant” only because the structural condition is not satisfied.

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:

<http://inex.lip6.fr/2004/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 recent "gecko" browsers: we highly recommend you to use either

- **Mozilla** (version 1.7 or up) at <http://www.mozilla.org/products/firefox/>.
- **Firefox** (version 0.9.1 or 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**.

X-Rai uses MathML in order to render mathematical formulas. Some specific fonts have to be downloaded if you want a perfect rendering of mathematical formulas. More details can be found at the following address: <http://www.mozilla.org/projects/mathml/fonts/>.





Before detailing the assessment system, first we define describe the graphical scheme employed to represent the possible combinations of relevance values.

5.1. Relevance values

In the on-line assessment system, the following scheme is used for representing assessment values (see also Table 1):

- **Exhaustivity** level is displayed in different shades of blue, where the darker the blue, the more exhaustive the element.
- **Specificity** level is represented by two overlapping discs: the filled in disc denoting the relevant content and the white disc representing the irrelevant content. A single blue disc represents a highly specific (S3) element; a blue disc in front of the white disc denotes a fairly specific (S2) element; a blue disc behind the white disc corresponds to the marginally specific (S1) degree; and finally a single white disc represents a not-specific (S0) component.

The tables below show the different icons used to indicate the relevance value of an XML element.

	Element is not assessed
	Element is to be assessed
	Element is irrelevant (E0S0)
	Element is inconsistent












Exhaustivity Specificity	Highly exhaustive (E3)	Fairly exhaustive (E2)	Marginally exhaustive (E1)
Highly specific (S3)			
Fairly specific (S2)			
Marginally specific (S1)			

Table 1. Icons used to indicate relevance values

Note that all icons except the  icon can be used by assessors to specify the relevance value (the exhaustivity and specificity level) of an element. The  icon is used by the on-line assessment system only to mark components that are in an inconsistent state.

5.2. Home page

After logging in, you will be presented with the Home page (see Figure 2a) 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.

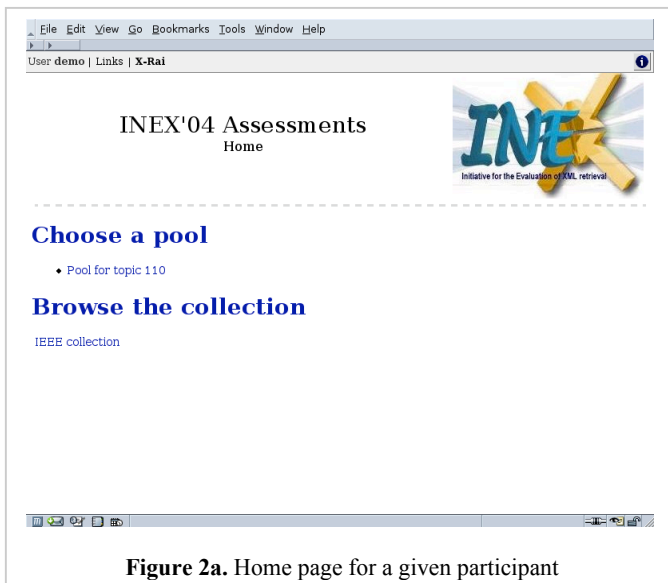


Figure 2a. Home page for a given participant

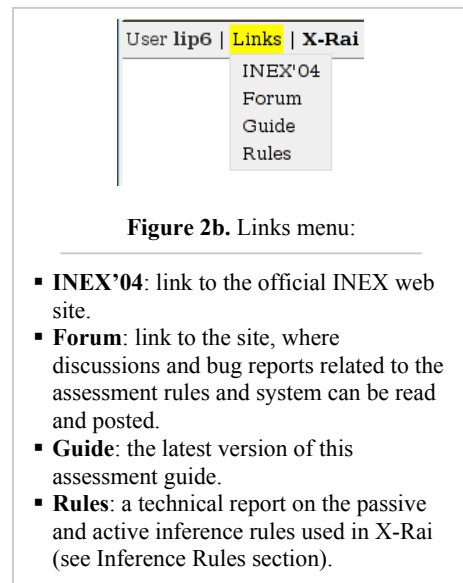












Figure 2b. Links menu:

- **INEX'04**: link to the official INEX web site.
- **Forum**: link to the site, where discussions and bug reports related to the assessment rules and system can be read and posted.
- **Guide**: the latest version of this assessment guide.
- **Rules**: a technical report on the passive and active inference rules used in X-Rai (see Inference Rules section).

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 2a and “lip6” in Figure 2b),
 2. A list of menu items, which can be accessed by holding the mouse over the menu label (e.g. “Links” in Figure 2b.),
 3. The location within X-Rai, where each location step is a hyperlink (in Figure 2a, 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.
 -  The tree view of the article (only available in article view).
 -  The current list of bookmarks (only available in article view).
- The main window.
- An optional status bar (see Figure 3a and 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 statistics on the current view for each relevance value, e.g. how many elements have been assessed as highly exhaustive and highly specific (), as highly exhaustive and fairly specific (), etc; as irrelevant (); and how many elements remain to be assessed (). Only when no more elements remain to be assessed is the assessment for that view (pool / sub-collection / article) complete.

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 **shift + left / up / down**. 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:

1. Pool view,
2. Sub-collection view, and
3. Article view

5.3. Pool view

Clicking on a topic ID will display the Pool main page for that topic (see Figure 3a).

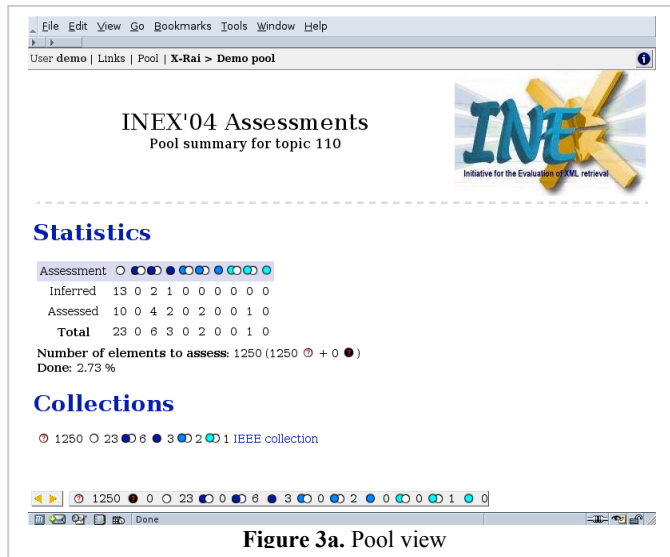


Figure 3a. Pool view

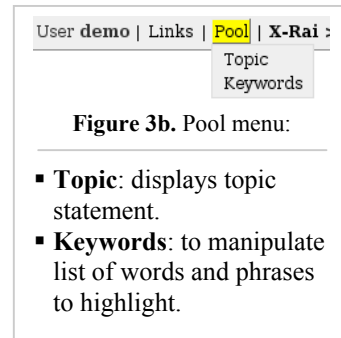


Figure 3b. Pool menu:

- **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’04 (ad hoc task) there is only one such collection, the IEEE collection.

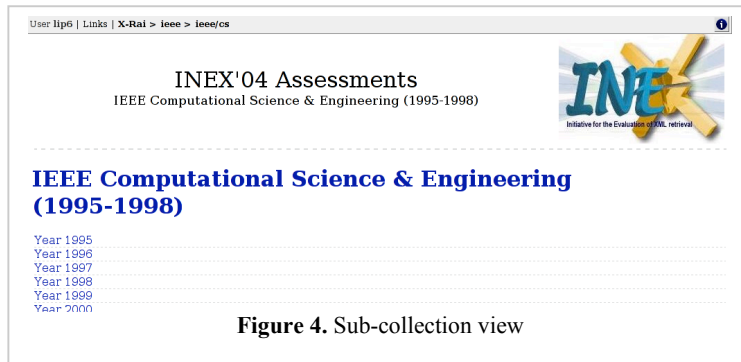
The left or right arrows on the status bar move the focus to the previous or next collection, where there is at least one element to assess (since there is only one collection, the focus will remain).

Clicking the hyperlink of “**IEEE collections**” will take you into the sub-collection view.

5.4. Sub-collection view

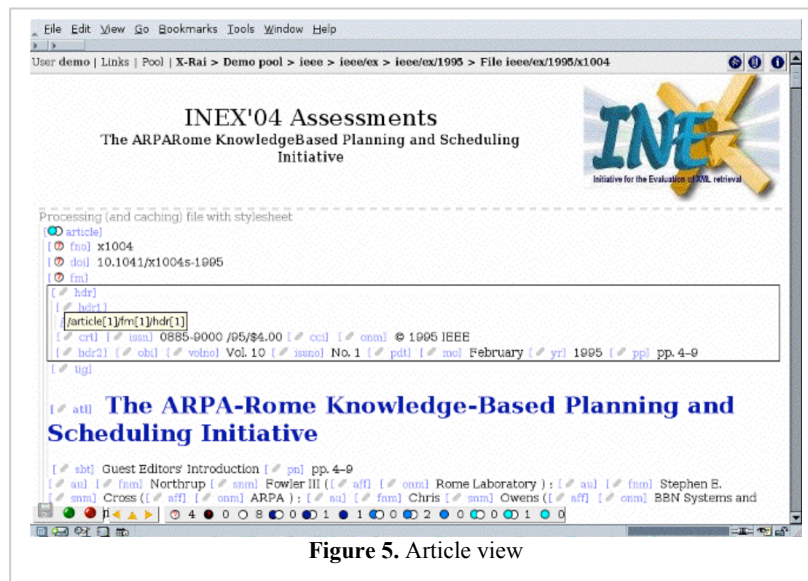
The sub-collection views allow you to browse the different sub-collections within the IEEE collection, *i.e.* volumes, years within a given volume (see Figure 4), the collection of articles within a given volume and year. Note that this view will show all elements within a sub-collection, *i.e.* all articles within a given volume and year, and not only the ones that need to be assessed.

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 element to assess. You can also directly click on a link to a sub-collection.



5.5. Article view

It is in this article view that elements can be assessed. The article view (see Figure 5) displays all components of an article, whether these elements are to be assessed or not. In addition, the article view shows every XML tag in the article while keeping an eye-friendly view of the article. XML tags are displayed between brackets, in light blue font. For each component the currently assigned (or inferred) relevance values are displayed in front of the XML tag name.





For instance, an `abs` element, which has been assessed as highly exhaustive and highly specific (E3S3), is displayed with the following XML tag syntax:




[● abs]




To make an assessment, first hold the mouse over an XML tag name. The cursor will turn into a cross shape. You can then:

- Left-click to display the assessment panel for the element. The assessment panel has three components: the XPath of the selected element (first line), the current assessment value (second line), and the set of 11 icons (reflecting all possible assignments as shown in Table 1). Forbidden assessments (e.g. assessing a parent element as not relevant where one of its child elements is relevant) are displayed in a grey box. To assess the current element, click on the icon with the corresponding relevance value. To hide the panel, click anywhere else in the panel.

- It is possible to assess groups of elements. Control-click to select the element or control-double-click to select the sibling elements in the same “state” (i.e. elements which are assessed or un-assessed). Then click on the green button  or press the key **shift-G** to display an assessment panel for the selected elements. Click on the red button  or press the key **control-shift-G** to clear the current selection.
- Right-click to display the navigation panel (see figure below). Depending on the element you clicked on, there might be up to three arrows. Click on the left (or right) arrow to access the previous (or next) sibling element in the article tree. Click on the up arrow to access the parent. If necessary, the window scrolls up or down to make the element tag visible. The element tag is then highlighted in red for a brief moment.



The  icon at the bottom centre of the navigation panel can be used to add the currently selected element to the list of bookmarks. Click on  to remove the element from the bookmarks. To display the list of bookmarks, click on the clip icon  or press the key **B**. The bookmarks are ordered with respect to the article they occur in. Click on a bookmark to highlight the element.

It is also possible to use a document-tree view in order to navigate into the article. Click on the icon  or press the key **T**. A panel then appears with the document tree view. Click on  to expand a sub-tree or on  to collapse a sub-tree. Click on any element to highlight it in the article view.



5.6. Saving your assessments

Contrary to last year, 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.

5.7. Inference rules

This year, the assessment system makes use of two types of inference mechanisms to ensure exhaustive and consistent assessments: we refer to these as passive and active inferences. The passive type simply identifies new elements to be assessed based on those already assessed. For example, for any relevant element (e.g. any component assessed other than “irrelevant”), the relevance of its child elements must be assessed, even if these were not part of the original assessment pool (i.e. have not been retrieved). With the application of the passive inference rules, these need-to-be-assessed components will be marked with the  icon. Unlike the passive rules, the active inference rules are able to derive the relevance value of some elements. These inferred relevance values will be marked using a red border. For example,  denotes “inferred as not relevant”, which is (for example) assigned to a component if all its child elements have been assessed as “not relevant”.

August 2004

Gabriella Kazai, Mounia Lalmas, and Benjamin Piwowarski

Evaluation Metrics 2004

Arjen P. de Vries
CWI
arjen@acm.org

Gabriella Kazai
Queen Mary University of London
gabs@dcs.qmul.ac.uk

Mounia Lalmas
Queen Mary University of London
mounia@dcs.qmul.ac.uk

1. INTRODUCTION

The ‘official’ INEX 2004 rankings have been based on the results of the *inex-2002* metric (implemented in the `inex_eval` script), where a number of new quantisation functions have been introduced besides the original strict and generalised quantisations. The main evaluation index corresponds to a combination (macro-average) of the *inex-2002* evaluation scores on each of these quantisations. We also report the (*set-based*) *overlap indicator* that characterises a run by the percentage of overlapping items in the submission.

2. THE INEX-2002 METRIC

The *inex-2002* metric computes the *recall* measure [4] over returned document components, and computes the probability $P(\text{rel}|\text{retr})$ that a component viewed by the user is relevant:

$$P(\text{rel}|\text{retr})(x) := \frac{x \cdot n}{x \cdot n + \text{esl}_{x \cdot n}} \quad (1)$$

where $\text{esl}_{x \cdot n}$ denotes the *expected search length* [1], i.e. the expected number of non-relevant elements retrieved until an arbitrary recall point x is reached, and n is the total number of relevant components with respect to a given topic.

To apply the above metric, the two relevance dimensions (i.e. exhaustivity and specificity) are first mapped to a single relevance scale by employing a quantisation function, $\mathbf{f}_{\text{quant}}(e, s): ES \rightarrow [0, 1]$, where ES denotes the set of possible assessment pairs (e, s) :

$$ES = \{(0, 0), (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)\}$$

Two quantisation functions are used: $\mathbf{f}_{\text{strict}}$ (Equation 2) and \mathbf{f}_{gen} (Equation 3). The former is used to evaluate retrieval methods with respect to their capability of retrieving highly exhaustive and highly specific document components. The generalised function credits document components according to their *degree of* relevance.

$$\mathbf{f}_{\text{strict}}(e, s) := \begin{cases} 1 & \text{if } e = 3 \text{ and } s = 3, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

$$\mathbf{f}_{\text{gen}}(e, s) := \begin{cases} 1 & \text{if } (e, s) = (3, 3), \\ 0.75 & \text{if } (e, s) \in \{(2, 3), (3, \{2, 1\})\}, \\ 0.5 & \text{if } (e, s) \in \{(1, 3), (2, \{2, 1\})\}, \\ 0.25 & \text{if } (e, s) \in \{(1, 2), (1, 1)\}, \\ 0 & \text{if } (e, s) = (0, 0). \end{cases} \quad (3)$$

3. ADDITIONAL QUANTISATION FUNCTIONS

Kazai et al. [3] defined a so-called *specificity-oriented generalised* quantisation function, that better reflects the evaluation criterion for XML retrieval defined in INEX, according to which specificity plays a more dominant role than exhaustivity:

$$\mathbf{f}_{\text{sog}}(e, s) := \begin{cases} 1 & \text{if } (e, s) = (3, 3), \\ 0.9 & \text{if } (e, s) = (2, 3), \\ 0.75 & \text{if } (e, s) \in \{(1, 3), (3, 2)\}, \\ 0.5 & \text{if } (e, s) = (2, 2), \\ 0.25 & \text{if } (e, s) \in \{(1, 2), (3, 1)\}, \\ 0.1 & \text{if } (e, s) \in \{(2, 1), (1, 1)\}, \\ 0 & \text{if } (e, s) = (0, 0). \end{cases} \quad (4)$$

Following the conclusions of the metrics workgroup at INEX 2003 [2], two other classes of ‘strict’ quantisation functions have been defined: specificity-oriented and exhaustivity-oriented functions.

The specificity-oriented functions apply strict quantisation with respect to the specificity dimension only, while allow to consider different degrees of exhaustivity. They aim to evaluate systems according to their ability to retrieve the most specific relevant components, where the exhaustivity of the component may vary from marginally and fairly exhaustive to highly exhaustive (Equation 5) or only from fairly to highly exhaustive (Equation 6).

$$\mathbf{f}_{\text{s3.e321}}(e, s) := \begin{cases} 1 & \text{if } e \in \{3, 2, 1\} \text{ and } s = 3, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

$$\mathbf{f}_{\text{s3.e32}}(e, s) := \begin{cases} 1 & \text{if } e \in \{3, 2\} \text{ and } s = 3, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Similarly to the specificity-oriented functions, exhaustivity-oriented quantisation functions have been defined as well: Equations 7 and 8. Note, however, that these exhaustivity-oriented functions suffer from the same overlap problem as the generalised quantisation functions.

$$\mathbf{f}_{e3_s321}(e, s) := \begin{cases} 1 & \text{if } e = 3 \text{ and } s \in \{3, 2, 1\}, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

$$\mathbf{f}_{e3_s32}(e, s) := \begin{cases} 1 & \text{if } e = 3 \text{ and } s \in \{3, 2\}, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

4. OVERLAP INDICATOR

The (*set-based*) *overlap indicator* is defined for a result list R as follows:

$$\frac{|\{p \in R | \exists q \in R \wedge p \neq q \wedge \text{overlap}(p, q)\}|}{|R|} \quad (9)$$

5. REFERENCES

- [1] W. Cooper. Expected search length: A single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. *American Documentation*, 19(1):30–41, 1968.
- [2] G. Kazai. 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, December 2003*, pages 184–190, Mar. 2004.
- [3] G. Kazai, M. Lalmas, and A. 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*, pages 72–79, 2004.
- [4] V. Raghavan, P. Bollmann, and G. Jung. A critical investigation of recall and precision. *ACM Transactions on Information Systems*, 7(3):205–229, 1989.

Guidelines for Topic Development in Heterogeneous Collections

Virginia Dignum, Roelof van Zwol
Utrecht University
{virginia, roelof}@cs.uu.nl

July 12, 2004

1 Introduction

The current INEX collection is based on a single document structure (DTD) [1]. In practical environments, such a restriction will hold in rare cases only. Instead, most XML collections will comprise documents from different sources, and thus with different DTDs. Also, there will be distributed systems (federations or peer-to-peer systems), where each node manages a different type of collection. The heterogeneous track (in the remainder of this document referred to as the het track) comes forth from the realization that an information seeker is interested in semantically meaningful answers irrespectively of the structure of the documents. As in the ad-hoc track, both content-only (CO) and content-and-structure (CAS) should be possible in the het track.

CO queries are DTD-independent and as such, running CO queries defined for the ad-hoc track in the het collection should provide reasonable answers. CAS topics are more interesting and challenging. For CAS queries, there is the problem of mapping structural conditions from one DTD onto other (possibly unknown) DTDs. Methods from federated databases could be applied here, where schema mappings between the different DTDs are defined manually. However, for a larger number of DTDs, automatic methods must be developed, e.g. based on ontologies. Some work has been done on the derivation of DTDs from ontologies (e.g. [2]). Although interesting, such work is actually the inverse of what the het-track requires. The ultimate aim of this track is (1) to develop a generic ontology for scientific publications information, and (2) automatically generate the equivalence between that ontology and different document DTDs.

The current aim of this document is to present initial ideas concerning the development and assessment of CAS topics for the het track. Different approaches are described from basic single constraint to complex topics which assessment will require path transformation methods and probabilistic approaches. Since

this year is the first time that we will assess heterogeneous collections, we think that it is more important to get a good feeling of the challenges and problems related to heterogeneous collections than to achieve the best retrieval results possible.

This document is organized as follows. In the next section, we introduced three different types of CAS topics for the het track (plus a brief reference to CO topics) including example topics. In section 4 we present the format to be used in the specification of het topics.

2 Topic Creation

At this early stage of the Het track, the focus of topic creation is, in our opinion, not so much the retrieval performance for the topic, but to gain experience with retrieval in heterogenous collections.

Usually, the effectiveness of a retrieval strategy is judged in terms of performance over a large set of topics. In the current stage of het track, an effective retrieval strategy is one that is able to query several collections.

3 Het Topic Types

Given the objectives for het retrieval strategies described in the previous sections, we propose to use different topics types, as follows:

- CO: is not challenging for the het track. Since CO queries do not take structural information into account, however, any CO query used in the ad-hoc track can be used in the het track and give similar results (because the test collection used for the ad-hoc track is part of the het track).
- BCAS (Basic Content and Structure Topics): This type of topics focuses on the combination of singular structural constraints with a content-based constraint. The aim is synonym matches for structural constraints.
- CCAS (Complex Content and Structure Topics): are the het track equivalent of the CAS topics of the ad-hoc track, specified used the NEXI language. The aim is to enable transformations and partial mappings of the topic path upon the different collections in the het track, without loosing the IR component of the topic.
- ECCAS (Extented Content and Structure Topics): extend CCAS to enable the specification of the correctness path transformation and mapping probabilities.

3.1 Content-only

The treatment of CO queries, as it does not depend on the structure of the document collection, is exactly the same as in the ad-hoc track. Since the collection used in the ad-hoc track is one of the collections available for the het track, the aim is that all documents assessed relevant in the ad-hoc track for a given CO topic must be as well retrieved by retrieval tools to be used in the het track. As such, we will not further discuss CO queries but, at least for this year, will use the CO topics available for the ad-hoc track.

3.2 Basic CAS

The motivation for BCAS is based on the assumption that each retrieval system in the Het track should be able to retrieve related XML fragments in semantically similar document collections. In its simplest form the retrieval system should be capable to successfully match an atomic structural constraint upon semantically similar XML fragments. Figure 1 shows three XML fragments, and an example topic for the BCAS sub track. It illustrates how a single atomic structural constraint and a single content-based constraint can be expanded, and matched upon several semantically similar structures. We think its worthwhile to investigate the systems ability to find semantically and textually relevant pieces of information that match a single structural constraint.

<pre><author>William Tell</author></pre>
(a) XML fragment 1
<pre><writer fname='William' lname='Tell' /></pre>
(b) XML fragment 2
<pre><author> <firstname>William</firstname> <lastname>Tell</lastname> </author></pre>
(c) XML fragment 3
<pre>//author[about(., 'William Tell')]</pre>

Figure 1: XML document fragments

The format of a BCAS topic can thus be described using NEXI syntax as follows:

```
//struct_constraint[about(.,content_constraint)],
```

where *struct_constraint* is an atomic structural reference and *content_constraint* a textual reference. At this stage, we think that structural constraints should be specified using DTD concepts as in the ad-hoc collection. A possible extension will be to form topics using other DTDs in the het-collection.

Examples are:

- `//author[about(.,'JohnSmith')]`
- `//title[about(.,'InformationRetrieval')]`
- not ad-hoc DTD: `//writer[about(.,'JohnSmith')]`

3.3 Complex CAS

The CCAS sub-track makes use of the system's ability to resolve combined (complex) structural constraints, that allow a wide range of path transformations to be incorporated and matched upon the XML fragments that are considered relevant for a topic. This sub-track can be compared to CAS topics in the ad-hoc track.

Query format: CCAS titles are 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; the arithmetic operators `<`, `<=`, `>`, and `>=` for numbers; and the connectives and and or (cf. [1]).

As in the BCAS sub-track, we think that in a first step, elements of the DTD used currently for the ad-hoc track should be used to specify structural references. Extensions to this track will allow the use of other DTDs to specify structural constraints.

3.4 Extended Complex CAS

With this sub-track, we mean to explore the possibilities to describe the uncertainty of the information seeker irt the specific structure constraint sought. We assume here that the user is able to express the probability of the likelihood of a given structural constraint.

For example, a seeker might be looking for experts in information retrieval. She expects that authors of publications on IR, with a probability of 70% (this means that there is a 30% probability that the information she is looking for is represented using a semantically different tag). To determine that a publication is about IR, she expects that in 50% of the cases, the title of the publication will contain IR In this case, a ECCAS query would look like:

`//author(0.8)[about(title(0.5),'InformationRetrieval')]`

Query format: ECCAS queries are specified as XPath expressions of the form:

A(P)[B]
or

A(P)[B]C(P)[D]

where (P) is the probability value given by the user to a structural constraint.

4 Topic Format

Title Short description of information need.

- BCAS
- CCAS
- ECCAS

Content Description (as in ad-hoc track) natural language definition of the content needed

Structure Description (!new for Het track!) natural language definition of the structure characteristics been looked for (e.g. The user is looking for authors that wrote about XML retrieval: Author is a tag identifying the person, persons, or institutions that have written/compiled the document. It is usually part of the front matter/editorial section of a document)

Narrative (as in ad-hoc track) detailed explanation of the information needs and the description of what makes a document relevant or not

Keywords (as in ad-hoc track) a set of comma separated scan terms.

5 Het Topic Assessment

In the following, some thoughts concerning the assessment of het-topics.

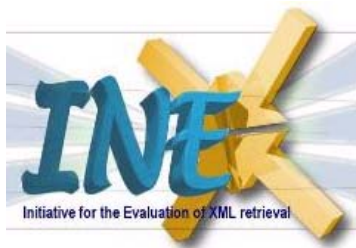
1. Structure is described using standard language (e.g. the one used in the ad-hoc track)
2. structure is described in terms of any one of the DTDs in the collection
3. description of the structure includes as many as possible synonyms of terms
4. natural language description of content (as is the case for content description, in the ad-hoc track)

The choice for either of these options has consequences for the retrieval performance of the tool (precision and recall):

- if equivalence classes for DTDs are available, then in the first three case above, a possible solution is to 'translate' the query into as many DTDs as possible and run all those sub-queries on documents of the given DTD. For example, if it is known that 'author', 'name' and 'writer' are all possible classes names to indicate the author of a document, a query on 'author' should be able to return as relevant, documents that contain information on 'name' and 'writer' as well.
- if equivalence classes are not available, in the first two cases above, queries will only hit and return answers from documents that follow the DTD used in the query. Taking, the same example, now a query on 'author' only identifies as relevant documents which DTD contains class 'author' and ignores documents which describe 'writer' or 'name'.
- (semi) automatic learning of DTDs.

References

- [1] INEX04 Guidelines for Topic Development, <http://inex.is.informatik.uni-duisburg.de:2004/internal/pdf/INEX04TopicDevGuide.pdf>
- [2] Michael Erdmann, Rudi Studer: How to structure and access XML documents with ontologies. *Data & Knowledge Engineering*, 36(3): 317-335 2001



Track Result Submission Specification

Result Submission

For at least **5** (freely chosen) topics, for each topic up to **3** runs may be submitted. The results of one run must be contained in one submission file (e.g. up to 3 files can be submitted per topic). A submission may contain up to 1500 retrieval results for each of the topics. Runs should be submitted until **October 18, 2004**.

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 overall submission format is defined in the following DTD:

```
<!-- XML predefined entities -->
<!ENTITY lt      "&#38;#60;">
<!ENTITY gt      "&#62;">
<!ENTITY amp     "&#38;#38;">

<!ELEMENT inex_het_track_submission (description, topic+)>
<!ATTLIST inex_het_track_submission
participant-id CDATA #REQUIRED
run-id CDATA #REQUIRED
query (automatic | manual) #REQUIRED
topic-part (T|D|K|TD|TK|DK|TDK) #IMPLIED
task CDATA #IMPLIED
>

<!ELEMENT description (#PCDATA)>

<!ELEMENT topic (subcollections, result*)>
<!ATTLIST topic
topic-id CDATA #REQUIRED
>
<!ELEMENT subcollections (subcollection+)>
<!ELEMENT result (subcollection, file, path, rank?, rsv?)>
<!ELEMENT subcollection EMPTY>
<!ATTLIST subcollection name CDATA #REQUIRED>

<!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:2004/inex04/servlet/ShowParticipants>), 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 whether the query was constructed automatically or manually from the topic. The submission may contain topic-part that was used for searching and task and task type.

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 topic the considered subcollections must be specified and a maximum of 1500 result elements may be included. A result element is described by a file name, the subcollection in that the result element has been found, an element path and it may include rank and/or retrieval status value (rsv) information.

Before detailing the various elements of the above DTD, here is a sample submission file:

```
<inex_het_track_submission participant-id="16" run-id="VSM_Aggr_06" query="manual">
  <description>Using VSM to compute RSV at leaf level combined with aggregation
  at retrieval time, assuming independence and using
  augmentationweight=0.6.</description>
  <topic topic-id="01">
    <subcollections>
      <subcollection name="Berkeley"/>
      <subcollection name="CompuScience"/>
      <subcollection name="hcibib"/>
    </subcollections>
    <result>
      <subcollection name="hcibib"/>
      <file>hcibib</file>
      <path>/article[1]/bm[1]/ack[1]</path>
      <rank>1</rank>
      <rsv>.05234</rsv>
    </result>
    <result>
      <subcollection name="CompuScience"/>
      [...]
    </result>
    <result>
      [...]
    </result>
    [...]
  </topic>
  <topic topic-id="02">
    [...]
  </topic>
</inex_het_track_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.

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. Within INEX Het Track submissions, elements are identified by means of a file name, subcollection name and an element (node) path specification, which must be given in XPath syntax. File names must be given relative to the INEX Het Track subcollection's directory (excluding the subcollection directory from the file path). The file path should use `'` for separating directories. Note that only article files (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 subcollection, file name and element path.

Examples:

```
<result>
  <subcollection name="qmuldcfdbpub">
    <file>qmuldcfdbpub</file>
    <path>/document[1]/title[1]</path>
  </result>

<result>
  <subcollection name="Berkeley">
    <file>marc_cs_data_v2_utf8_hettrack</file>
    <path>/document[1]/title[1]</path>
  </result>
```

Result Submission Procedure

To submit a run, please use the following link:

http://inex.is.informatik.uni-duisburg.de:2004/cgi-bin/inex?mode=browse_submissions_het

October 13, 2004

Zoltán Szlávik, Thomas Rölleke



Track Guidelines

Motivation

The main motivation for the track 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.

For 2004, we address the first issue: to investigate the behaviour of searchers when presented with the retrieved components of XML documents.

Data collection to be used

The INEX data collection from the ad hoc track (version 1.4) will be used in the track.

Tasks/Topics

The tasks used are derived from the CO INEX 2004 topics. In order to make the topics comprehensible by other than the topic author, it was required that the INEX 2004 topics not only detail *what* is being sought for, but also *why* this is wanted, and in what *context* the information need has arisen. The aim of this extension of the topics was for them to act as simulated work task situations 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.

Four tasks will be used, divided into two task categories. The idea is to explore the effect that task type may also have on user behaviour. The categories to be used are:

- Background category (B): Most of the INEX topics are in this category. The topics express an information need in the form of "I'd like to find out about X". Two tasks will be given in this category, based on topics 192 and 180. The tasks will correspond to part of the Narrative and will be distributed to participants.
- Comparison category (C): There are a number of topics whose subject is along the lines of: "Find differences between X and Y". Two tasks will be given in this category, based on topics 188 and 198. The tasks will correspond to part of the Narrative and will be distributed to participants.

Each searcher will choose one task from each category. This allows the topic to be more "relevant" and interesting to the searcher. A maximum time limit of 30 minutes will apply. Sessions can finish before this if searchers feel they have completed the task.

Instructions to be given to Searchers

A document with "Instructions for searchers" will be made available for download from the track pages (<http://inex.is.informatik.uni-duisburg.de:2004/tracks/int/>). This will briefly outline the purpose of the study, the nature of the corpus, and how to assess the components returned by the system.

Please note that in contrast to the assessments made for the ad hoc track, there is no requirement on the test persons to view each retrieved component as independent from other components viewed. Experiences from user studies clearly show that users learn from what they see during a search session. To impose a requirement on searchers to discard this knowledge creates an artificial situation and will restrain the test persons from interacting with the retrieved components in a natural way.

Two aspects of relevance are measured in the assessments: the extent to which the retrieved component contains information that is *useful* for solving the given task, and the extent to which the retrieved

component is focussed on the topic of the task, i.e., how *specific* the retrieved information is in relation to the task.

The *usefulness* is measured on the following scale:

Highly useful (3): The displayed component contains information that is useful for solving most or all aspects of the task.

Fairly useful (2): The displayed component contains information that is useful for solving many aspects of the task.

Marginally useful (1): The displayed component contains information that is only useful for solving a single or very few aspects of the task.

Not useful (0): The displayed component does not discuss the topic of interest at all, and is of no help in solving the task.

The *specificity* is measured on the following scale:

Highly specific (3): The topic of the task is the only theme of the displayed component.

Fairly specific (2): The topic of the task is a major theme of the displayed component.

Marginally specific (1): The topic of the task is a minor theme of the displayed component.

Not specific (0): The topic of the task is not a theme of the displayed component.

As can be noted the definitions of usefulness and specificity are close to those in the ad hoc track. The amount of information valuable for solving a task is labelled usefulness rather than exhaustiveness because we believe that usefulness is a much more understandable concept to test persons and easier to relate to the task.

The two scales are combined into the following 10 categories on the relevance assessment list:

Very useful & Very specific
Very useful & Fairly specific
Very useful & Marginally specific
Fairly useful & Very specific
Fairly useful & Fairly specific
Fairly useful & Marginally specific
Marginally useful & Very specific
Marginally useful & Fairly specific
Marginally useful & Marginally specific
Contains no relevant information

The intention with the scale is to measure the amount of relevant information contributed to the solving of the task by the displayed components, and at the same time measure how specific the information in the component is.

System

For this first year of the interactive track a system will be provided by the organisers. The system used is the HyREX XML search engine for the retrieval of document components, and a custom made interface to display results, collect relevance assessments and allow browsing. A brief guide to the system will be made available to participants along with information on how to access the system via the web, or how to install it locally. Two interfaces are available in the system: A simple baseline interface and an additional graphical interface. Each site must submit logs and questionnaires for *at least 8 searchers* with the former interface, i.e. the simple baseline interface described in Section 4 of the System guide. The latter interface is an advanced graphical interface based on Partial Treemaps (See Section 5 of the system guide). The use of the advanced interface is optional and will require additional searchers.

Experimental Design

A minimum of 8 searchers from each participating site will be used. Each searcher will search on one task from each task category. The task will be chosen by the searcher. The order in which task categories are performed by searchers will be permuted. This means that one complete round of the experiment requires only 2 searchers.

The order of an experimental session will be:

- 1 Introductory stuff: General briefing of the experiment and procedures
- 2 Before-experiment questionnaire
- 3 Hand out Instructions for Searchers
- 4 System tutorial
- 5 Category task selection from the first category
- 6 Before-task questionnaire
- 7 Search session
- 8 After-task questionnaire
- 9 Repeat 5-8 for the other task category
- 10 After-experiment questionnaire
- 11 Informal discussion/interview: any additional views on the experiment, system, etc. the searcher wishes to share.

It is important that the experiment takes place in an undisturbed environment, and that an experimenter is present in the room at all times. The role of the experimenter is to remain in the background and be ready to help out in case of problems with the system, and to oversee that everything goes according to plan. The system training and the two tasks should be performed in one, continuous session.

The minimum experimental matrix for one site will be:

Searcher	First task category	Second task category
1	Background (B)	Comparison (C)
2	(C)	(B)
3	(B)	(C)
4	(C)	(B)
5	(B)	(C)
6	(C)	(B)
7	(B)	(C)
8	(C)	(B)

This matrix can be augmented by adding blocks of 4 users (a total of 12, 16, 20, etc. users).

System Logs

The bulk of the data from this study will be collected in the system logs. It is important to ensure that each search session is saved in an appropriately named log file. The file name must consist of the id of the participating site (the one used for logging into the participants area on the INEX web site, e.g., qmul, lip6, dbdk, etc.), searcher id, and task type. For example,

qmul_searcher001_B.log
qmul_searcher001_C.log

Questionnaires

A number of questionnaires are provided by the track organisers, and each of these must be completed for every searcher/task. It is important to ensure that the searcher id, experimental condition and other ids are clearly marked on each questionnaire. To allow this, there are fields to be filled by the experimenter at the top of the questionnaires:

- Participating site: the name of the organisation (e.g. qmul, cwi, lip6, etc.)
- Searcher id: an identifier for the searcher indicated by a number (searcher id must match the id used in the respective log filename – see previous section)
- Searcher condition (BC/CB): Indicates whether the searcher executes a background task first and then a comparison one (BC) or vice-versa (CB)

- Task ID (B1 B2 C1 C2): the four ids correspond to the task ids; the appropriate id should be circled.

In addition to the questionnaires, there is a form that can be used by the experimenter present at the study for noting any problems, comments, questions or interesting observations about the searchers' behaviour. In addition, the times when searches begin and end a task should be noted, as well as any irregularities in the experiment (for instance, if the searching is interrupted for some reason). A spreadsheet will be provided by the track organisers in which to record the data from the different paper questionnaires filled in by each searcher. The spreadsheets should be saved using file names similar to those of the logs, e.g.,

qmul_searcher001.xls
qmul_searcher002.xls

Submission of Results

The 2 logs from each searcher session, and the spreadsheet with the questionnaire data from each searcher must be e-mailed back to the track organisers at tassos@dcs.qmul.ac.uk

In addition, a photocopy of the completed questionnaires should be posted to:

Tassos Tombros, Department of Computer Science, Queen Mary University of London, Mile End Rd., London E1 4NS, U.K.

After submission the track organisers will check the submissions and bundle the log and questionnaire data into a package that will be made available to all active participants in the track. The package will also be made available to the group under INEX working with metrics for the purpose of developing better ones. An initial analysis of the results will be done by the track organisers and the results will be presented at the INEX Workshop. The track participants may also analyse the data in any way they see fit, and present the results of such analyses at the INEX workshop or in other fora. It is also hoped that after the workshop in Dagstuhl interested participants will agree to work together on analysing specific aspects of the data.

Schedule

Distribution of Guidelines, Tasks, System and System Documentation: 15 September 2004

Submission of results to track organisers: 30 October 2004

INEX 2004 Workshop: 6-8 December 2004

September 16, 2004

Tassos Tombros, Birger Larsen, Saadia Malik

HyREX for INEX iTrack

Saadia Malik
University of Duisburg-Essen

Anastasios Tombros
Queen Mary University of London

Birger Larsen
Royal School of Library and Information Science

17th September 2004

1 Introduction

For iTrack, we plan to use the *Hyper-media Retrieval Engine for XML (HyREX)* information retrieval system with two different interfaces. One, simple interface, presents the results in simple ranked list and will be used as a baseline system while the other, graphical interface, presents the results in Partial Treemaps. The interfaces are optimised for and tested on Mozilla 1.6 and later versions, as well as Mozilla Firefox 0.9.3 (can be downloaded from <http://www.mozilla.org>)

The aim of this document is to describe these interfaces, as well as the logging and downloading information in detail.

2 Logging in

On the login page the experimenter must choose between the simple baseline system or the graphical interface. Either option will result in a prompt for User Name and Password. The organisers will provide user names and passwords for every test person that comply with the naming rules detailed in the Track guidelines (the use of which will ensure that the log files are named correctly).

The login of the current user is displayed in the top left corner of the search and result pages. To login as a different user all browser windows need to be closed. In addition to the logins for the interface, the user name and password for the INEX participants' area is needed in order to display the retrieved documents. This will be prompted for once on displaying the first document.

New users may be created from the login page, but are not effective until the server is restarted by the organisers.

3 Query Posting

After logging in a search form appears. The search form accepts Content-only (CO) Queries. These ignore the document structure and contain only content related conditions, e.g., only specifying what a document/component should be about. A CO query can be a set of space separated terms where a term may be a single word or a phrase

encapsulated in double quotes. No other operators are currently supported by the interface.

4 Simple Interface

After the query submission, a new browser window is opened. This contains the result list with up to 100 top-ranked search results with 10 results per page. Results from 11 to 100 can be viewed by clicking the 'next' link or a page number on the top of the result list. Each result is presented in the form of a summary consisting of rank, article title, author information, retrieval status value (rsv), and a pointer to the path of retrieved document component.

The last query issued is shown on top of the page along with a search form. A new, modified query can be entered in the search form. It is also possible to close the result list window and submit a new query in the initial search form.

A screenshot of the result list can be seen in Fig 1.

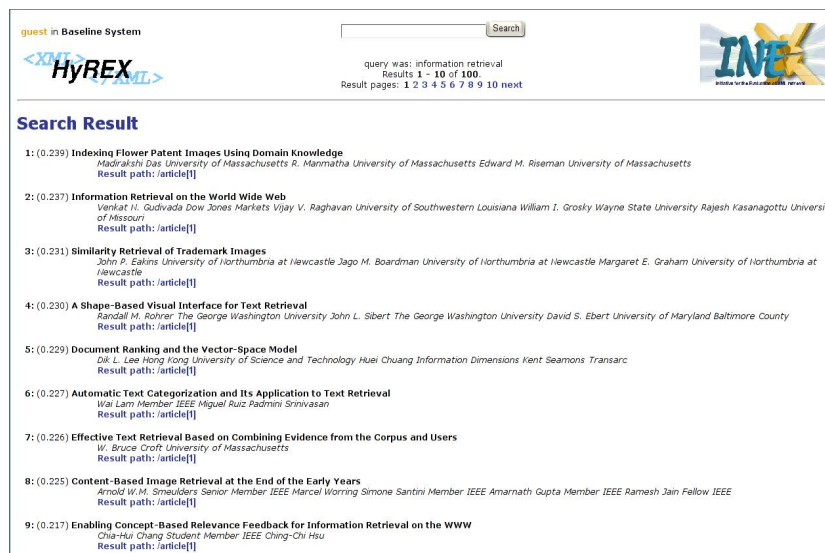


Figure 1: Ranked result list in simple interface

4.1 Result Detail

Clicking on the path of a retrieved component opens that specific document part in a new browser window together with a table of contents view. The table of contents is a tree view of the document but certain unimportant XML elements are left out to constrict the size of the tree. The retained elements are those that contribute to the overall logical structure of the document. Typically, section elements would be included but, e.g., the bold element would not be included. The document component currently selected is highlighted in yellow in the table of contents view. Sometimes this may not work so a **Refresh** button is available at the top of the table of contents view.

Next and Previous buttons (when possible) are available at the top of the document view to navigate back and forth in the document. Query terms are highlighted in red colour.

On the top and bottom of the document part view the user is asked to provide relevance assessments from a drop down box preceded by *Relevance of the displayed information*. Two aspects of relevance are to be assessed: the extent to which the retrieved component contains information that is *useful* for solving the given task, and the extent to which the retrieved component is focussed on the topic of the task, i.e., how *specific* the retrieved information is in relation to the task. See the *Instructions for searchers* for a precise definition of the two aspects.

Graded expressions of the two aspect are combined in a single 10 point scale. The possible values are:

- Very useful & Very specific
- Very useful & Fairly specific
- Very useful & Marginally specific
- Fairly useful & Very specific
- Fairly useful & Fairly specific
- Fairly useful & Marginally specific
- Marginally useful & Very specific
- Marginally useful & Fairly specific
- Marginally useful & Marginally specific
- Contains no relevant information

Once the user has finished browsing and assessing a document, the document should be closed before moving on to a new hit in the result list window. This can be done with the 'Close document' link in the top right corner, or by closing the browser window. A screenshot of the result detail window can be seen in Fig 2.

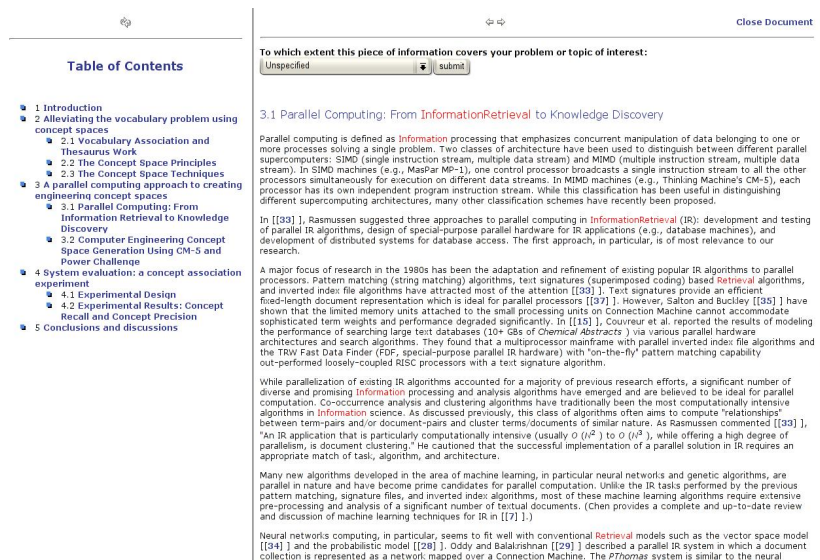


Figure 2: Result detail screenshot

5 Graphical Interface

5.1 Overview Page

At the overview page, the result set is clustered document-wise and a summary of each retrieved document is presented. The summary contains rank, title of the article, author(s), a grey shaded rectangle, and a red bar below the grey rectangle. The grey colour shades illustrate the retrieval weights and the red bar of varying length shows the number of hits. On mouseover, a Tooltip is provided on the grey rectangles and on the red bars to provide information about the retrieval weight and no of hit(s) respectively. Clicking on a title presents the result detail in Partial TreeMaps.

A screenshot can be seen in Fig 3.

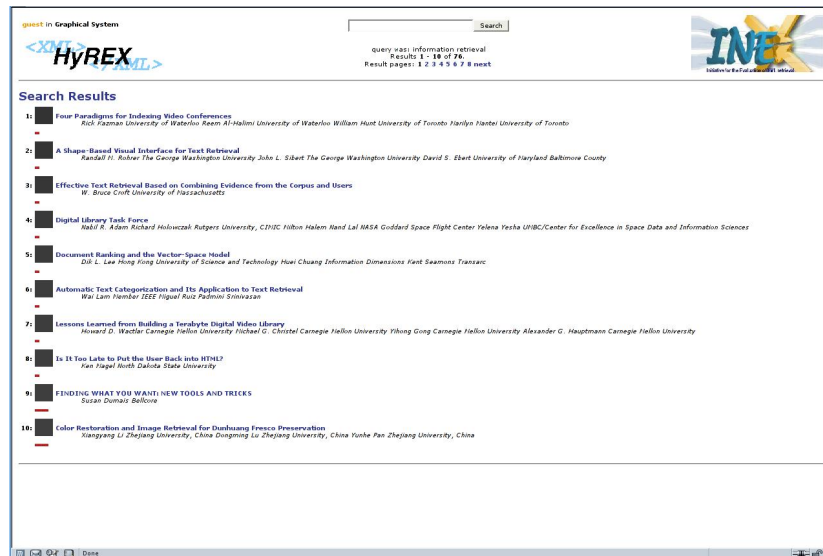


Figure 3: Screenshot depicting graphical overview clustered document wise

5.2 Partial Treemaps

This interface is developed in order to cover two aspects of XML retrieval:

1. structural or hierarchal relationship among the document elements
2. varying granularity or size of answer elements

The main idea is based on TreeMaps [Johnson and Shneiderman(1991)] that uses two dimensions in order to illustrate the structure of XML documents. A document is represented in a rectangular area and split horizontally and vertically alternatively to represent the different levels. For example, horizontal splitting for first level nodes, vertical splitting for second level nodes and horizontal splitting again for third level nodes, and so on. However, for XML documents this representation is too cluttered. Therefore, the treemap concept is augmented and the concept of *Partial Treemaps* is introduced in which non-retrieved nodes and ancestors of non-retrieved items are omitted [Kriewel(2001)].

Tooltips provide additional information about the retrieved components such as first 150 characters of contents and component name, e.g., article, section, subsection

etc. On the top of the Treemap view, all the retrieved documents are shown as small rectangles in grey shades along with the *Next* and *Previous* hyperlinks. In addition to Partial Treemaps and overview, the contents of a particular document component are also shown together with the table of contents view of the document as described above in subsection 4.1. A screenshot can be seen in Fig 4.

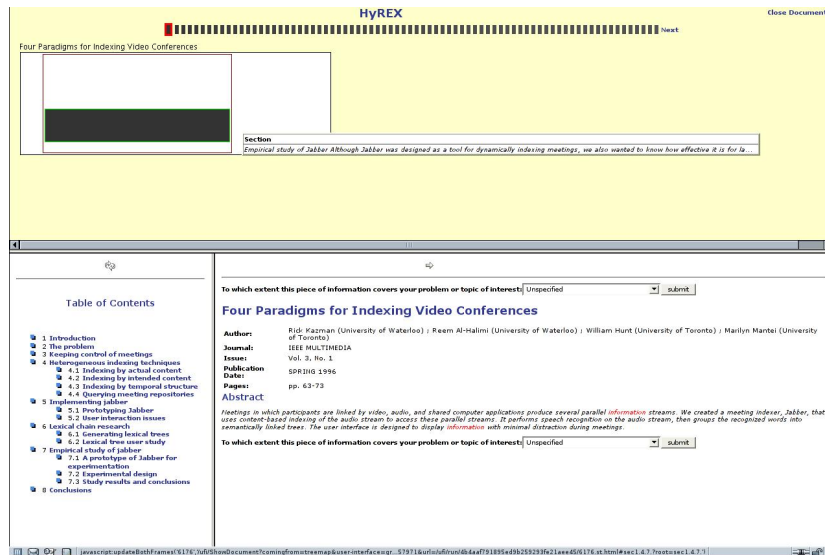


Figure 4: Screenshot depicting Partial Treemaps

6 Logging

All the events are logged in xml and plain text file format. These events include

- Feedback given by user
- document components viewed by user
- which path is followed to view this document component such as overview, structure, next previous button, treemaps.

For all these events query, docid, file, document component, rsv, and relevance of the document or document components are logged.

7 Downloading

The system is made available in two versions: One with the HyREX engine and interface mounted centrally at servers in Duisburg, and one with a local, offline installation of the interface which connects to the HyREX engine in Duisburg. The local interface comes with a detailed installation guide. The Online system can be accessed via

<http://inex1.is.informatik.uni-duisburg.de:8080/ufi/>
and <http://inex2.is.informatik.uni-duisburg.de:8080/ufi/>

The local system is made available on the following ftp server for download:

<ftp://ftp.is.informatik.uni-duisburg.de/pub/projects/hyrex/>

References

- [Johnson and Shneiderman(1991)] Brian Johnson and Ben Shneiderman. Tree-maps: A space filling approach to the visualization of hierarchical information structures. Technical Report CS-TR-2657, University of Maryland, Computer Science Department, April 1991.
- [Kriewel(2001)] Sascha Kriewel. Visualisierung für retrieval von XML-dokumenten. Master's thesis, University of Dortmund, CS Dept., December 2001.

Instructions for 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 goal of the study is to investigate the performance of a new system for information retrieval.

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 only use the terms “Information retrieval”. 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 2002.

In this experiment you will be asked to search on tasks from two task categories. For each category, you will have a choice of one from two available tasks. For each task you can take a maximum of 30 minutes. If you feel you have completed the task before the 30 minute period you can notify the experimenter and the session will be stopped.

The system may present whole articles or parts of articles to you. 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 a score from the relevance assessment list *for each viewed piece of information*. After selecting a score press ‘Submit’ to record the score in the system.

You will be asked to assess two aspects of relevance: the extent to which the retrieved component contains information that is *useful* for solving the given task, and the extent to which the retrieved component is focussed on the topic of the task, i.e., how *specific* the retrieved information is in relation to the task.

The *usefulness* is measured on the following scale:

Highly useful (3): The displayed component contains information that is useful for solving most or all aspects of the task.

Fairly useful (2): The displayed component contains information that is useful for solving many aspects of the task.

Marginally useful (1): The displayed component contains information that is only useful for solving a single or very few aspects of the task.

Not useful (0): The displayed component does not discuss the topic of interest at all, and is of no help in solving the task.

The *specificity* is measured on the following scale:

Highly specific (3): The topic of the task is the only theme of the displayed component.

Fairly specific (2): The topic of the task is a major theme of the displayed component.

Marginally specific (1): The topic of the task is a minor theme of the displayed component.

Not specific (0): The topic of the task is not a theme of the displayed component.

The two scales are combined into the following 10 categories on the relevance assessment list:

Very useful & Very specific
Very useful & Fairly specific
Very useful & Marginally specific
Fairly useful & Very specific
Fairly useful & Fairly specific
Fairly useful & Marginally specific
Marginally useful & Very specific
Marginally useful & Fairly specific
Marginally useful & Marginally specific
Contains no relevant information

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)

Thank you for your help!