

INEX 2006 Workshop Pre-Proceedings

December 18-20, 2006
Schloss Dagstuhl
International Conference and Research
Center for Computer Science

<http://inex.is.informatik.uni-duisburg.de/2006/>



**Edited by
Norbert Fuhr
Mounia Lalmas
Andrew Trotman**



TABLE OF CONTENTS

Organizers	vii
Preface	ix
Acknowledgements	x
Schloss Dagstuhl	xi

METHODOLOGY

Choice of Parameter Values for the INEX Evaluation Metrics: Sensitivity Analysis	1
G. Kazai	
XML Retrieval Evaluation Revisited: A Comparison of Metrics	3
J. Pehcevski, J.A. Thom and A.-M. Vercoustre	

AD HOC TRACK

Efficient, Effective and Flexible XML Retrieval Using Summaries	6
M.S. Ali, M. Consens, X. Gu, Y. Kanza, F. Rizzolo and R. Stasiu	
Using Topic-shifts in XML Retrieval at INEX 2006	21
E. Ashoori and M. Lalmas	
Structured Content-Only Information Retrieval Using Term Proximity and Propagation of Title Terms	29
M. Beigbeder	
Influence Diagrams and Structured Retrieval: Garnata implementing the SID and CID models at INEX'06	37
L.M. de Campos, J.M. Fernandez-Luna, J.F. Huete and A.E. Romero	
Information theoretic retrieval with structured queries and documents	49
C. Carpineto, G. Romano and C. Caracciolo	
Dynamic Element Retrieval in the Wikipedia Collection	53
C.J. Crouch, D.B. Crouch, M. Ganapathibhotla and V. Bakshi	
The University of Kaiserslautern at INEX 2006	55
P. Dopichaj	
Indexing "Reading Paths" for a Structured Information Retrieval at INEX 2006	62
M. Gery	
GPX at INEX 2006	67
S. Geva	
Robert Gordon University at INEX 2006: Adhoc Track	70
F. Huang, S. Watt, D. Harper and M. Clark	
Tuning and evolving retrieval engine by training on previous INEX testbeds: preliminary work	79
G. Hubert	

The University of Amsterdam at INEX 2006 J. Kamps, M. Koolen and B. Sigurbjornsson	88
Using Language Models and the HITS Algorithm for XML Retrieval B. Kimelfeld, E. Kovacs, Y. Sagiv and D. Yahav	100
CSIRO's participation in INEX 2006 A. Krumpholz and D. Hawking	101
EXTIRP: baseline retrieval from Wikipedia M. Lehtonen and A. Doucet	102
CISR at INEX 2006 W. Lu, S. Robertson and A. Macfarlane	104
A scalable XML component ranking algorithm Y. Mass	111
Indian Statistical Institute at INEX 2006 Adhoc track: A Preliminary VSM Approach S. Pal, M. Mitra and P. Majumder	118
SIRIUS XML IR System at INEX 2006: Approximate Matching of Structure and Textual Content E. Popovici, G. M�nier and P.-F. Marteau	121
A Method of Preferential Unification of Plural Retrieved Elements for XML Retrieval Task H. Tanioka	134
TopX – AdHoc and Feedback Tasks M. Theobald, A. Broschart, R. Schenkel, S. Solomon and G. Weikum	140
Supervised and Semi-supervised Machine Learning Ranking J.-N. Vittaut and P. Gallinari	150
PF/Tijah at INEX 2006 T. Westerveld, H. Rode, R. van Os, D. Hiemstra, G. Ramirez, V. Mihajlovic and A.P. de Vries	159
XSee: Structure Xposed R. van Zwol and W. Weerkamp	160
 NATURAL LANGUAGE TRACK <hr/>	
Using Rich Document Representation in XML Information Retrieval F. Oroumchian, F. Raja, M. Keikha and M. Rahgozar	170
NLPX at INEX 2006 A. Woodley and S. Geva	178
Shallow parsing of INEX Queries H. Zargayouna, V. Rosas and S. Salotti	179

HETEROGENEOUS TRACK

- The Heterogeneous Collection Track at INEX 2006** 189
I. Frommholz and R. Larson
- Probabilistic Retrieval approaches for Thorough and Heterogeneous XML Retrieval** 196
R. Larson

MULTIMEDIA TRACK

- Social Media Retrieval using Image Features and Structured Text** 208
D.N.F. Awang Iskandar, J. Pehcevski, J.A. Thom and S.M.M. Tahaghoghi
- XFIRM at INEX 2006 - Preliminary work. Ad-hoc, Relevance Feedback and MultiMedia tracks** 221
L. Hlaoua, M. Torjmen, K. Pinel-Sauvagnat and M. Boughanem
- Information Fusion in XML Document Searches by Combining Text and Image Retrieval Techniques** 231
D. Tjondronegoro, C. Lau, J. Zhang and S. Geva
- Benchmarking Multimedia Search in Structured Collections** 236
T. Westerveld and R. van Zwol

INTERACTIVE TRACK

- Kyungpook National University at INEX 2006: Interactive Track** 249
Y. Jung and H. Kim
- The Interactive Track at INEX 2006** 250
B. Larsen, S. Malik and A. Tombros
- Context revisited - element retrieval behaviour and genre dependency** 262
R. Nordlie and N. Pharo
- Evaluating Tasks by Type and Form** 264
E. Toms, L. Fruend, C. Jordan, T. Mackenzie, S. Toze and H. O'Brien

USE CASE TRACK

- A Taxonomy for XML Retrieval Use Cases** 267
M. Lehtonen, N. Pharo and A. Trotman
- Inter-assessor agreement at INEX 06** 273
N. Pharo, A. Trotman, S. Geva and B. Piwowski
- XML-IR Users and Use Cases** 274
A. Trotman, N. Pharo and M. Lehtonen
- What XML-IR Users Want** 287
A. Woodley, S. Geva and S.L. Edwards

DOCUMENT MINING TRACK

On the unsupervised classification of text-centric XML document collections	288
A. Doucet and M. Lehtonen	
XML Document Mining using Contextual Self-Organizing Maps for Structures	292
M. Kc, M. Hagenbuchner, A.C. Tsoi, F. Scarselli, M. Gori and A. Sperduti	
FAT-CAT: Frequent Attributes Tree based Classification	307
J. De Knijf	
XML Structure Mapping Application to the PASCAL/INEX 2006 XML Document Mining Track	319
F. Maes, L. Denoyer and P. Gallinari	
Clustering XML Documents by Structural Similarity with PCXSS	331
T. Tran, R. Nayak and K. Raymond	
Classifying XML Documents Based on Structure/Content Similarity	342
G. Xing and Z. Xia	
XML Document Mining using Graph Neural Network	354
S.L. Yong, M. Hagenbuchner, A.C. Tsoi, F. Scarselli and M. Gori	

APPENDIX

ADHOC TRACK

The Wikipedia XML Corpus	367
L. Denoyer and P. Gallinari	
INEX 2006 Guidelines for Topic Development	373
B. Larsen, A. Trotman, <i>et al.</i>	
INEX 2006 Retrieval Task and Result Submission Specification	381
C. Clarke, J. Kamps and M. Lalmas	
INEX 2006 Relevance Assessment Guide	389
M. Lalmas and B. Piwowarski	

HETEROGENEOUS TRACK

INEX'06 Het Track Retrieval Task and Result Submission Specification	396
I. Frommholz and R. Larson	

MULTIMEDIA TRACK

INEX 2006 Multimedia Track Guidelines	399
T. Westerveld, R. van Zwol, <i>et al.</i>	

XML ENTITY RANKING

Entity Ranking – Guidelines DRAFT v1
A.P. de Vries and N. Craswell

413

ORGANIZERS

PROJECT LEADERS

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

CONTACT PEOPLE

Saadia Malik (University of Duisburg-Essen)
Zoltán Szlávik (Queen Mary University of London)

WIKIPEDIA DOCUMENT COLLECTION AND EXPLORATION

Ludovic Denoyer (Université Paris 6)
Martin Theobald (Max-Planck-Institute for Informatics)

USE CASE STUDIES

Andrew Trotman (University of Otago)
Nils Pharo (Oslo University College)

TOPIC FORMAT SPECIFICATION

Andrew Trotman (University of Otago)
Birger Larsen (Royal School of Library and Information Science)

TASK DESCRIPTION

Jaap Kamps (University of Amsterdam)
Charlie Clarkes (University of Waterloo)

ONLINE RELEVANCE ASSESSMENT TOOL

Benjamin Piwowarski (Yahoo! Research Latin America)

METRICS

Gabriella Kazai (Microsoft Research Cambridge)
Stephen Robertson (Microsoft Research Cambridge)
Paul Ogilvie (Carnegie Mellon University)

RELEVANCE FEEDBACK TASK

Yosi Mass (IBM Research Lab)
Ralf Schenkel (Max-Planck-Institute for Informatics)

NATURAL QUERY LANGUAGE TASK

Shlomo Geva (Queensland University of Technology)
Xavier Tannier (Xerox)

HETEROGENEOUS COLLECTION TRACK

Ingo Frommholz (University of Duisburg-Essen)
Ray Larson (University of California, Berkeley)

INTERACTIVE TRACK

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

DOCUMENT MINING TRACK

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

XML MULTIMEDIA TRACK

Roelof van Zwol (Yahoo! Research)
Thijs Westerveld (CWI)

XML ENTITY SEARCH TRACK

Arjen de Vries (CWI)
Nick Craswell (Microsoft Research Cambridge)

PREFACE

Welcome to the 5th workshop of the Initiative for the Evaluation of XML Retrieval (INEX)!

Now, in its fifth year, INEX is an established evaluation forum for XML information retrieval (IR), with over 80 participating organizations worldwide. Its aim is to provide an infrastructure, in the form of a large XML test collection and appropriate scoring methods, for the evaluation of XML IR systems.

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

2006 was an exciting year for INEX, and brought with it a lot of changes and new aspects to the evaluation. In total nine research tracks were included in INEX 2006, which studied different aspects of XML information access: Ad-hoc, Interactive, Use Case, Multimedia, Relevance Feedback, Heterogeneous, Document Mining, Natural Language (NLP), and Entity Ranking. The Use Case and Entity Ranking tracks were new for the 2006 campaign. The consolidation of the existing tracks, and the expansion to new areas offered by the two new tracks, allows INEX to grow in reach.

The aim of the INEX 2006 workshop is to bring together researchers in the field of XML IR who participated in the INEX 2006 campaign. During the past year participating organizations 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, summarizes and addresses encountered issues and devises a work plan for the future evaluation of XML retrieval systems.

ACKNOWLEDGEMENTS

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

We gratefully thank organizers of the various tracks for their great work in setting up the new tracks, and carrying on and refining the existing tracks. Thanks also to those involved in running and coordinating the *ad hoc* track which each year involves a major effort.

As always, special thanks go to the participating organizations and people for their contributions and hard work throughout the year! The first point of contact of many of us is either Saadia Malik or Zoltán Szlávik for whom we, and we are sure every participant gives thanks.

We hope you have enjoyed the INEX 2006 campaign and have fruitful and stimulating discussions at the workshop.

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

December 2006

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/>

Choice of Parameter Values for the INEX Evaluation Metrics: Sensitivity Analysis

Gabriella Kazai

Microsoft Research,
Cambridge, UK
gabkaz@microsoft.com

Abstract. This paper investigates the measures of retrieval effectiveness employed in the ad-hoc track of INEX 2006. In particular, it looks at how sensitive the different metrics are to various parameters and how the choice of certain parameter values may influence the conclusions of the evaluation. One such parameter - which is the main focus of the investigation - is the methodology employed in generating an ideal recall-base for the evaluation of the Focused task. The paper also examines the correlation between the different INEX measures and traditional IR metrics.

1 Introduction

INEX 2006 defined four retrieval tasks within the ad-hoc track: Thorough, Focused, Relevant in Context, and Best in Context retrieval. The evaluation of all these tasks relies on the same set of relevance assessments collected from human judges. The assessments are collected in the form of highlighted text passages which are then converted into assessments on XML elements. The conversion involves the calculation of a specificity score for each XML element that contains highlighted text fragments. The specificity score is given as the ratio of the number of highlighted characters contained within the XML element to the length of the element. The resulting recall-base consists of overlapping XML elements with varying specificity scores. For example, a partially highlighted paragraph text fragment will add, e.g., a paragraph, a section and an article XML element to the recall-base. This recall-base can then be used directly to evaluate the Thorough task, where systems are required to rank all elements of the collection by their estimated relevance. However, due to the introduced overlap of XML elements, this recall-base is not directly suitable for the evaluation of the Focused task, where systems are required to return only the “most focused” relevant XML elements. This necessitates the identification of these most focused XML elements from the collected assessments. In [2, 1] a procedure was proposed to filter the above-derived recall-base and generate a so-called ideal recall-base, where the overlap is removed. The methodology has since been questioned and alternative methods have also been proposed, e.g. in [3]. The question of which method is correct, however, remains open. In this paper, a number of different

methods for the construction of the ideal recall-base are proposed and compared. More importantly, however, the paper investigates how the evaluation results are actually affected by the choice of methodology.

In addition, a comparison is made between the measures used at INEX and those used traditionally in IR, e.g. precision and recall.

References

1. G. Kazai and M. Lalmas. eXtended Cumulated Gain Measures for the Evaluation of Content-oriented XML Retrieval. *ACM Transactions on Information Systems (ACM TOIS)*, To appear.
2. G. Kazai, M. Lalmas, and A. de Vries. Reliability tests for the xcg and inex-2002 metrics. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlavik, editors, *Advances in XML Information Retrieval. Third Workshop of the INitiative for the Evaluation of XML Retrieval INEX 2004, Schloss Dagstuhl, 6-8 December 2004*, volume 3493 of *Lecture Notes in Computer Science*, pages 60–72. Springer-Verlag, 2005.
3. B. Piwowarski, P. Gallinari, and G. Dupret. An extension of precision-recall with user modelling (PRUM): Application to XML retrieval. *Transactions on Information Systems (To appear)*, 2006.

XML Retrieval Evaluation Revisited: A Comparison of Metrics

Jovan Pehcevski^{1,2}, James A. Thom¹, and Anne-Marie Vercoustre²

¹ School of Computer Science and Information Technology, RMIT University,
Melbourne, Australia

{jovanp, jat}@cs.rmit.edu.au

² INRIA, Rocquencourt, France

Anne-Marie.Vercoustre@inria.fr

Extended Abstract

Finding the appropriate approach to evaluate XML retrieval effectiveness is the subject of ongoing debate within the XML information retrieval (IR) research community. Indeed, there is an abundance of metrics (and measures) that can be used to evaluate the effectiveness of XML retrieval systems [1–3, 5–7]. However, these metrics are based on different relevance assumptions, incorporate different hypotheses of the expected user behaviour, and implement custom evaluation methodologies to handle the overlap problem. This results in different XML retrieval behaviours being measured by different metrics [9].

Over the past four years, INEX has been used as an arena to investigate the behaviour of these different XML retrieval evaluation metrics. From 2005, the eXtended Cumulated Gain (XCG) family of metrics were adopted as official INEX metrics [5]. The XCG metrics are extensions of the cumulated gain metrics initially used in traditional document retrieval [4]. For an INEX topic, an *ideal* cumulated gain vector is first constructed by sorting the document components in the recall-base in decreasing order of their relevance scores. The *actual* cumulated gain vector is then compared to the ideal vector by plotting the corresponding relevance scores against each rank position. Two monotonically increasing curves are observed as a result, which should level out after no more relevant documents are found. To measure the retrieval performance of an XML IR system, the cumulated gains of the actual vector, obtained for each rank, are divided by those of the ideal vector. The ideal retrieval performance at a rank cutoff is therefore achieved when the obtained normalised value is 1, whereas the area between the normalised actual and ideal curves shows the quality of the retrieval approach (the less wide the area is, the better the retrieval performance).

We contend that the purpose of an XML retrieval system is to identify and retrieve elements that contain *as much relevant information as possible*, while *minimising the amount of non-relevant information retrieved*. To measure the extent to which an XML retrieval system returns relevant information, we use the HiXEval metric [6] that extends the traditional definitions of precision and recall and considers only the *Specificity* value of a retrieved element (the amount of highlighted relevant text in the element). This is supported by the fact that,

from 2006, the INEX relevance definition only uses *Specificity* as a relevance dimension.³

In this paper we revisit the XML retrieval evaluation by comparing HiXEval with the two official XCG metrics (nxCG and ep/gr) on the *Thorough* and *Focussed* tasks of the INEX 2006 Ad-hoc track. We do this in two ways.

First, we make direct use of the INEX evaluation methodology — its aim to order XML retrieval runs to understand which retrieval techniques work well and which do not — to find how the run orderings obtained by the HiXEval measures correlate to the run orderings obtained when using measures from the two XCG metrics.

Second, we test the reliability of the three metrics by investigating the extent to which they are capable at distinguishing between different XML retrieval approaches. To test metric reliability, we pursue a simplification of the methodology introduced by Sanderson and Zobel [8] that enables us to identify *significance* and *error* rates for measures in both HiXEval and the two XCG metrics. The methodology is as follows. We first divide the topics that belong to the official INEX 2006 Ad-hoc topic set into four random subsets. Under each of the two INEX 2005 Ad-hoc tasks (*Thorough* and *Focussed*), we then use these topic subsets to pairwise compare the runs submitted by the INEX 2006 participants. For a pair of runs on the first topic subset, a *t-test* is used to decide whether the observed performance difference between the pair is *significant* at the 0.05 confidence level. If it is, and if the same two runs have the opposite numeric ordering on any of the other three topic subsets, an *error* is recorded (the total pairwise comparisons made for a task may vary and will depend on the number of submitted runs). A metric is considered more reliable than another metric if its measures identify more significant differences than do measures in the other metric, and at the same time the obtained error rates are no worse.

References

1. A. de Vries, G. Kazai, and M. Lalmas. Tolerance to irrelevance: A user-effort evaluation of retrieval systems without predefined retrieval unit. In *Proceedings of RIAO 2004*, pages 463–473, Avignon, France, 2004.
2. N. Gövert, N. Fuhr, M. Lalmas, and G. Kazai. Evaluating the effectiveness of content-oriented XML retrieval methods. *Information Retrieval*, 9(6):699–722, 2006.
3. D. Hiemstra and V. Mihajlovic. The simplest evaluation measures for XML information retrieval that could possibly work. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 6–13, Glasgow, UK, 2005.
4. K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.
5. G. Kazai and M. Lalmas. INEX 2005 evaluation measures. In *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November*

³ M. Lalmas and B. Piwowarski, “INEX 2006 relevance assessment guide”. Available at: <http://inex.is.informatik.uni-duisburg.de/2006/adhoc-protected/assessments.html> (INEX authentication required).

- 28-30, 2005, *Revised Selected Papers*, volume 3977 of *Lecture Notes in Computer Science*, pages 16–29, 2006.
6. J. Pehcevski and J. A. Thom. HiXEval: Highlighting XML retrieval evaluation. In *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005, Revised Selected Papers*, volume 3977 of *Lecture Notes in Computer Science*, pages 43–57, 2006.
 7. B. Piwowarski and G. Dupret. Evaluation in (XML) information retrieval: Expected precision-recall with user modelling (EPRUM). In *Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 260–267, Seattle, USA, 2006.
 8. M. Sanderson and J. Zobel. Information retrieval system evaluation: effort, sensitivity, and reliability. In *Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 162–169, Salvador, Brazil, 2005.
 9. A. Trotman. Wanted: Element retrieval users. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 63–69, Glasgow, UK, 2005.

Efficient, Effective and Flexible XML Retrieval Using Summaries

M. S. Ali, Mariano Consens, Xin Gu, Yaron Kanza, Flavio Rizzolo, and Raquel Stasiu

University of Toronto

{sali, consens, xgu, yaron, flavio, raquel}@cs.toronto.edu

Abstract. Retrieval queries that combine structural constraints with keyword search are placing new challenges on retrieval systems. This paper presents *TReX*—a new retrieval system for XML. *TReX* uses *structural summaries* to efficiently retrieve elements given structural constraints. *TReX* can efficiently return either all the answers to a given query or only the top- k answers. In this paper, we discuss our participation in the annual Initiative for the Evaluation of XML Retrieval (INEX) workshop in the ad-hoc track. Specifically, we investigate the use of summaries and the flexibility they provide when dealing with structural constraints. We present an algorithm for retrieval using summaries. Finally, experimental results are presented showing that *TReX* answers queries efficiently and effectively.

1 Introduction

Recent research efforts have combined the structured data management capabilities of databases with the powerful keyword search capabilities of information retrieval (IR) systems. One of the best known of these research efforts is the INEX [1] initiative. INEX is a forum dedicated to research in information retrieval from collections of XML documents. In XML retrieval, queries are combinations of keywords (content queries), structural hints (vague queries) and structural constraints (strict queries). Query responses are composed of XML document fragments (*i.e.*, specific elements) that satisfy the structural conditions and are returned ranked according to relevance criteria based on the content and structural components of the query.

To assess the *effectiveness* of the ranked answers returned by XML retrieval systems, human judgments are collected for the answers to standard queries, which are called topics, on XML collections. The collections are shared among all of the INEX participants. Based on the collections, INEX participants propose and agree on the topics for the human judges. System implementors develop their ranking criteria and assess the quality of the answers from their systems against the human judgments. Participants' ranking criteria generally use well-established IR techniques for content scoring that have been extended to incorporate the structural conditions specified in the topic. We refer to this extension as *structural scoring*. The XML retrieval community is just starting to develop an

understanding of structural scoring. We expect that in the coming years a wide range of different techniques will be proposed and assessed. To this effect, our efforts have concentrated on developing an XML retrieval system that supports *flexible* structural scoring. We believe that this will foster more experimentation and will help move forward the state-of-the-art over the long term as we begin to understand the different ways that structure is used in XML retrieval. Our contention is that XML retrieval systems must be capable of *efficiently* combining IR evaluation techniques with new structural ranking capabilities. There are still a wide spectrum of challenges to overcome. As an example, this is illustrated in the strict interpretation of structural constraints because these constraints have the same efficiency demands on the system as those placed on a structured XML query engine (*i.e.*, those posed on an XPath or XQuery capable processor). *TReX* is a step toward overcoming these challenges.

In this paper we describe the techniques used by the *TReX* system to support efficient, effective and flexible XML retrieval. *TReX* retrieves relevant XML fragments by simultaneously using indexes on paths in the XML (*summaries*) and indexes on keywords (*inverted lists*). Previous work has established the advantages of using summaries for structured XML queries [6]. This paper applies summaries to content and vague structure retrieval queries. Two methods for computing queries are considered. In the *exhaustive* method, queries are computed directly from the indexes. Our second method is meant for quickly computing the top-*k* answers to a query. It relies on the exhaustive method to first pre-compute and store lists of ranked elements for each query keyword and path expression. Then, the system employs the *threshold algorithm* (TA) for efficiently combining the ranks according to the keywords in the query. We provide experimental results showing the efficiency and the effectiveness of *TReX*'s use of summaries in support of flexible structural scoring in XML retrieval.

Several proposals in the literature extend the traditional keyword-style retrieval to the XML model [8, 12, 13]. Vague structural conditions were introduced in [22] and complemented with full-text conditions in [3, 4]. A query algebra for IR style processing of XML data was introduced in [5]. Although only for keyword queries, XRANK [12] is the only system that provides efficient support for finding the top-k results. Other recent proposals for XML ranked retrieval include [15] and [17]. The former uses dataguides and TA-style top-k algorithms [10], but differs from our work in that their experiments are limited to DB-like queries rather than XML retrieval queries. In contrast, [17] focuses on efficient evaluation of approximate structural matches without considering keyword search. The closest work to ours is TopX [23]. We follow the baseline top-k algorithm described in that work, but we do not use their probability predictor function nor invoke costly random access to resolve structural constraints. Our scoring model is similar to the model in [23]. The main difference is that tags (element names) in TopX are the only structural constraints influencing the score whereas, in *TReX*, the scoring function uses more flexible summary-based constraints.

The structure of the paper is as follows. Section 2 introduces the retrieval queries supported by the *TReX* system. Section 3 introduces summaries. Section

4 describes the evaluation mechanisms used by *TReX*. Finally, Section 5 presents experimental evidence of the effectiveness and efficiency of *TReX*.

2 Retrieval Queries

TReX is designed for evaluating *NEXI* queries [24] over a given XML corpus. *NEXI* (*Narrowed Extended XPath I*) is a query language for specifying retrieval queries. It was devised and has been used in the context of the *Initiative for the Evaluation of XML Retrieval* (INEX)[21]. *NEXI* is built upon XPath [7]. On the one hand, it narrows XPath by excluding function symbols and some axes. On the other hand, it extends XPath with the function `about()`, which denotes a vague interpretation of its input. A *NEXI* query is composed of two types of constraints, structural and textual. The `about()` function can be applied to both. The structural constraints are expressed in XPath-like syntax and the textual constraints are keywords.

Example 1. Consider the following *NEXI* query
`//article[about(., XML retrieval)]//sec[about(., inverted list)].`
 This query specifies a search for sections that are relevant to the keywords “inverted list” that appear in articles that are relevant to “XML retrieval”.

The answer to a query consists of elements that satisfy the structural and textual constraints. The elements, in an answer, are ranked according to their relevance to the search. In general, elements that contain the specified search terms should be ranked higher than elements that do not. For instance, the answer to the query in Example 1 are `sec` elements that are descendants of `article` elements, *i.e.*, elements that are in the answer to the XPath expression `//article//sec`. All `sec` elements in the answer should be ranked according to their relevance to the keywords “inverted” and “list”, and the relevance of their ancestor `article` elements to the keywords “XML” and “retrieval”.

The scoring function *TReX* uses is a version of the Okapi BM25 formula [20] modified for XML. The *TReX* function is a generalization of the scoring function employed in the TopX query engine [23]. Its novelty is that the score of an element is given w.r.t. a set S of elements specified by the structural constraints of the query. Before presenting the formula, we provide some necessary notation. We denote by $tf(t, e)$ the *term frequency* of the term t in the element e . This function returns the number of occurrences of t in the textual content of e , where the textual content is considered a bag of terms. We denote by $ef_S(t)$ the element frequency of a search term t , w.r.t. a set S of elements. This function returns the number of elements that contain t , among the elements in S . The *length* of an element e , denoted $length(e)$, is the number of words in the textual content of e . That is, $length(e) = \sum_{\{t|t \text{ is a term in } e\}} tf(t, e)$. Finally, we denote the size of a set S by $|S|$.

Given a list t_1, \dots, t_m of terms, an element set S and an element e in S , the BM25 score of e is given by the following formula.

$$score_s(e | t_1, \dots, t_m) = \sum_{i=1}^m \frac{(k_1 + 1) \cdot tf(t_i, e)}{K + tf(t_i, e)} \cdot \log \left(\frac{|S| - ef_S(t_i) + 0.5}{ef_S(t_i) + 0.5} \right)$$

where

$$K = k_1 \left((1 - b) + b \cdot \frac{length(e)}{\text{avg}\{length(e') \mid e' \in S\}} \right)$$

Okapi BM25 was originally developed using statistics of all documents in the corpus. In the context of XML, BM25 has been modified to use statistics at the granularity of elements. In comparison, *TReX* uses statistics within groups of elements defined by structural constraints. More formally, our BM25 formula uses frequency statistics w.r.t. an element set S rather than using only statistics w.r.t. entire documents or individual elements. Usually, S is taken to be the set of all elements satisfying the structural constraints of the query. For instance, in the query from Example 1, S contains all the elements in the answer to the XPath expression `//article//sec`.

As tuning parameters we use the same values used in TopX. Thus, we set k_1 to 10.5 and b to 0.75. Note that k_1 controls the non-linear term-frequency effects, and b controls the element-length normalization [20]. In order to answer retrieval queries efficiently, *TReX* uses inverted lists for finding elements that contain the keywords, and summaries for finding elements that comply with the structural constraints. Summaries are discussed in the next section.

3 Structural Summaries

Structural summaries are data structures used for locating specific fragments of the data, such as nodes and subtrees. They group together elements that are indistinguishable w.r.t. a query or a class of queries in some XML query language. By accessing relevant data directly, summaries help to avoid sequential scans of entire documents during query evaluation. In addition, they can be used to *describe* the instance by keeping record of its structural properties, such as hierarchical relationships, degree of nesting, and label paths. A typical summarization of the XML tree structure is a *labeled tree* that describes its labels and edges in a concise way. In addition, XML tree nodes are partitioned into equivalence classes according to their labels or the label paths they belong to. Each node in the summary tree has one such equivalence class (usually called its *extent* in the literature) associated to it.

The partition can be induced by different criteria. For instance, the *tag summary* clusters together nodes with the same tag. The tag summary has as many extents (equivalence classes) as different tags are in the XML tree. The *incoming summary*, in contrast, partitions nodes based on the label paths from the root to the nodes, *i.e.*, the incoming label paths. Thus, nodes with the same incoming label path will belong to the same extent. It is easy to see that the extents of the incoming summary are in fact a refinement of the tag summary extents, because

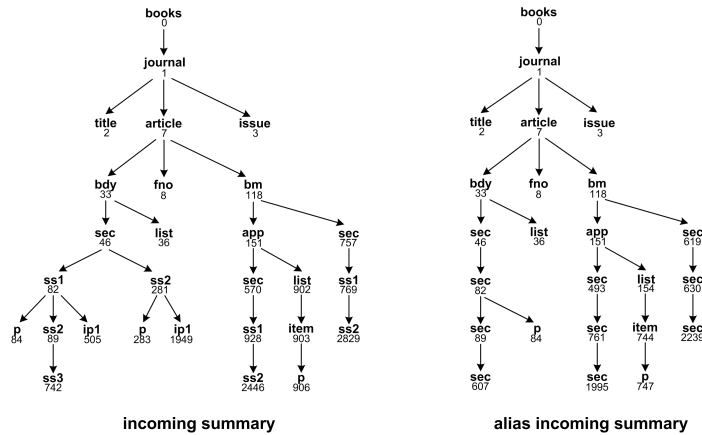


Fig. 1. Fragment of the incoming and alias incoming summary trees for the INEX IEEE collection

in order for two nodes to have the same incoming label path they also need to have the same label. The left-hand side of Figure 1 shows a fragment of the incoming summary tree for the INEX IEEE collection. (The complete incoming summary with no aliases has 11563 nodes. For the tag summary, the number of nodes is 185. The total size of the alias incoming summary is 7860. The alias tag summary has 145 nodes.) In Figure 1, the numbers below the nodes are the *summary node identifiers*, or sid's for short. For instance, all XML nodes that end with the path `books/journal/article` belong to the same incoming summary extent and, according to our summary in Figure 1, have sid 7. A sid not only identifies a summary node but also includes all XML nodes that belong to the summary node's extent. Note that if two XML nodes have the same sid, then by definition one node cannot encapsulate the other.

In an XML retrieval environment, oftentimes different elements with different tags represent the same type of information. For instance, article sections in the IEEE collection are in some places referred to as `sec` and in other places as `ss1` or `ss2`. Since `sec`, `ss1` and `ss2` are semantically the same. For a summary to reflect that fact, we make use of the *alias mapping* provided by INEX to replace all synonyms by their alias (`sec` in our example). The right-hand side of Figure 1 shows a fragment of the *alias incoming summary* tree for the INEX IEEE collection.

An alias mapping collapses different summary nodes in the non-aliased summary into a single summary node in the aliased summary. This collapse can happen for two different reasons. The first one is that nodes are combined into one because their tags are aliases of the same tag. For instance, nodes with sid's 82 and 281 in the incoming summary of Figure 1 are combined into summary node sid 82 in the alias incoming because tags `ss1` and `ss2` are mapped to (aliased with) `sec`. This type of collapse can happen in both tag and incoming

summaries. The second type of collapse is only possible in the incoming summary: two nodes collapse because their ancestors collapse. This is the case of nodes with sid’s 84 and 283 on the left-hand side of the figure. When nodes with sid’s 82 and 281 were combined into one, the incoming label path to nodes with sid’s 84 and 283 became the same and thus the two nodes were also combined into one.

Our system generates an XPath expression for each sid, which computes precisely the set of document nodes in its extent. Attaching an arbitrary XPath expression to each sid gives us the ability to precompute arbitrary path conditions in our summaries. In addition, the use of XPath provides us with a uniform mechanism for creating and manipulating *TReX* summaries.

Since our system uses sid’s internally, changing the summary only impacts the sid’s used during query evaluation. This provides the flexibility to use different summaries transparently in *TReX*. Any summary proposal in the literature can in fact be used in *TReX*. Examples of such proposals are region inclusion graphs (RIGs) [9], dataguides [11], the T-index family [18], ToXin [19], A(k)-index [16], F&B-Index and F+B-Index [14]. RIGs are examples of tag summaries whereas dataguides, 1-index, ToXin, and A(k)-index are incoming summaries. All these proposals can be expressed in our system using XPath expressions, which gives us the ability mix and match them in our summaries.

3.1 NEXI Evaluation Using Summaries

We now explain how to use structural summaries for evaluating retrieval queries. The evaluation of a *NEXI* retrieval query in *TReX* is done in two phases: translation and retrieval.

In the translation phase, each path p in the query from the root to an `about()` function is translated to a set of sid’s and a set of terms. Let E_p be the set of elements in the result of evaluating p on all the documents in the corpus. The set of sid’s consists of all the summary nodes whose extent has a non-empty intersection with E_p . The set of terms consists of all the terms that appear in the `about()` function at the end of each path p . For example, consider the query in Example 1 over the INEX IEEE collection, and the incoming summary with aliases shown on the right-hand side of Figure 1. Then, the set of sid’s for the path `//article//sec` is $\{46, 82, 89, 493, 607, 619, 630, 761, 1995, 2239\}$. The set of terms is $\{\text{inverted, list}\}$. For the path `//article` that also leads to an `about()` function, the set of sid’s is $\{7\}$ and the set of terms is $\{\text{XML, retrieval}\}$.

In the retrieval phase, elements are retrieved according to the sets of sid’s and terms generated in the translation phase. For a set of sid’s $[sid_1, \dots, sid_m]$ and a set of terms $[t_1, \dots, t_n]$, the system retrieves the elements that (1) are in the extent of a node with sid in sid_1, \dots, sid_m , and (2) contain at least one of the terms t_1, \dots, t_n . For each such element e , term and element frequencies are computed and a BM25 score $score_s(e \mid t_1, \dots, t_n)$ is calculated, where S is the extent in which e is a member. The following section discusses how the algorithms of the retrieval phase were implemented in *TReX*.

Elements(<u>SID</u> , <u>docID</u> , <u>endPos</u> , length)
PostingLists(<u>token</u> , <u>docID</u> , <u>offset</u> , postingDataEntry)
RPL(<u>token</u> , <u>iR</u> , <u>SID</u> , <u>docID</u> , <u>endPos</u> , rplDataEntry)

Fig. 2. The schemes of the tables *TReX* stores.

4 Exhaustive Retrieval Algorithm

In this section, we describe the exhaustive retrieval algorithm (*ERA*) for the retrieval phase of query evaluation. As explained in Section 3.1, the input to *ERA* consists of a list of sid’s and a list of terms. An element of a document in the corpus is considered *relevant*, if (1) it is in the extent of one of the given sid’s and (2) it contains at least one of the given terms. *ERA* finds all the relevant elements. In addition, for each relevant element e and for each term t among the given terms, *ERA* computes the frequency of t in e , (*i.e.*, the number of times that t appears in e). These term frequencies are the basis for ranking the elements of the result, as was discussed in Section 2. Note that *ERA* can be used not only with BM25 but also with any other ranking method that is based on term frequency.

For evaluating queries, *ERA* uses a structural summary of the corpus and inverted lists. An inverted list stores all the positions where each term appears. Positions are represented in *TReX* as pairs of a document identifier and an offset from the beginning of the document. Summaries and inverted lists are stored as indexed relational tables. The following section describes these tables, and in Section 4.2 we present *ERA*.

4.1 Data Structures

In *TReX*, the structural summary and the inverted lists are stored in two indexed tables named **Elements** and **PostingLists**. The schemes of these tables are shown in Figure 2. In the figure, keys are underlined. For each table, an index provides ordered sequential access to the tuples according to the keys.

The **Elements** table contains an entry for each element in the corpus. **SID** is the summary id of the element. The field **docID** holds the identifier of the document in which the element appears. The **endPos** is the position in the document where the element ends, and **length** is the length of the element. Note that we can compute the start position of each element by subtracting the length from the end position.

The **PostingLists** table is actually the inverted lists. For each term, all the positions where this term appears are stored in the table. The position of the term is represented by the identifier of the document in which the term appears and an offset from the beginning of this document. The **token** field is the token (*i.e.*, term) that the entry represents. In each tuple, the **postingDataEntry**

is a list of the form $doc_1, o_1^1, \dots, o_{i_1}^1, doc_2, o_1^2, \dots, o_{i_2}^2, \dots, doc_k, o_1^k, \dots, o_{i_k}^k$ where doc_1, \dots, doc_k is a sorted list of document identifiers, and each $o_1^j, \dots, o_{i_j}^j$ is a sorted list of offsets indicating the positions where the token appears in the document doc_j . The posting list may become too long for storing it in a single tuple. So it may be divided and stored across several tuples. In order to access the parts of the posting list in order of position, the fields `docID` and `offset` in `postingDataEntry` are part of the key.

For technical reasons, we also add a *maximal* dummy position denoted *m-pos* to the end of the last `postingDataEntry` list of each term. The position *m-pos* is maximal in the sense that no real position can exceed it. This is done to detect the end of each posting list.

4.2 The Exhaustive Algorithm

We now show how *ERA* computes a query result from the data in the `Elements` and `PostingLists` tables. The main code is presented in Figure 3. Before we explain the code, we describe the iterators used in *ERA*. There are two principle iterators; one for the `Elements` table and the second for the `PostingLists` table. The first iterator searches over the index of `Elements`. For a sid s , let iterator I_s return all the positions of relevant elements in s in ascending order of (`docID`, `endPos`). The function call $I_s.firstElement()$ returns the first tuple in `Elements` whose sid is equal to s . The function call $I_s.nextElementAfter(p)$ returns the element with the lowest position greater than p in extent s where p is a tuple of the form (`docID`, `endPos`). If no element is found then a dummy element is returned—an element with end position equal to *m-pos* and length equal to zero. The second iterator searches over the index of `PostingLists`. For a given term t , an iterator I_t over the posting list of t is created. It contains a single function $I_t.nextPosition()$ that successively returns the next position in the posting list of t .

We now explain the code of *ERA* given in Figure 3. The input to the algorithm consists of a list of sid's sid_1, \dots, sid_m and a list of terms t_1, \dots, t_n . The initialization of the algorithm involves creating variables for results and the necessary iterators. Lines 1 and 2 creates an empty list L to store the results of the computation and an array C of size $m \times n$ to keep intermediate count values of appearances of terms in elements. The purpose of C is to record for m different elements how many times each term among t_1, \dots, t_n has been seen in these elements. For each sid and term, iterators over `Elements` and `PostingLists` respectively, are constructed in lines 3–8 and the initial values from these iterators are stored in vectors e_i and pos_j , respectively.

After the initialization, the algorithm iterates over all the positions where one of the given terms appears. In each iteration, the lowest position not handled so far is being considered. We denote this position by pos_x and the term that it refers to by t_x . For the term t_x and each one of the elements that are currently being processed, the algorithm checks whether these elements contain t_x and updates C accordingly. More precisely, when an element e_i is being processed, it has three possible relationships with t_x , which we explain next.

$ERA((sid_1, \dots, sid_m), (t_1, \dots, t_n))$

Input: A list of sid's and a list of terms

Output: The relevant elements with their term frequencies

```

1: let  $L$  be a new empty list
2: let  $C[m][n]$  be an array of size  $m \times n$  having 0 in all the cells
3: for  $i = 1$  to  $m$  do
4:   create a new iterator  $I_{sid_i}$  over elements in the extent of  $sid_i$ 
5:    $e_i \leftarrow I_{sid_i}.firstElement()$ 
6:   for  $j = 1$  to  $n$  do
7:     create a new iterator  $I_{t_j}$  over the positions of  $t_j$ 
8:      $pos_j \leftarrow I_{t_j}.nextPosition(t_j)$ 
9:   repeat
10:    let  $x$  be the index for which  $pos_x = \min\{pos_1, \dots, pos_n\}$ , and let  $t_x$  be
    the term that starts in position  $pos_x$ 
11:    for  $i = 1$  to  $m$  do
12:      if  $pos_x < start(e_i)$  then
13:        {do nothing}
14:      else if  $start(e_i) < pos_x < end(e_i)$  then
15:         $C[i][x] \leftarrow C[i][x] + 1$ 
16:      else if  $end(e_i) < pos_x$  then
17:        if there is a non-zero cell in the row  $C[i][1, \dots, n]$  then
18:          create a new list  $tf_{e_i}$  from the  $n$  values  $C[i][1, \dots, n]$ 
19:          add  $(e_i, tf_{e_i})$  to  $L$ 
20:          reset all the cells  $C[i][1, \dots, n]$  to 0
21:           $e_i \leftarrow I_{sid_i}.nextElementAfter(pos_x)$ 
22:          if  $start(e_i) < pos_x < end(e_i)$  then
23:             $C[i][x] \leftarrow C[i][x] + 1$ 
24:           $pos_x \leftarrow I_{t_x}.nextPosition()$ 
25:    until for all the terms, the maximal position  $m-pos$  has been reached
26: return  $L$ 

```

Fig. 3. Retrieving the relevant elements.

If the element e_i starts after pos_x , then t_x is not contained in e_i and the counts in C should not be changed. Yet, at this point, term appearances in positions greater than pos_x may be inside e_i . Thus, e_i still needs to be processed. In this case, no action is being done (lines 12–13). If pos_x is between the start position of e_i and the end position of e_i then we encountered an appearance of t_x inside e_i . In this case, the counting in C is updated (lines 14–15).

If the element e_i ends before pos_x , then there is no need to change C . Furthermore, since all the following appearances of terms will be in a position greater than pos_x , at this point in the run, the counting of frequencies for e_i is complete and we can replace e_i with the next element from the extent of sid_i . If at least one of the term frequencies of e_i is greater than zero, then we add e_i and its frequencies to the list L (lines 17–20). We then replace e_i with the next element

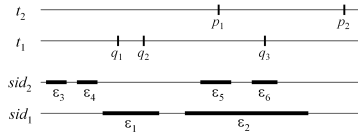


Fig. 4. Positions of elements in the extent of the sid's sid_1 , sid_2 and of appearances of the terms t_1 and t_2 .

in the extent of sid_i (line 21) and start the counting for this element. Note that the term being processed can be inside the new element and in this case we need to immediately update the counting for this new element (lines 22–23).

When the dummy maximal position has been reached for all terms, the computation is complete and L can be returned. *TReX* implements *ERA* using iterators so that relevant elements can be provided as soon as the computation of their term frequencies is complete. We do not provide the details in this paper. In post-processing, we compute the BM25 scores for the retrieved elements and sort them by their respective scores.

4.3 Relevance Posting Lists

ERA finds the relevant elements and, initializes them with their term frequencies, and sorts them by their end position. After computing the BM25 score of each element and sorting the elements by these scores, the result is stored because these results can be used to efficiently evaluate the query as a top- k query. *TReX* stores these results as *relevance posting lists (RPLs)* of the terms. An RPL of a term t is a list of elements that contain t , with each element's relevance score and sid. Elements in an RPL are sorted according to their relevance, in descending order. Rather than physically storing and maintaining many different lists, in *TReX*, all RPLs are stored in a single relation named `RPL`. The schema of this relation is shown in Figure 2. Each tuple in the `RPL` relation contains part of the RPL of some term t . The term t is stored in the `token` field, and the RPL (or a part of it) is stored in the `rplDataEntry` field. The field `rplDataEntry` holds a list of 5-tuples, where each 5-tuple identifies an element and consists of (1) a relevance score, (2) an sid, (3) a document identifier, (4) an offset to end position, and (5) a length. The elements in `rplDataEntry` are sorted in a decreasing order according to their relevance score. For each 5-tuple, the combination of sid, document identifier and offset-to-end are used as unique identifiers for elements. The attributes `iR`, `SID`, `docID` and `endPos` in `RPL` contain the values of the first element in `rplDataEntry` for ordering divided lists.

In *TReX*, given a list sid_1, \dots, sid_m of sid's and a list t_1, \dots, t_n of terms, RPLs can be used to efficiently compute top- k answers. Let t be one of the terms t_1, \dots, t_n . The top- k relevant elements w.r.t. t and sid_1, \dots, sid_m can be easily retrieved from the RPL of t by iterating over this RPL and selecting the top elements whose sid is among sid_1, \dots, sid_m . Note that the elements are provided

sorted by their rank. We can then use threshold algorithm (TA), similar to the one used in TopX [23], in order to combine for each element its scores in the n RPLs, and return the top- k answers. Note that this algorithm is a version of the TA algorithm proved by Fagin *et al.* [10] which is instance optimal in terms of the number of readings from the lists.

5 Experimental Results

We experimented with *TReX* in order to measure the efficiency and effectiveness of our retrieval methods. Two other goals of our experiments were to investigate the influence of using different summaries on the system’s performance and to compare the running time of *ERA* against TA. We implemented *TReX* in Java and used Berkeley DB (BDB) for the indexed tables. Our initial experiments were conducted over the IEEE collection provided in the INEX 2005 benchmark. This collection contains 16819 XML documents, and it has a size of 0.76GB. For the IEEE collection, the sizes of the tables `Elements` and `PostingLists`, stored in BDB, were 1.52GB and 8.05GB, respectively. Follow up experiments were conducted on the Wikipedia collection which contains approximately 645,719 documents and has a size of 5.01GB. The follow up experiments used the same basic configuration as was used for the IEEE collection.

Table 1. NEXI Queries and Translations for IEEE.

Query ID	NEXI Query		
203	sec[about(., code signing verification)]		
223	article[about(./sec, wireless ATM multimedia)]		
233	article[about (./bdy, synthesizers) and about (./bdy, music)]		
236	article[about(., machine translation approaches -programming)]		
260	bdy//*[about(., model checking state space explosion)]		
Query ID	Tag sid’s	Incoming sid’s	Keywords
203	6, 40	7, 46, 82, 89, 493, 607, 619, 630, 761, 1995, 2239	code, signing, verification
223	6, 40	7, 46, 82, 89, 493, 607, 619, 630, 761, 1995, 2239	wireless, ATM, multimedia
233	6,32	7,33	synthesizers, music
236	6	7	machine, translation, approaches
260	6, 32	7, 33	model, checking, state, space, explosion

We tested *TReX* on many INEX queries; however, we report here only the detailed results of five arbitrary queries from IEEE that seemed to us as representing the typical behavior of all the other queries. Similarly, the follow up results from five arbitrary queries from Wikipedia show that performance with a larger corpus was comparable to IEEE results. Table 1 shows the queries we chose and the translation of the IEEE queries for both the tag summary and incoming summary. Table 2 shows the queries we chose and the number of sid’s used in the query translations for the incoming summary.

Table 2. NEXI Queries and Number of sid's in Translations for Wikipedia

Query ID	NEXI Query	# of sid's
291	article//figure[about(., Olympian god goddess)]	1388
292	article//figure[about(., Renaissance painting Italian Flemish -French -German)]	1388
346	article[about(.,+unrealscript language api tutorial)]	4
356	article[about(.,natural language processing) and about(.,information retrieval)]	4
388	article[about(.,rhinoplasty)]	4

Table 3. Evaluation Time (in seconds) ERA Using Incoming and Tag Summaries for IEEE.

Query ID	Tag Summary	Incoming Summary	Efficiency Improvement
203	4873	1651	66%
233	1991	696	65%
236	5643	1812	68%
260	8860	1640	81%

Table 4. Evaluation Time (in seconds) TA Using Incoming Summary for IEEE.

Query ID	top10	top50	top100	top500	top1000	top1500
203	28	61	93	227	312	486
233	0.59	0.94	0.98	1	0.77	0.73
236	4	16	21	41	53	60
260	14	59	92	237	359	460

Table 5. Evaluation Time (in seconds) ERA Using Incoming for Wikipedia.

Query ID	Incoming Summary
291	1953
292	3435
346	1092
356	1283
388	637

The Wikipedia results in Table 5 were generated using incoming summaries with alias. The structure of the summary tree for Wikipedia is significantly larger and more complex than that of the IEEE corpus. The Wikipedia contains about 6 times more sid's than IEEE. The evaluation times for Wikipedia were in the same scale of magnitude as IEEE. The IEEE topics were structurally constrained to article bodies and article sections. Wikipedia queries 291 and 292 were constrained to figures in articles. Wikipedia queries 346, 356 and 388 were structurally constrained to articles. From these results, we conjecture that the factors in determining the running time of queries are the number of sid's considered, the size of the sid summary extents, and, most importantly, the number of matching tokens in the corpus.

Although we evaluated our queries on both the IEEE collection and Wikipedia, we measure the effectiveness of our retrieval techniques only on the IEEE collection. The results are presented in Tables 3 and 4. We compared our results to the results of other INEX participants. This is shown below in Figure 5. We ran the comparisons using the INEX Evaluation Package EvalJ[2]. In this comparison, recall and precision of query results are computed based on ranking performed by humans.

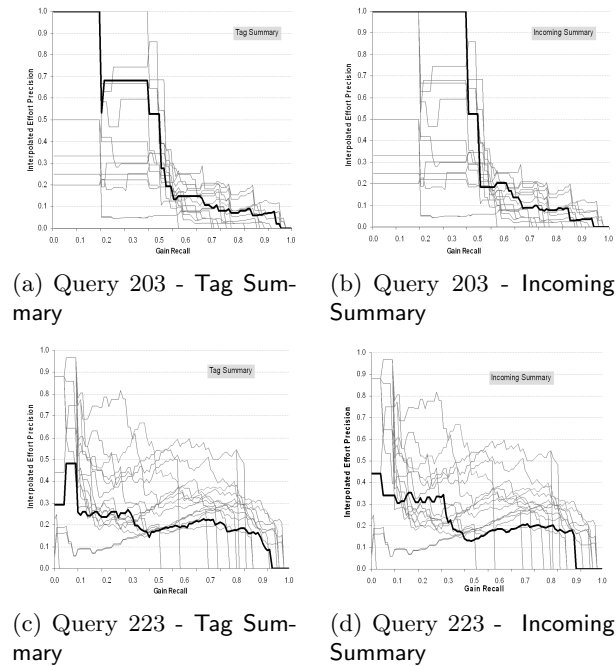


Fig. 5. Comparative of *TReX* effectiveness among other INEX 2005 participants.

Figure 5 shows the comparative effectiveness of two representative queries, Query 203 and Query 223 (listed in Table 1), using the tag summary and the incoming summary. The results of *TReX* are depicted with a bold line whereas the results of other INEX participants are depicted with light gray lines. Intuitively, each line shows the precision gained, as a function of the recall, for a single system. That is, a line of a system S going through a point (r, p) means that for a given k the top- k answers have a recall of r , and the precision of these k answers is p . The graphs in Figure 5 show that the incoming summary provides better results than the tag summary; however, the superiority of the incoming summary is not always the case. Note that, for Query 203, when using the incoming summary, 50% of the elements a human would include in the answer were given the highest scores by *TReX*, which means that they could be retrieved with 100% precision. Our tests suggest that the effectiveness of *TReX* is comparable to, and in many cases better than, the effectiveness of other systems that participated in INEX.

An important conclusion from our experiments is that summaries have a major influence on the efficiency and effectiveness of the system. Specifically, *TReX* had performed better with incoming summary than with tag summary. One explanation of this is that the tag summary does not take into account the ancestor-descendant relationship among elements, and thus, the partition it provides for the elements is coarser than the partition provided by the incoming summary. This means that every sid represents more elements, and so, more elements need to be processed by *ERA*. Also, using summaries causes query results to be less accurate because the structural constraints are evaluated in a flexible way that stems from the type of summary employed. We leave the question of how to choose an appropriate summary for future work.

6 Conclusion

In this paper we presented *TReX*—a system for efficient XML retrieval using summaries. The main contribution of our work is showing how to utilize summaries for a vague interpretation of structural constraints: either when all the answers to a query must be returned or when only the top- k answers are needed. We tested our retrieval algorithm on data and queries from INEX. The tests show that our retrieval method is efficient and effective. Our results provide a new and general perspective to structural evaluation in INEX. The flexibility and efficiency of the approach is coupled with a general framework so XML summaries can be easily incorporated into any XML retrieval system. Future work includes a study of the potential of using summaries for answering queries under a strict interpretation of the structural constraints. It also includes a study of the relationship between exhaustive retrieval and top- k query answering.

References

1. INEX: Initiative for the evaluation of XML retrieval. <http://inex.is.informatik.uni- duisburg.de:2005>, 2005.

2. EvalJ: INEX evaluation package. <http://evalj.sourceforge.net>, 2006.
3. S. Al-Khalifa, C. Yu, and H. V. Jagadish. Querying structured text in an XML databases. In *Proc. SIGMOD Conf.*, pages 4–15, 2003.
4. S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. TeXQuery: a full-text search extension to XQuery. In *Proc. WWW Conf.*, pages 583–594, 2004.
5. S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit. FleXPath: flexible structure and full-text querying for XML. In *Proc. SIGMOD Conf.*, pages 83–94, 2004.
6. A. Barta, M. P. Consens, and A. O. Mendelzon. Benefits of path summaries in an xml query optimizer supporting multiple access methods. In *Proc. VLDB Conf.*, pages 133–144, 2005.
7. J. Clark and S. DeRose. XML Path Language (XPath) version 1.0. <http://www.w3.org/TR/xpath>, 1999.
8. S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSearch: A semantic search engine for XML. In *Proc. VLDB Conf.*, pages 45–56, 2003.
9. M. P. Consens and T. Milo. Optimizing queries on files. In *Proc. SIGMOD Conf.*, pages 301–312, 1994.
10. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proc. PODS Conf.*, pages 102–113, 2001.
11. R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proc. VLDB Conf.*, pages 436–445, 1997.
12. L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked keyword search over XML documents. In *Proc. SIGMOD Conf.*, pages 16–27, 2003.
13. V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword proximity search on XML graphs. In *Proc. ICDE Conf.*, pages 367–378, 2003.
14. R. Kaushik, P. Bohannon, J. F. Naughton, and H. F. Korth. Covering indexes for branching path queries. In *Proc. SIGMOD Conf.*, pages 133–144, 2002.
15. R. Kaushik, R. Krishnamurthy, J. F. Naughton, and R. Ramakrishnan. On the integration of structure indexes and inverted lists. In *Proc. SIGMOD Conf.*, pages 779–790, 2004.
16. R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *Proc. ICDE Conf.*, pages 129–140, 2002.
17. A. Marian, S. Amer-Yahia, N. Koudas, and D. Srivastava. Adaptive processing of top-k queries in XML. In *Proc. ICDE Conf.*, pages 162–173, 2005.
18. T. Milo and D. Suciu. Index structures for path expressions. In *Proc. ICDT Conf.*, pages 277–295, 1999.
19. F. Rizzolo and A. O. Mendelzon. Indexing XML data with ToXin. In *Proc. WebDB Workshop*, pages 49–54, 2001.
20. S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proc. SIGIR Conf.*, pages 232–241, 1994.
21. M. L. S. Malik, G. Kazai and N. Fuhr. Overview of INEX 2005. In *Proc. INEX Workshop*, 2005.
22. T. Schlieder and H. Meuss. Querying and ranking XML documents. *Journal of the American Society for Information, Science and Technology, JASIST*, 53(6):489–503, 2002.
23. M. Theobald, R. Schenkel, and G. Weikum. An efficient and versatile query engine for TopX search. In *Proc. VLDB Conf.*, pages 625–636, 2005.
24. A. Trotman and B. Sigurbjornsson. Narrowed extended XPath I (NEXI). In *Proc. INEX Workshop*, pages 16–39, 2004.

Using Topic-shifts in XML Retrieval at INEX 2006

Elham Ashoori and Mounia Lalmas

Queen Mary, University of London
London, E1 4NS, UK
elham,mounia@dcs.qmul.ac.uk

Abstract. This paper describes the retrieval approaches used by Queen Mary, University of London in the INEX 2006 adhoc track. In our participation, we mainly investigate element-specific smoothing method within the language modelling framework. We adjust the amount of smoothing required for each XML element depending on its number of topic shifts to provide a focused access on Wikipedia collection.

1 Introduction

In this paper we describe the Queen Mary, University of London’s participation in the INEX 2006 adhoc track.

To score XML elements according to how exhaustive and specific they are given a query, various sources of evidence have been exploited. These include the content, the logical structure represented by the XML mark-up and the length of XML elements. In this work, we consider a different source of evidence, **the number of topic shifts in an XML element**. Our motivation stems from the definition of a relevant element at the appropriate level of granularity in INEX, which is expressed in terms of the “quantity” of topics discussed within each element. We therefore propose to use the number of topic shifts in an XML element, to express the “quantity” of topics discussed in an element as a mean to capture specificity.

For the Thorough task, we experiment with two different ways of smoothing, element-dependent smoothing and fixed approach within the language modeling framework. We incorporate the number of topic shifts in the smoothing process. We also compare topic shifts to element length, by incorporating each of them as prior probability of relevance in a retrieval setting and examining their effects on the effectiveness.

For the Focused task, we apply a post-filtering algorithm to remove overlapping elements. We follow similar approaches for thorough task and focused task to investigate the differences of the two tasks.

For the All In Context task, we took our focused runs, reordered the first 1500 elements in the list such that results from the same article are clustered together. For the Best In Context task, we investigate whether retrieving the most focused element in a relevant article as the best entry point is a useful approach.

The paper is organised as follows. In section 2, we define topic shifts and how we calculate it. Section 3 and 4 describe the methodology and the experimental setting used in our investigation. The experiments and results are discussed in Section 5. Section 6 concludes the paper.

2 Topic shifts

In this section, we describe how we measure the number of topic shifts of the elements forming a XML document. For this purpose, both the logical structure and a semantic decomposition of the XML document are needed. Whereas the logical structure of XML documents is readily available through their XML markup, their semantic decomposition needs to be extracted. To achieve that, we apply a topic segmentation algorithm based on lexical cohesion, TextTiling, which has been successfully used in several IR applications, is TextTiling¹ [1]. The underlying assumption of topic segmentation algorithms based on lexical cohesion, is that a change in vocabulary signifies that a topic shift occurs. This results in topic shifts being detected by examining the lexical similarity of adjacent text segments. TextTiling is a linear segmentation algorithm which considers the discourse unit to correspond to a paragraph and therefore subdivides the text into multi-paragraph segments.

The semantic decomposition of an XML document is used as a basis to calculate the number of topic shifts in each XML element forming that document. We consider that a topic shift occurs (i) when one segment ends and another segment starts, or (ii) when the starting (ending) point of an XML element coincides with the starting (ending) point of a semantic segment.

The *number of topic shifts* in an XML element e in document d is defined as:

$$score(e, d) := actual_topic_shifts(e, d) + 1 \quad (1)$$

where $actual_topic_shifts(e, d)$ are the actual occurrences of topic shifts in element e of document d . We are adding 1 to avoid zero values. For simplicity, when we refer to the number of topic shifts, we shall be referring to $score(e, d)$.

With the above definition, the larger the number of topic shifts – i.e. the larger the $score(e, d)$ – the more topics are discussed in the element, i.e. the content of element is less focussed with respect to the overall topic discussed in the element.

By considering the number of topic shifts occurring in an element instead of the number of topics discussed, we are able to distinguish the cases where the topic shift occurs not within the actual content of an element, but at its boundaries.

3 Retrieval Framework

For our experiments, we have implemented retrieval approaches for ranking XML elements based on a statistical language modelling approach [2]. Language mod-

¹ <http://elib.cs.berkeley.edu/src/texttiles/>

elling approaches have shown satisfactory results in content-oriented XML retrieval (e.g. [3, 4]). The language modelling approach allows us to combine “non-content” features of elements (or documents) (e.g. length, topic shifts) with the scoring mechanism.

If we estimate a language model for each element, then the relevance of an element e to a given query q is computed as how likely the query can be generated from the language model for that element. We rank elements based on the likelihood for a query $q = (t_1, t_2, \dots, t_n)$ to be generated from an element e as:

$$P(t_1, \dots, t_n|e) = P(e) * \prod_{i=1}^n (\lambda_e P(t_i|e) + (1 - \lambda_e) P(t_i|C)) \quad (2)$$

where

- t_i is a query term in q ,
- $P(e)$ is the prior probability of relevance for element e ,
- $P(t_i|e)$ is the probability of generating the query term t_i from element e ,
- $P(t_i|C)$ is the probability of query term t_i in the collection, and
- λ_e (element-dependent weight on the element language model) is a weighting parameter between 0 and 1 which is used in smoothing the element model with the collection model.

In Equation 2, $P(t_i|e)$ and $P(t_i|C)$ are defined as follows:

$$P(t_i|e) = \frac{tf(t_i, e)}{\sum_t tf(t, e)} \quad (3)$$

$$P(t_i|C) = \frac{ef(t_i)}{\sum_t ef(t)} \quad (4)$$

where $tf(t, e)$ is the number of occurrences of term t in element e , and $ef(t)$ is the total number of XML elements in which term t occurs.

We define the element-dependent smoothing parameter, λ_e , to be inversely proportional to the number of topic shifts in element e :

$$\lambda_e = \frac{\lambda}{topic\ shifts(e)} \quad (5)$$

where:

λ is a constant parameter between 0 and one (we consider $\lambda = 0.1$ for all the experiments).

We experiment with two different prior probabilities of relevance $P(e)$. First, we define the prior probability to be proportional to the length of an element which we refer to it as **length prior**:

$$P(e) = \frac{length(e)}{\sum_C length(e)} \quad (6)$$

, and second to be proportional to the number of topic shifts in an element which we refer to it as **topic shift prior**:

$$P(e) = \frac{\text{topic shifts}(e)}{\sum_C \text{topic shifts}(e)} \quad (7)$$

4 Retrieval Setting

For calculating the number of topic shifts in each XML elements, our first step is to decompose the Wikipedia XML documents into semantic segments through the application of TextTiling. We consider the discourse units in TextTiling to correspond to *paragraph* XML elements. We considered paragraph elements to be the lowest possible level of granularity of a retrieval unit. For the remainder of the paper, when we refer to the XML elements considered in our investigation, we will mean the subset consisting of paragraph elements and of elements containing at least one paragraph element as a descendant element.

Accordingly, the generated semantic segments can only correspond to paragraph elements and to their ancestors. As TextTiling requires a text-only version of a document, each XML document has all its tags removed and is decomposed by applying the algorithm to sequences of paragraphs. We set the TextTiling parameters to $W = 10$ and $K = 6$. A heuristic $W * K$ is equal to the average paragraph length (in terms of the number of terms).

After the application of TextTiling in the above data sets, we compute the number of topic shifts in elements.

In this work, only the title field of the CO queries is used. No stemming is applied. Elements with size smaller than 20 has been removed when indexing the Wikipedia collection. When we refer to the size or the length of an element, we mean the number of terms after removing stopwords. For each of the retrieval approaches, the top 1,500 ranked elements are returned as answers for each of the CO topics.

5 Experiments

Within the ad-hoc XML retrieval task we consider the following retrieval approaches:

5.1 Thorough Task

In our thorough task we experiment with two different ways of smoothing, element-dependent smoothing and fixed approach ($\lambda = 0.1$ for all elements). We also consider both length and the number of topic-shifts as prior probability of relevance. We submitted three runs:

- A run using the above language modelling approach with topic shift prior. We set $\lambda_e = 0.1$ (Lm.ToicShiftsPrior_T).

- A retrieval approach based on the above language modelling approach with a length prior. We set $\lambda_e = 0.1$ (Lm_LengthPrior_T).
- A retrieval approach based on the above language modelling approach with topic shift prior, where we use an element-dependent smoothing approach as discussed in Equation 5 (Lm_ToicShiftsPrior_TermWeighted_T).

Table 1. Thorough retrieval task: Evaluation based on Mean Average effort precision(MAep)

Approach	MAep
Lm_ToicShiftsPrior_T	0.0185
Lm_LengthPrior_T	0.0181
Lm_ToicShiftsPrior_TermWeighted_T	0.0163

The evaluation results with respect to the **system-oriented evaluation**, i.e. in terms of mean average effort-precision (MAep), are shown in Table 1. Focussing on the two approaches employing non-uniform priors, Lm_ToicShiftsPrior_T and Lm_LengthPrior_T, we observe that they perform comparably when evaluated using the system-oriented measures. This shows that the ability of both approaches in estimating the relevance of XML elements is almost the same.

We also investigate the element-specific smoothing approach in thorough task. Table 1 shows that using the element-dependent smoothing approach, Lm_ToicShiftsPrior_TermWeighted_T, slightly decreases the overall effectiveness compared to the fixed smoothing, Lm_ToicShiftsPrior_T.

5.2 Focused Task

The INEX 2006 Focused task asks systems to find the most focused elements that satisfy a (focused) information need, without returning “overlapping” elements.

In our focused task we aim at investigating the differences between the user-oriented Focused task and the system-oriented Thorough task. For this purpose, we submit similar approaches to our thorough runs. Overlap was removed by applying a post-filtering on the retrieved ranked list by selecting the highest scored element from each of the paths. In case of two overlapping elements with the same relevance score, the child element is selected. We submitted three runs:

- We removed overlapping elements from the Lm_ToicShiftsPrior_TermWeighted_T run (Lm_ToicShiftsPrior_TermWeighted_F).
- We removed overlap from the Lm_ToicShiftsPrior_T (Lm_ToicShiftsPrior_F).
- Since we expect that runs using element-based smoothing perform better for the Focused task, we submit a version of Lm_LengthPrior_T which use element-based smoothing (Lm_LengthPrior_TermWeighted_F).

The evaluation results with respect to the **user-oriented evaluation**, i.e. in terms of nxCG at three different early cut-off points (5, 10, 25, 50), are shown

Table 2. Focused retrieval task: normalised eXtended Cumulated Gain (nxCG) at different cut-off points

Approach	nxCG@5	nxCG@10	nxCG@25	nxCG@50
	Overlap On			
Lm.ToicShiftsPrior.TermWeighted.F	0.3455	0.2965	0.2351	0.1774
Lm.ToicShiftsPrior.F	0.3429	0.2953	0.2223	0.1649
Lm.LengthPrior.TermWeighted.F	0.3525	0.2957	0.2276	0.1708

in Table 2. Focussing on the two approaches employing element-based smoothing, Lm.ToicShiftsPrior.TermWeighted.F and Lm.LengthPrior.TermWeighted.F, we observe that both perform better than the approach using fixed smoothing, Lm.ToicShiftsPrior.F, when evaluated using the user-oriented measures. This shows that although using element-based smoothing hurts thorough task but improves focused task. This indicates that two tasks address different problems and needs different techniques.

Furthermore, using length prior at the very early rank (nxCG@5) improves the performance while using topic shifts prior seems more useful for the remaining cutoff points.

5.3 All In Context

For the ALL In Context task, we took our focused runs, reordered the first 1500 elements in the list such that results from the same article are clustered together. We submitted three runs:

- Lm.LengthPrior.TermWeighted.F_Clustered.R
- Lm.TopicShiftsPrior.F_Clustered.R
- Lm.TopicShiftsPrior.TermWeighted.F_Clustered.R

At the time of writing, none of the All In Context runs have been evaluated with an official INEX metrics.

5.4 Best In Context

For the Best In Context task, we examine whether the most focused element in a relevant document is a good choice as the best entry point in a relevant article. For this task we submitted three runs:

- We took our focused run, Lm.TopicShiftsPrior.TermWeighted.F, and return the elements with the maximum relevance score for each article as the best entry point (Lm.TopicShiftsPrior.TermWeighted.F_B).
- We took Lm.TopicShiftsPrior.F and return the elements with the maximum relevance score as the best entry point (Lm.TopicShiftsPrior.F_B)

- This run is slightly different from Lm_TopicShiftsPrior_TermWeighted_F_B such that in overlap-removal phase, in case of two overlapping elements with the same relevance score, the parent element is selected (Lm_TopicShiftsPrior_TermWeighted_Fparent_B).

We report the results using the EPRUM-BEP-Exh-BEPDistance and BEPD metrics at A=0.01 as shown in table 3. When evaluating these runs with EPRUM-BEP-Exh-BEPDistance at A=0.01, our runs ranked 2nd, 3rd and sixth. This shows that based on our approaches and using EPRUM-BEP-Exh-BEPDistance evaluation measure, the most focused elements are good estimates for the best entry points in relevant articles.

Table 3. Best In Context task: EPRUM-BEP-Exh-BEPDistance and BEPD metrics at A=0.01

Approach	@A=0.01
EPRUM-BEP-Exh-BEPDistance	
Lm_TopicShiftsPrior_TermWeighted_F_B	0.0325
Lm_TopicShiftsPrior_F_B	0.0314
Lm_TopicShiftsPrior_TermWeighted_Fparent_B	0.0300
BEPD	
Lm_TopicShiftsPrior_TermWeighted_F_B	0.1259
Lm_TopicShiftsPrior_TermWeighted_Fparent_B	0.1201
Lm_TopicShiftsPrior_F_B	0.1129

6 Discussion and summary

We described our approaches for the various adhoc tasks. Our main findings are that using element-specific smoothing within the language modeling framework depending on the number of topic shifts improves focused access to Wikipedia collection. We also found that the elements recognized as the most focused elements in our approaches are good estimates as the best entry points in the enclosing relevant articles. We further investigate using more sophisticated algorithms in incorporating the number of topic-shifts in the smoothing process in the future work.

References

1. M. A. Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 9–16, Morristown, NJ, USA, 1994. Association for Computational Linguistics.
2. D. Hiemstra. *Using Language Models for Information Retrieval*. Phd thesis, University of Twente, 2001.
3. J. Kamps, M. de Rijke, and B. Sigurbjörnsson. The importance of length normalization for XML retrieval. *Information Retrieval*, 8(4):631–654, 2005.

4. G. Ramirez, T. Westerveld, and A. P. de Vries. Using structural relationships for focused XML retrieval. In *Proceedings of the Seventh International Conference on Flexible Query Answering Systems (FQAS 2006)*. Springer, 2006.

Structured Content-Only Information Retrieval Using Term Proximity and Propagation of Title Terms

Michel Beigbeder

École Nationale Supérieure des Mines de Saint-Étienne
michel.beigbeder@emse.fr

1 Introduction

The needs for information retrieval are now quite well established and the tools have a large acceptance from the users. Though quite every documents are created with some structure in mind, the methods and tools are mainly dedicated to flat documents as opposed to structured documents.

Moreover most of the methods used for information retrieval on flat texts don't even take into account the basic structure of text: its linearity. In fact they are based on frequencies of terms (both in the documents and in the collection) and on the document lengths. Though there were some attempts to use the position of word occurrences in the text with either explicit proximity operators in the query language or ranking based on proximity of the query terms. These attempts are reviewed in section 2.

Concerning the logical structure which is the structure commonly referred to when speaking about structured documents, it is only quite recently that a sufficiently widespread representation for it is available so that large corpora of structured documents are available. So it is now possible to experiment in the large some of the ideas developed for structured information retrieval in the past and to design new methods.

We present in this paper an extension to structured documents retrieval of a proximity based method originally dedicated to flat texts. Our model can easily compute a score for any segment of text, in particular for any section or the whole document. First in section 3, we present the document model this method deals with, and in section 4 the method itself. In section 5 we present the experiments made within the INEX 2006 campaign.

2 Proximity use in flat document retrieval

The idea of using the proximity of the query keywords for retrieving flat documents was first implemented in boolean systems with a NEAR operator. This operator itself was an extension of the ADJ operator. These two operators can be used between keywords in a boolean query and its truth value is related to the positions of the two connected keywords. The NEAR operator evaluates to

true if the two terms appear within k words of each other (k is *one* for the ADJ operator).

The motivation for the ADJ (resp. NEAR) operator is to be able to describe in the query the needs for phrases (resp. loose phrases). These operators still are in use in tools used for searching in library catalogs. Though, from a technical point of view, they suffer from two handicaps that slowed down their use in plain text search engines. The first one is that they are closely linked to the boolean retrieval model which does not allow to rank the retrieved documents. The second one is that they do not fit well in the boolean query language model itself because they can only connect keywords and cannot be consistently extended to connect boolean sub-expressions.

More recent ideas for using keyword proximity were developed and they don't have these two limitations. Concerning the second one, the queries accepted by the query language model are either bags of terms or classic boolean expressions (only AND and OR operators). About the first one all the methods score the documents with respect to the positions of the keywords occurrences, taking into account their proximity. We will now describe the basis of some of these methods

2.1 Interval based methods

For their participation to the TREC-4 campaign, both Clarke *and al.* and Hawking *and al.* developed similar methods to rank text documents according to the proximity of the query keywords. The ideas are to select some intervals of text that contain all the keywords; to attribute a score to these intervals (the shorter the interval, the greater the score) and to sum up all these scores to score the document.

The two methods differ in the selected intervals: for Clarke *and al.* intervals cannot be nested because only the shortest ones are selected. For Hawking *and al.*, for each occurrence of any of the keywords, the shortest interval that contains all the keywords is selected. So if there are two successive occurrences of the same keyword without any occurrences of any other keyword in between, two nested intervals are selected.

The two methods also differ in the interval scoring, Clarke *and al.* chose a score that is roughly inversely proportional to the interval length and Hawking *and al.* chose a score roughly inversely proportional to the square root of the interval length.

The idea of using intervals was then revisited by Rasolofo *and al.* They chose to base their method on *Okapi* and they add an additional score to the *Okapi* probability. This additional score is based on the intervals containing any query terms pair: Each of the intervals shorter than a specified constant (6 in their experiments) that contains occurrences of two query terms contribute to this additional score.

2.2 Fuzzy influence function model

Beigbeder *et al.* developed a retrieval model based on *the fuzzy proximity of the keywords*. More precisely each occurrence of a keyword has a fuzzy influence on its neighbouring. This influence reaches its maximum value *one* at the keyword occurrence position and decreases with the distance to this position. The most simple function that have this behaviour is a triangle function. Moreover there is an easy to control parameter in such a function: its width, the length of the triangle basis. We will call k half of this length, it controls the range of the influence of an occurrence.

Given a term the influences of its occurrences are combined with a maximum operator. If the influence function is symmetrical, it consists in considering that at a given position the influence is determined by the nearest occurrence of the term.

Their query language model is that of the classical boolean model with AND, OR and NOT operators (neither NEAR nor ADJ). The influences of the query terms are combined in the query tree according to the fuzzy logic interpretation of the union and intersection operators

Let us consider an example with the document X X X X A X X X B X X X X X where there is an occurrence of the term A (resp. B) at position 5 (resp. 9) and where X denotes any term different from the terms A and B. Figure 1 shows the proximities to the terms A and B in this sample document (with $k = 5$) and their combination with a minimum corresponding to the AND operator.

Finally the score of a document is the summation of the influence function over all the positions in the text. It consists in evaluating the area under the curve associated to the root of the query tree. With our example, this is the area under the triangle of the curve A AND B.

This is this model that we extended to some kind of structured documents.

3 Our model of structured documents

Our work is pragmatic with respect to the structure of documents. We want to take into account the basic structure of many kinds of document models: nested sectionning and titles. This is the basis for scientific articles and technical documents but also for many more informal documents. We ignore any other structure, such lists and emphasis for instance. As a particular case, we consider that a document is the highest level in the sectionning hierarchy.

Another point is that sectionning and titles are tightly related so that in the L^AT_EX styles, only sectionning commands (`\section`, `\subsection`, ...) are available and the titles are given as parameters to these commands.

So the basis for our document model is the family of document which could be coded in the L^AT_EX styles with the sectionning commands only. Here is an example:

```
\title{title 1}           % highest level, level 0
                          % the document level and its title
```

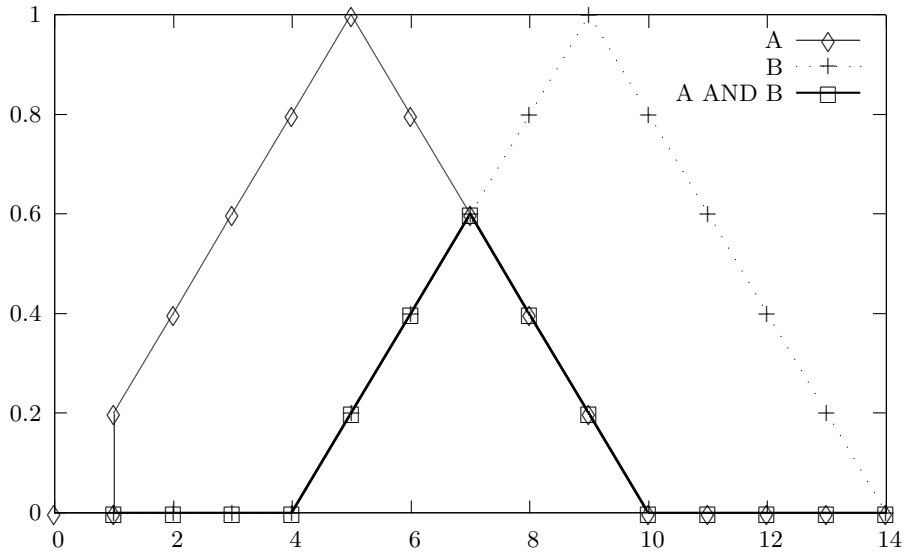


Fig. 1. Proximities to the terms A and B and their combination for the query A AND B: *x-axis*: position in the text, *y-axis*: fuzzy proximities.

```

bla bla           % level 0 text
  \section{title 2} % level 1 and its title
    bla bla       % level 1 text
      \subsection{title 3} % level 2 and its title
        bla bla   % level 2 text
      \section{title 4} % level 1 and its title
    bla bla       % level 1 text

```

This example can be coded in XML with:

```

<section><title>title 1</title>
bla bla
  <section><title>title 2</title>
    bla bla
      <section><title>title 3</title>
        bla bla
      </section>
    </section>
  <section><title>title 4</title>
    bla bla
  </section>
</section>

```

Formally the grammar of our document model is:

document = *section*

```

section          = '<section>''<title>' title_text '</title>'
                   section_content
                   '</section>'
section_content = (section_text | section)*

```

4 Influence of keywords occurrences

In the model presented in section 2.2, the influence of a term was only modeled for linear text. With our model of structured document, we have to modelize the influence of an occurrence of a query term depending on the structural part in which this occurrence does appear. As our document model is very simple there are only two cases: Occurrences can appear in *section_text* parts or *title_text* parts.

For a term occurrence which appears in the *section_text* parts, the basis is the same as in linear text: A decreasing value of the distance to the occurrence. But we add another constraint, the influence is limited to the *section_text* part in which the occurrence does appear.

Let us consider a document with the same text than the sample document of section 2.2 but with some structural tags: `<section> <title> X X X X A </title> X X X B X X X X X </section>`. The occurrence of the term B is in the *section_text* part of the section. Figure 2 shows the limitation of the triangle proximity to the term B in the document to the surrounding section.

For term occurrences which appear in the *title_text* parts their influence is extended to the full content of the section and recursively the subsections contained in the corresponding *section_content* part.

Considering our sample structured document the occurrence of the term A is in the *title_text* part of the section. Figure 2 shows the propagation of the influence of the occurrence of the term A that appears in the title to the whole section.

Otherwise like in the model presented in section 2.2, we use a boolean language query model and we combine the influence functions with `min` and `max` operators on the internal nodes of the boolean query tree. The basic score of a section is the summation of the influence function at the root of the query tree. We normalize this score by the maximum score reachable for this section. As the maximum value of the influence function is *one*. The maximum score simply is its length. Note that this maximum can actually be reached, for instance if all the query terms appear in the title of a section.

5 Experiments and implementation

5.1 Converting the documents to our document model

The documents of the Wikipedia collection used for the 2006 INEX campaign are written in XML. But the structure of the documents is more complex than that of our document model of section 3, as 1056 different tags are used.

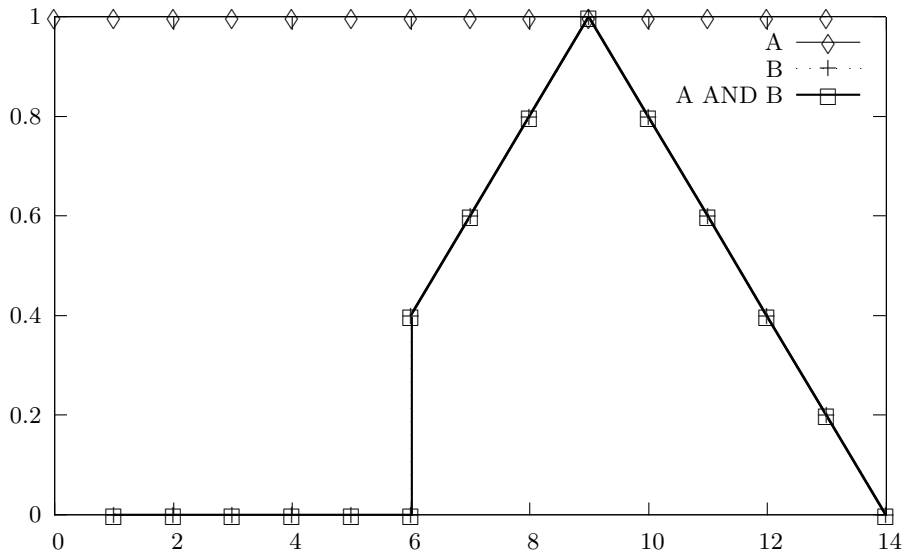


Fig. 2. Limitation of the influence of an occurrence to the *section_text* part in which it appears (proximity to the term B); propagation of a title term (proximity to the term A): *x-axis*: position in the text, *y-axis*: fuzzy proximities.

The main part of the conversion consists in keeping the text and the section and title tags with their corresponding closing tags. This can easily be done with an `xslt` processor, but some non obvious choices have to be made about the textual content, particularly concerning the spaces. Unfortunately, at the syntactic level no right choice can be made because of an inconsistent use of some tags. For instance, the document number 1341796 contains the following highlight (the attributes of the `collectionlink` tags are removed):

```
This is a
<emph3>List of
<collectionlink ...>poison</collectionlink>ings</emph3>in
alphabetical order of victim. It also includes confirmed attempted
and fictional poisonings. Many of the people listed here committed
or attempted to commit
<collectionlink ...>suicide</collectionlink>by
poison;
others were poisoned by others.
```

The question is to insert or not a space after the closing of the `collectionlink` tag. If a space is inserted, the following text is generated (mistake is emphasized):

```
This is a List of poison ings in alphabetical order of victim. It also
includes confirmed attempted and fictional poisonings. Many of the people
```

listed here committed or attempted to commit suicide by poison; others were poisoned by others.

which is correct for the second instance of the tag, but not for the first one. If no space are inserted, the following text is generated:

This is a List of poisonings in alphabetical order of victim. It also includes confirmed attempted and fictional poisonings. Many of the people listed here committed or attempted to commit *suicideby* poison; others were poisoned by others.

with the reverse correctness. Notice that a choice about spaces are to be made for each tag, and in the above examples, the `emph3` tags were replaced by spaces.

As no consistant choice could be made, we chose to insert space for each tag.

5.2 Indexation tool and index structure

We used as a basis for the indexation the tool LUCY in version 0.5.4. Though it is an outdated version that is now replaced by different versions of ZETTAIR, we had some experience with it as we extended it with an implementation of the model presented in section 2.2. It is a good basis because it keeps within its index the position of every occurrence of every term, and its lexical analyzer can recognize the syntax for any XML tags. At the indexation phase, we added the code necessary to keep track of the position and the nesting of the `section` and `title` tags. Remember that all other tags were removed in the previous step when the documents were converted to our model.

5.3 Building the queries

Queries could be automatically built with the conjunction of the terms that appear in the title field of the topics. As our method is highly selective, there would be very few results if any in the retrieved list of documents with such queries. So either the basic conjunctive queries or the retrieval procedure have to be relaxed in some way. Keeping these automatic conjunctive queries it is possible to enlarge the result set by using a lemmatization both in the indexation phase and in the query analysis. We didn't try this solution but we chose to build the queries manually.

With a basis of a conjunction of the terms found in the title field, sometimes, some terms were removed, but more often, these terms were expanded with disjunction of variations of the terms. These variations could simply be flexionnal ones (plural vs. singular) or derivational ones (verb, noun, adjective) or even semantic ones (synonyms or related concepts).

For instance, the title field of the topic number 289 is

emperor "Napoleon I" Polish

With a simple conjunction, the query could be (the '&' symbol is used for the boolean AND operator):

emperor & Napoleon & I & Polish

But some relaxation of it can be derived, for instance:

emperor & Napoleon & Polish
Napoleon & Polish
Napoleon & (Polish | Poland)

By using the description, narrative and ontopic_keywords fields, other queries can be formulated, for instance:

Napoleon & (Polish | Poland | Laczynska | Malewski | Poniatowski)

5.4 Runs

Given the queries and the value of the parameter k , our method is able to compute the fuzzy proximities to the query terms for each leaf of the query tree and to combine these influences up to the root of the tree. We used two query sets and two values of k : 50 and 200.

Then a score can be computed for any span in the document. We computed scores for the whole documents and all its sections and subsections recursively. This was our participation to the THOROUGH task.

For the BEST IN CONTEXT task we searched for the maximum of the influence function at the root of the query tree and returned the section of highest level which contained the position at which this maximum was reached.

Finally for the FOCUSED task, the different parts of the documents were sorted with two keys: first, the score of the document to which it belongs; and then its own score.

Influence Diagrams and Structured Retrieval: Garnata implementing the SID and CID models at INEX'06

Luis M. de Campos, Juan M. Fernández-Luna,
Juan F. Huete, and Alfonso E. Romero

Departamento de Ciencias de la Computación e Inteligencia Artificial
E.T.S.I. Informática y de Telecomunicación, Universidad de Granada,
18071 – Granada, Spain
{lci,jmfluna,jhg,aeromero}@decsai.ugr.es

Abstract. This paper presents the results of our participation in INEX'06. Two runs were submitted to the Ad Hoc Thorough track obtained with Garnata, and Information Retrieval Model for structured documents, and implementing two different models based on Influence Diagrams, the SID and CID models. The result of this first participation has been very poor. In the paper, we describe the models, the system, and analyse the possible reason of such a bad performance.

1 Introduction

Although the research group “Uncertainty Treatment in Artificial Intelligence” at University of Granada has been participating in INEX since its beginnings, this year it is the first time that their members submit a run to the official tasks. Until now, our contribution to INEX has been the design of several topics and the assessments of relevance judgements, but in this new edition we have participated with a new experimental platform to perform structured retrieval using Probabilistic Graphical Models.

We have participated in the Adhoc Track (Thorough) with the retrieval results of two models based on Influence Diagrams [5]: the Simple Influence Diagram Model (SID Model) and the Context-based Influence Diagram Model (CID Model) [2, 3]. These models have been implemented in the Garnata Retrieval System [4], a software specifically designed and implemented to work with Probabilistic Graphical Model-based structured retrieval models, like Bayesian Networks and Influence Diagrams [7].

It is also fair to comment that the results of this first participation are not good. They are clearly disappointing. In fact, we are in the last positions of the ranking in the Thorough track. The main reason of this behaviour could be due to a software bug that we have found in Garnata after studying the bad performance obtained by the SID and CID models once the official results were published. In the moment of writing this paper, we are re-evaluating both models in order to know the correct values of the official INEX evaluation measures. The

new results will be detailed in the INEX workshop presentation and published in the final proceedings.

Another reason to get such a so bad position in the ranking could be that the parameters that we used in the experimentation were not the best, because they were selected in a hurry as the deadline to submit the results was approaching and we had not completely finished the development of the software. We think that the performance of both models could be clearly improved with a more systematic experimentation.

In order to describe the models and the software that we have used in this edition, this paper is organised as follows: The next section will introduce the reader the formalism of the Influence Diagrams. Sections 3 and 4 will describe the SID and CID models, and Garnata, the Information Retrieval System, which implements them, respectively. The following section will discuss how the experimentation was performed, and try to explain the reasons of the bad performance of the models. Finally, this paper will finish with the conclusions and future works.

2 Introduction to Influence Diagrams

An Influence Diagram [5, 9] provides a simple notation for creating decision models by clarifying the qualitative issues of the factors which need to be considered and how they are related, i.e. an intuitive representation of the model. It also has associated an underlying quantitative representation in order to measure the strength of the relationships: we can quantify uncertain interactions between random variables and also the decision maker's options and preferences. The model is used to determine the optimal decision policy. More formally, an influence diagram is an acyclic directed graph containing three types of nodes (decision, chance, and value nodes) and two types of arcs (influence and informative arcs).

Nodes in an influence diagram represent various types of variables.

- **Decision nodes:** usually drawn as rectangles, these represent variables that the decision maker controls directly. These variables model the decision alternatives available for the decision maker.
- **Chance nodes:** usually drawn as circles, these represent random variables, i.e. uncertain quantities that are relevant to the decision problem and cannot be controlled directly. They are quantified by means of conditional probability distributions, identical to those used in Bayesian networks¹. Predecessors (parents) of chance nodes that are decision nodes act in exactly the same way as those predecessors that are chance nodes – they index the conditional probability tables of the child node.
- **Utility nodes:** usually drawn as diamonds, these represent utility, i.e. they express the profit or the preference degree of the consequences derived from

¹ In fact, the subset of an influence diagram that consists only of chance nodes is a Bayesian network, i.e., an influence diagram can also be viewed as a Bayesian network enlarged with decision and utility nodes.

the decision process. They are quantified by the utility of each of the possible combinations of outcomes of their parent nodes.

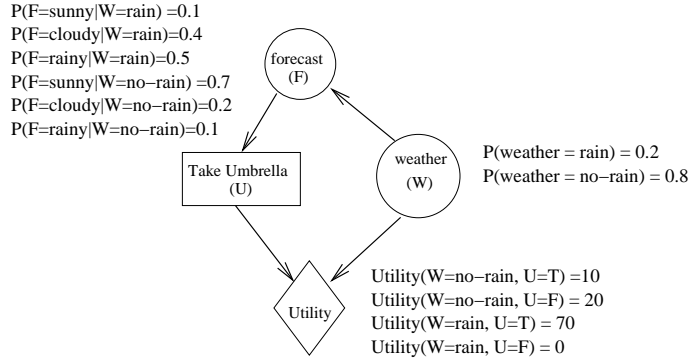


Fig. 1. An example of an Influence Diagram.

There are also different types of arcs in an influence diagram, which generally represent influence. The arcs between chance nodes represent probabilistic dependencies (as occurs in Bayesian networks). The arcs from a decision node to a chance node or to a utility node establish that the future decision will affect the value of the chance node or the profit obtained, respectively. Arcs between a chance node and a decision node (also called *informative*) only say that the value of the chance node will be known at the moment of making the decision. Finally, arcs from a chance node to a utility node will represent the fact that the profit depends on the value that this chance node takes. The absence of an arc between two nodes specifies (conditional) independence relationships. It should be noted that the absence of an arc is a stronger statement than the presence of an arc, which only indicates the possibility of dependence.

Some arcs in influence diagrams clearly have a causal meaning. In particular, a directed path from a decision node to a chance node means that the decision will influence that chance node, in the sense of changing its probability distribution.

A simple example of an influence diagram appears in Figure 1. It has two chance nodes, F and W, representing, the weather forecast in the morning (sunny, cloudy or rainy), and whether it actually rains during the day (rain or no-rain), respectively. It has one decision node U, take an umbrella (with possible values true or false). The utility node measures the decision maker's satisfaction.

With each chance node X in the graph, the quantitative part of an influence diagram associates a set of conditional probability distributions $p(X|pa(X))$, one for each *configuration* $pa(X)$ from the *parent set* of X in the graph, $Pa(X)$, i.e. for each allocation of values to all the variables in the parent set of X .

If X has no parents ($Pa(X) = \emptyset$), then $p(X|pa(X))$ equals $p(X)$. For each utility node V , a set of utility values $v(pa(V))$ is associated, specifying for each combination of values for the parents of V , a number expressing the desirability of this combination for the decision maker.

The goal of influence diagram modeling is to choose the decision alternative that will lead to the highest expected gain (utility), i.e. to find the *optimal policy* [8]. In order to compute the solution, for each sequence of decisions, the utilities of its uncertain consequences are weighted with the probabilities that these consequences will occur.

3 The SID and CID models

In this section, we shall briefly describe the SID and CID models for structured retrieval. A complete description of these models can be found in [2, 3].

We start with a document collection containing M documents, $\mathcal{D} = \{D_1, \dots, D_M\}$, and the set of the *terms* used to index these documents (the glossary of the collection). We assume that each document D_i is organized hierarchically, representing structural associations of elements in D_i , which will be called *structural units*. Each structural unit is composed of other smaller structural units, except some ‘terminal’ or ‘minimal’ units which are indivisible, they do not contain any other unit. Instead, these are composed of terms: each term used to index the complete document D_i will be assigned to all the terminal units containing it. Conversely, each structural unit, except the one corresponding to the complete document, is included in only one structural unit.

These models are Influence Diagrams, so they are based on an underlying Bayesian networks which represent a structured document set. The Bayesian network will contain two kinds of nodes, representing the terms and the structural units. The former will be represented by the set $\mathcal{T} = \{T_1, T_2, \dots, T_l\}$. There are two types of structural units: *basic structural units*, those which only contain terms, and *complex structural units*, that are composed of other basic or complex units. The notation for these nodes is $\mathcal{U}_b = \{B_1, B_2, \dots, B_m\}$ and $\mathcal{U}_c = \{S_1, S_2, \dots, S_n\}$, respectively. Therefore, the set of all structural units is $\mathcal{U} = \mathcal{U}_b \cup \mathcal{U}_c$. In this paper, T or T_k will represent a term; B or B_i a basic structural unit, and S or S_j a complex structural unit. Generic structural units (either basic or complex) will be denoted as U_i or U . Each node T , B or S has associated a binary random variable, which can take its values from the sets $\{t^-, t^+\}$, $\{b^-, b^+\}$ or $\{s^-, s^+\}$ (the term/unit is not relevant or is relevant), respectively. A unit is relevant for a given query if it satisfies the user’s information need expressed by this query. A term is relevant in the sense that the user believes that it will appear in relevant units/documents.

Regarding the arcs of the model, there is an arc from a given node (either term or structural unit) to the particular structural unit node it belongs to, expressing the fact that the relevance of a given structural unit to the user will depend on the relevance values of the different elements (units or terms) that

comprise it. It should be noted that with this criteria, terms nodes have no parents.

It should be noticed that the hierarchical structure of the model determines that each structural unit $U \in \mathcal{U}$ has only one structural unit as its child, the unique structural unit containing U (except for the leaf nodes, i.e. the complete documents, which have no child). We shall denote indistinctly by $Hi(U)$ or $U_{hi(U)}$ the single child node associated with node U (with $Hi(U) = null$ if U is a leaf node).

The numerical values for the conditional probabilities have also to be assessed: $p(t^+)$, $p(b^+|pa(B))$, $p(s^+|pa(S))$, for every node in \mathcal{T} , \mathcal{U}_b and \mathcal{U}_c , respectively, and every configuration of the corresponding parent sets ($pa(X)$ denotes a configuration or instantiation of the parent set of X , $Pa(X)$). A canonical model proposed in [1] will be used to represent the conditional probabilities which supports a very efficient inference procedure. These probabilities are defined as follows:

$$\forall B \in \mathcal{U}_b, p(b^+|pa(B)) = \sum_{T \in R(pa(B))} w(T, B), \quad (1)$$

$$\forall S \in \mathcal{U}_c, p(s^+|pa(S)) = \sum_{U \in R(pa(S))} w(U, S), \quad (2)$$

where $w(T, B)$ is a weight associated to each term T belonging to the basic unit B , $w(U, S)$ is a weight measuring the importance of the unit U within S . In any case $R(pa(U))$ is the subset of parents of U (terms for B , and either basic or complex units for S) relevant in the configuration $pa(U)$, i.e., $R(pa(B)) = \{T \in Pa(B) | t^+ \in pa(B)\}$ and $R(pa(S)) = \{U \in Pa(S) | u^+ \in pa(S)\}$. These weights can be defined in any way, the only restrictions are that $w(T, B) \geq 0$, $w(U, S) \geq 0$, $\sum_{T \in Pa(B)} w(T, B) \leq 1$, and $\sum_{U \in Pa(S)} w(U, S) \leq 1$.

Once the Bayesian network has been constructed, it is enlarged by including decision and utility nodes, thus transforming it into an influence diagram.

- **Decision nodes:** these nodes model the decision variables, representing the possible alternatives available to the decision maker. One decision node, R_i , for each structural unit $U_i \in \mathcal{U}$. R_i represents the decision variable related to whether or not to return the structural unit U_i to the user. The two different values for R_i are r_i^+ and r_i^- , meaning ‘retrieve U_i ’ and ‘do not retrieve U_i ’, respectively.
- **Utility nodes:** we shall also consider one utility node, V_i , for each structural unit U_i . V_i will measure the value of utility of the corresponding decision.

We shall also consider a utility node that represents the joint utility of the whole model. This node will be denoted by Σ , representing the fact that we are assuming an additive behavior of the model.

In addition to the arcs between chance nodes (already present in the Bayesian network), a set of arcs pointing to utility nodes are also included, employed to indicate which variables have a direct influence on the desirability of a given

decision, i.e. the profit obtained will depend on the value of these variables. We shall consider two different set of arcs, which will consistently generate two different influence diagrams models:

1. *Simple Influence Diagram (SID)*: we shall only take into account arcs from chance nodes U_i and decision nodes R_i to the utility nodes V_i . These arcs mean that the utility function of V_i obviously depends on the decision made and the relevance value of the structural unit considered.
Finally, the utility node Σ has all the utility nodes V_i as its parents. These arcs represent the fact that the joint utility of the model will depend on the values of the individual utilities of each structural unit.
2. *Context-based Influence Diagram (CID)*: In order to represent that the utility function of V_i obviously depends on the decision made and the relevance value of the structural unit considered, we use arcs from each chance node U_i and decision node R_i to the utility node V_i . Another important set of arcs are those going from $Hi(U_i)$ to V_i , which represent that the utility of the decision about retrieving the unit U_i also depends on the relevance of the unit which contains it (obviously, for the units which are not contained in any other unit these arcs do not exist).
The utility node Σ will have the same set of parents as in the SID model.

Figure 2 shows an example of both influence diagram models: the SID (left-hand side) and the CID (right-hand side).

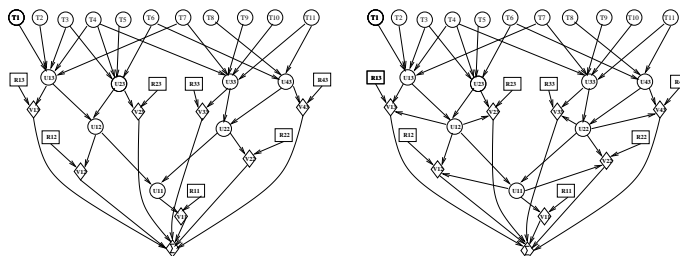


Fig. 2. SID and CID models.

Finally, for each node V_i , the associated utility functions must be defined.

1. *Utility nodes in SID*: for each node V_i , we need to assess a numeric value that represents the utility for the corresponding combination of the decision node R_i and the chance node representing the structural component U_i : $v(r_i^+, u_i^+)$, $v(r_i^-, u_i^+)$, $v(r_i^+, u_i^-)$ and $v(r_i^-, u_i^-)$.
2. *Utility nodes in CID*: for each utility node V_i we need eight numbers, one for each combination of values of the decision node R_i and the chance nodes U_i and $Hi(U_i)$ (except for the leaf nodes, which only require four values). These

values are represented by $v(r_i, u_i, u_{hi(U_i)})$, with $r_i \in \{r_i^-, r_i^+\}$, $u_i \in \{u_i^-, u_i^+\}$, and $u_{hi(U_i)} \in \{u_{hi(U_i)}^-, u_{hi(U_i)}^+\}$.

3.1 Inference and Decision Making

To solve an influence diagram, the expected utility of each possible decision (for those situations of interest) has to be computed, thus making decisions which maximize the expected utility. In our case, the situation of interest corresponds with the information provided by the user when he/she formulates a query. Let $\mathcal{Q} \subseteq \mathcal{T}$ be the set of terms used to express the query. Each term $T_i \in \mathcal{Q}$ will be instantiated to either t_i^+ or t_i^- ; let q be the corresponding configuration of the variables in \mathcal{Q} . We wish to compute the expected utility of each decision given q . As we have assumed a global additive utility model, and the different decision variables R_i are not directly linked to each other, we can process each one independently. The expected utilities for each U_i can be computed for each model by means of:

– *SID Model*:

$$EU(r_i^+ | q) = \sum_{u_i \in \{u_i^-, u_i^+\}} v(r_i^+, u_i) p(u_i | q). \quad (3)$$

$$EU(r_i^- | q) = \sum_{u_i \in \{u_i^-, u_i^+\}} v(r_i^-, u_i) p(u_i | q). \quad (4)$$

– *CID Model*:

$$EU(r_i^+ | q) = \sum_{\substack{u_i \in \{u_i^-, u_i^+\} \\ u_{hi(U_i)} \in \{u_{hi(U_i)}^-, u_{hi(U_i)}^+\}}} v(r_i^+, u_i, u_{hi(U_i)}) p(u_i, u_{hi(U_i)} | q). \quad (5)$$

$$EU(r_i^- | q) = \sum_{\substack{u_i \in \{u_i^-, u_i^+\} \\ u_{hi(U_i)} \in \{u_{hi(U_i)}^-, u_{hi(U_i)}^+\}}} v(r_i^-, u_i, u_{hi(U_i)}) p(u_i, u_{hi(U_i)} | q). \quad (6)$$

In the context of a typical decision making problem, once the expected utilities are computed, the decision with greatest utility is chosen: this would mean to retrieve the structural unit U_i if $EU(r_i^+ | q) \geq EU(r_i^- | q)$, and not to retrieve it otherwise. However, our purpose is not only to make decisions about what to retrieve but also to give a ranking of those units. The simplest way to do it is to show them in decreasing order of the utility of retrieving U_i , $EU(r_i^+ | q)$.

A detailed description of how to compute the posterior probabilities required in these previous equations can be found in [2, 3], but only to mention that the specific characteristics of the canonical model used to define the conditional probabilities will allow us to efficiently compute the posterior probabilities in the following way:

$$\forall B \in \mathcal{U}_b, p(b^+|q) = \sum_{T \in Pa(B) \setminus Q} w(T, B) p(t^+) + \sum_{T \in Pa(B) \cap R(q)} w(T, B). \quad (7)$$

$$\forall S \in \mathcal{U}_c, p(s^+|q) = \sum_{U \in Pa(S)} w(U, S) p(u^+|q). \quad (8)$$

4 Garnata: an Information Retrieval System for Structured Documents

Garnata was born as an implementation completely adapted to the models based on the above Probabilistic Graphical Models to retrieve structured documents, although other models following the same philosophy could be easily implemented in it. Written in C++, following the object-oriented paradigm, it offers a wide range of classes and a complete set of utility programs. It implements the SID and CID models.

It is able to manage different collections, and different indexes over the same collection. It can choose among different stopword lists (previously inserted into the system) and use (if desired) Porter’s stemming algorithm.

In our models, several valid weighting schemes could exist because of its experimental nature. As a consequence, in Garnata, indexing does not compute the weights (setting all of them to be zero). Instead of that, we have added the possibility to calculate weights (following a certain weighting scheme) for previously built indexes without inserting into them, and store them in files – the so-called *weight files*–. So, records of that precomputed weight files are kept in order to provide a fast way to insert one into the index itself to retrieve with it.

To store textual information (terms and identifiers of the final units where they appear), we use inverted indexes [6]. While the lexicon is kept entirely in memory (both while indexing and querying), the list of occurrences is read from disk. We use another file to write the list of relative positions of each term inside a unit in order to answer queries containing proximity operators or phrases (although in the current stage of Garnata, they are not used to formulate a query).

To maintain information about the structural units, we use one direct access file, except for the XPath routes, which are stored separately. Other files keep relations among units, being accessible with only two disk reads. So a large file contains data of each unit itself (identifier, tag, container, position, ...) and besides, we can easily manage the following relationships with two disk accesses (essential for our models):

- Given a non-final unit, returning the list of identifiers of the units that it contains.
- Given a final unit, returning the container unit and, recursively, all the containers until a leaf unit is found.

- Given a final unit, returning the list of contained terms (known as the *direct index*).

The Garnata’s indexing subsystem also implements file compression.

Querying subsystem is the most critical part of an IR system. In our case, we have built structures at indexing time to reduce at maximum the amount of disk accesses while processing a query, in order to save time and give a short response time. The algorithm for achieving this task comprises the following steps (not necessarily in this order):

1. Query is parsed, and occurrences of the component terms are retrieved from disk.
2. For each occurrence, implied final units are read into memory (if not already there).
3. For each final unit, its descendants are read into memory (if not already there).
4. Propagation is carried out, units are sorted by its probability of relevance, and the result is returned.

The first big bottleneck to be minimized is due to the reading from disk of the unit objects (containing information about each unit). We will keep two unit caches in memory: the first one, containing final units, and the second one, containing complex units. Both will be *static caches*, meaning that they will not change the unit stored in each cache slot. Cache is accessed doing a hash function-like scheme, so for each cache slot, we will have several candidates (those identifiers being the hash inverse of the slot identifier).

For the final units cache, in each slot, we will store the unit containing greater number of terms (among the candidates). For the complex units cache, in each slot, we will store the unit containing more final units. These two heuristics has shown very good time performance in our experiments.

The paper [4] contains a more detailed description of Garnata.

5 Experimental Setting for INEX’06

In this section, we shall describe the conditions under we have performed the two runs submitted to the Thorough track.

First of all, the weighting scheme used in equations 7 and 8 to compute posterior probabilities has been basically a normalized tf-idf scheme for weights of terms in basic units. On the other hand, the weights of units included in unit, U_i , measure to a certain extent, the proportion of the content of the unit U_i which can be attributed to each one of its components. A detailed explanation of how they are computed is shown in [2].

With respect to the prior probabilities of relevance of the terms, $p(t^+)$, they can also be defined in any reasonable way, for example an identical probability for all the terms, $p(t^+) = p_0, \forall T \in \mathcal{T}$, as proposed in [2], specifically, 0.5.

Regarding the utilities for the SID model, for a given unit U_i , the best situation is clearly for a relevant unit to be retrieved, and the worst situation, for a relevant unit to be hidden. We therefore fix $v(r_i^+, u_i^+) = 1$ and $v(r_i^-, u_i^+) = 0$. The other two are also set to 0: $v(r_i^+, u_i^-) = 0$ and $v(r_i^-, u_i^-) = 0$.

In the context of the CID model, we need to assess eight utility values. Once again following the ideas set out in the previous paragraph, two of these will be fixed: $v(r^+, u^+, w^-) = 1$ and $v(r^-, u^+, w^-) = 0$. But as the ranking criteria has been only the expected utility of retrieving, $EU(r_i^+|q)$, those utilities related to not retrieving (v^-) are not taken into account². Therefore, the utility values are those in Table 5.

v_{+-}^-	v_{+-}^+	v_{-+}^+	v_{++}^+	v_{+-}^-	v_{-+}^-	v_{--}^-	v_{+-}^+
-	0.0	0.0	0.0	-	-	-	1.0

Table 1. Utility configuration for the CID model.

In the first row of this table, the superscript is related to the value that the node R_i is taking. The subscripts refer to the value that units U and W are taking respectively at the same time.

The choice in the case of the SID model is very clear: we retrieve a unit when it is relevant. In the CID model, we retrieve a unit when it is relevant and the unit where it is contained is not relevant. We could say that these values are the default vectors of utilities for both models, reason why we selected them to run our experiments. We have to recognise that these are really strong restrictions and could be relaxed obtaining more appropriate values of the different utilities, but we could not do it due to the lack of time as we were finishing the development of Garnata when the submission deadline went off.

We also have considered a different information source to define the utility values: the query itself. We could say that a given structural unit U_i will be more useful (with respect to a query Q) as more terms indexing U_i also belong to Q . More formally, let us consider the sum of the inverted document frequencies of those terms indexing a unit U_i , $A(U_i)$, that also belong to the query Q , normalized by the sum of the idfs of the terms contained in the query ($nidf_Q(U_i)$):

$$nidf_Q(U_i) = \frac{\sum_{T_k \in A(U_i) \cap Q} idf_k}{\sum_{T_k \in Q} idf_k} \quad (9)$$

These values $nidf_Q(U_i)$ will be used as a correction factor of the previously defined utility values, for each utility node V_i :

$$\begin{aligned} v^*(r_i, u_i) &= v(r_i, u_i) \cdot nidf_Q(U_i) \\ v^*(r_i, u_i, u_{hi(U_i)}) &= v(r_i, u_i, u_{hi(U_i)}) \cdot nidf_Q(U_i) \end{aligned} \quad (10)$$

² This is also applied to the utilities in the SID model

With these experimental settings, according to the results published in the evaluation section of the INEX'06 website³, considering “Metric:ep-gr, Quantization: gen, Overlap=off”, we have obtained a mean of effort-precision of 0.004 with the runs of obtained by Garnata for the two models, occupying the disappointing 97th and 98th positions.

Studying the reasons of this bad behaviour, we discovered a bug in the software, by which instead of retrieving the i^{th} unit, we were returning the $i + 1^{th}$. Basically, the software returned a completely different unit to that which should have been returned. This problem could partially justify the awful results that we have obtained. At the moment of writing this paper, we are performing the retrieval and the list of retrieved units is being created. Later the new evaluation with EvalJ will be performed, so we shall know the true measures. These will be made public in the workshop and later in the final proceedings.

Then, as well as the bug, possible reasons that could explain this low performance would be:

- As mentioned above, a bad selection of the configurations for the utilities in both models. As an example, the value v_{++}^+ is set to 0.0 when it should be set to 1.0, meaning that the utility of retrieving a relevant unit inside a relevant unit is the maximum. The original value would be the correct if we were participating in the 'Best in Context' task where we are looking for best entry points.
- Poor processing of the query, because we consider it as a bag of words, without taking into account interesting features as proximity or exact patterns, among others.
- The weighting scheme of the units that we have used could not be enough. So a more sophisticated scheme where we could take into account the type of unit (for example, titles) could improve the results. Therefore, the unit weights would not depend only on the portion of the text that they would contain but also of the importance of the unit itself.

6 Conclusions and Future Works

In this paper, we have presented the SID and CID models, and the Information Retrieval System, which implements them, Garnata. These are the retrieval tools that we have used to participate in this our first edition of INEX, in the Thorough track.

The results are very disappointing as we are at the bottom of the raking. One of the reasons could be a bug in the software at the moment of adapting to work with Wikipedia, but other could be a bad selection of the parameters of the models (above all, the configurations of utilities that we selected are very strict). We are re-evaluating, once we corrected the mistake, so in brief we shall have the correct measures.

³ <http://inex.is.informatik.uni-duisburg.de/2006/adhoc-protected/results/thorough/Thorough.html>

With respect to the future works, we intend to have a detailed experimentation with the IEEE and Wikipedia collections, in order to find automatically those values for the utility configurations which perform best. We think that selecting correctly the utilities there is room for improvement. Also we want to endow the system with a powerful query processor, so we can use CAS topics.

For next edition of INEX, we plan to participate in other tasks, such as 'Focused' or 'Best in Context'. In the case of this last task, we think that our models, as the underlying formalism, influence diagram, is specially designed to make decisions considering the context, can perform acceptably.

Also, we are developing new models based in these Probabilistic Graphical Models, which improve the performance of the SID and CID models.

With these actions, we hope having a better performance, and therefore, improving the position in ranking of the next edition of INEX forum.

Acknowledgments. This work has been jointly supported by the Spanish Ministerio de Educación and Ciencia, and Junta de Andalucía, under projects TIN2005-02516 and TIC276, respectively.

References

1. L.M. de Campos, J.M Fernández-Luna, and J.F. Huete. The BNR model: foundations and performance of a Bayesian network-based retrieval model. *International Journal of Approximate Reasoning*, 34:265–285, 2003.
2. L.M. de Campos, J.M. Fernández-Luna, and J.F. Huete. Using context information in structured document retrieval: An approach using influence diagrams. *Information Processing & Management*, 40(5):829–847, 2004.
3. L.M. de Campos, J.M. Fernández-Luna, and J.F. Huete. Improving the Context-based Influence Diagram Model for Structured Document Retrieval: Removing Topological Restrictions and Adding New Evaluation Method. *Lecture Notes in Computer Science*, 3408:215–229, 2005.
4. L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, and A.E. Romero Garnata: An Information Retrieval System for Structured Documents based on Probabilistic Graphical Models. Proceedings of the Eleventh International Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU), 1024 – 1031, 2006
5. F.V. Jensen. *Bayesian Networks and Decision Graphs*. Springer Verlag, 2001.
6. I. H. Witten, A. Moffat, T. C. Bell *Managing Gigabytes*, Morgan Kaufmann, 1999.
7. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan and Kaufmann, San Mateo, 1988.
8. R. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.
9. R. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, 36(5):527–550, 1988.

Information theoretic retrieval with structured queries and documents (DRAFT)

Claudio Carpineto¹, Giovanni Romano¹, and Caterina Caracciolo²

¹ Fondazione Ugo Bordoni, Rome, Italy

{carpinet, romano}@fub.it

² FAO, Rome, Italy

caterina.caracciolo@gmail.com

1 Introduction

Information retrieval through statistical language modeling has become popular thanks to its firm theoretical background and good retrieval performance. One goal of current research on structured information retrieval is thus to extend such models to take advantage of structure information.

As a structure may be present on documents or queries or both, we are interested in supporting not only unstructured queries on structured documents, but also structured queries on unstructured documents as well as structured queries on structured documents. Most of research work has considered the first task, i.e., unstructured queries over structured docs, while some papers have addressed using structured or semistructured queries on unstructured docs. Here we take a unified approach.

Our basic retrieval model is the well known Kullback-Leibler divergence, with backoff smoothing. In this paper we show how it can be extended to model and support structured/unstructured queries on structured/ unstructured documents. We make a very general assumption on the type of structure imposed on queries and/or documents, suitable for describing various structured data. We also study how the extended model can be efficiently computed.

We finally report on our experiments at INEX 2006, in which we used a rough approximation of the presented model. A full implementation of the model and a more significant evaluation of its retrieval effectiveness are left for future work.

2 Information-theoretic retrieval model

Given a query Q and a document D , the score of D for Q is given by the negative of the Kullback-Leibler (KL) divergence of the query language model θ_Q from the document language model θ_D :

$$score(Q, D) = -KL(\theta_Q|\theta_D) = - \sum_{w \in V} p(w|\theta_Q) \log \frac{p(w|\theta_Q)}{p(w|\theta_D)} \quad (1)$$

This is a well known technique for ranking documents [4]. In order to compute expression 1, we need to estimate $p(w|\theta_Q)$ and $p(w|\theta_D)$, i.e., the probability of the word w given the query language model and the document language model, respectively.

Usually this is done by using two probability distributions, one for "seen" words that occur in the text (query or document), and one for "unseen" words that do not occur in the text. This "smoothing" is due to the fact that a given text is usually too small a sample to accurately estimate the language model. One classical smoothing technique is backoff ([3]). It is based on discounting the probabilities of the seen terms, while the probability mass recuperated in this way is redistributed over the unseen terms. Usually, the probability of seen words is given by the maximum likelihood estimate applied to the text, and the probability of unseen words is estimated from the whole document collection in the same manner.

Let $c(w, T)$ be the number of times the word w occurs in text T , $c(w, C)$ be the number of times the word w occurs in the collection C , $|T|$ the number of words in T , $|C|$ the number of words in C . The probability of the word w given the text language model is given by:

$$p(w|\theta_T) = \begin{cases} \psi \frac{c(w, T)}{|T|} & \text{if } w \in T \\ \xi p(w|C) & \text{if } w \notin T \end{cases} \quad (2)$$

This smoothing technique is very popular in the speech recognition field and it has also been used for text retrieval ([1], [6]).

3 Structured information-theoretic retrieval model

If the collection of documents is structured, the basic information retrieval model is not satisfactory because it ignores the relationships between the documents. For instance, in order to retrieve elements (components) from XML documents it is natural to exploit the tree-based structuring of documents to enrich each element's description with the description of related elements ([2], [5]).

We assume that there is a partial ordering relation (\leq) over the set of documents. For each document D , let D^* be the set formed by the words that are contained in any of the documents that are implied by D according to such a relation, except for D itself; i.e., $D^* = \{w | w \in D_i, D \leq D_i, D \neq D_i\}$.

We smooth the original document model by two probability distributions. The first, estimated from D^* , gives importance to the terms that are logically related to D . The second, estimated from the document collection, gives non-zero probabilities to the terms that are neither in the document nor in its implied documents.

$$p(w|\theta_D) = \begin{cases} \alpha \frac{c(w, D)}{|D|} & \text{if } w \in D \\ \beta \frac{c(w, D^*)}{|D^*|} & \text{if } w \in D^* \\ \mu p(w|C) & \text{if } w \notin D \cup D^* \end{cases} \quad (3)$$

In order to ensure that probabilities of all terms sum to 1, the following relation must hold:

$$\alpha + \beta + \sum_{w \notin D \cup D^*} \mu p(w|C) = 1 \quad (4)$$

The same approach can be also used to estimate the query language model. A query with an explicit structure, e.g. with a title, a description, and a narrative field, is usually considered as a bag of words. However, it may be not convenient to consider all the fields as equally important because some fields may just contain verbose descriptions of other, shorter fields, and thus the longer fields should be given a smaller weight.

By analogy with structured documents, we can smooth the original query model $p(w|Q)$, as determined by the query title, by two probability distribution, one estimated from the complementary query representation given by the union of description and narrative (denoted by Q^*), one estimated from the whole collection.

$$p(w|\theta_Q) = \begin{cases} \gamma \frac{c(w,Q)}{|Q|} & \text{if } w \in Q \\ \delta \frac{c(w,Q^*)}{|Q^*|} & \text{if } w \in Q^* \\ \pi p(w|C) & \text{if } w \notin Q \cup Q^* \end{cases} \quad (5)$$

The constraint on the sum of probability is in this case given by:

$$\gamma + \delta + \sum_{w \notin Q \cup Q^*} \pi p(w|C) = 1 \quad (6)$$

Thus, in all we have six parameters (i.e., $\alpha, \beta, \mu, \gamma, \delta, \pi$) and two equations (i.e., expressions 4 and 6). In the full paper we will show how to estimate the parameters in a more compact and elegant way. We will also show that the resulting model can be computed efficiently because it does not require to compute the probabilities of all terms in the collection for each document.

4 Experiments at INEX 2006

Due to tight scheduling and limited resources, we did not have time to experiment with the full model. In our experiments we used a rough approximation of it.

We used a plain search engine to select for each topic a set of relevant documents from the Wikipedia collection. We then performed an element level analysis for each retrieved document to choose the best element(s) according to the KL divergence. The first stage amounts to performing a fast discriminative selection of candidate results using a restricted set of features (i.e., full documents instead of single elements, no information about document structure, query titles only). In the second stage, the full set of features is brought to bear to perform fine selection/reordering of the results retrieved in the first stage. More details on our experiments will be given in the full paper.

The retrieval performance of this approach was of course in the low part of INEX 2006 ranking. However, given its simplicity and its very limited computational requirements, the results are quite interesting. They can be used as a baseline for the full model.

References

1. Carpineto, C., De Mori, R., Romano, G., Bigi, B.: An information theoretic approach to automatic query expansion. *ACM Transactions on Information Systems*, 19(1):1–27, 2001.
2. Fuhr, N., GrossJohann, K.: XIRQL: A Query Language for Information Retrieval in XML Documents: In *Proceedings of SIGIR 2001*, pages 172–180, New Orleans, LA, USA., 2001.
3. Katz, S.: Estimation of probabilities from sparse data for language model component of a speech recognizer. *IEEE Trans. Acoust. Speech Signal Process.* , 35:400–401, 1987.
4. Lafferty, J., Zhai, C.: Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research, Development in Information Retrieval*, pages 111–119, New Orleans, LA, USA, 2001.
5. Ogilvie, P., Callan, J.: Language Models and Structured Document Retrieval: In *Proceedings of the INEX 2002 Workshop*, pages 33–40, Schloss Dagstuhl, Germany, 2002.
6. Zhai, C, Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004.

Dynamic Element Retrieval in the Wikipedia Collection

Carolyn J. Crouch, Donald B. Crouch, Murthy Ganapathibhotla, Vishal Bakshi

Department of Computer Science
University of Minnesota Duluth
Duluth, MN 55812
(218) 726-7607
ccrouch@d.umn.edu

Abstract

When we began our initial experiments in XML retrieval in 2002, we were interested in determining the utility of Salton's vector space model for XML retrieval. Our interests shortly became centered on dynamic element retrieval, i.e., the dynamic retrieval of elements at the desired degree of granularity. During the past two years, our work [1, 2] has focused on building a system which, for a given query, dynamically produces a rank-ordered list of elements that is virtually identical to the list produced by a search of the same query against an all-element index of the collection. Dynamic retrieval requires only a single indexing of the collection at the level of the basic indexing node. An initial retrieval is performed against this index, returning in vector form the set of all nodes with positive correlations with the query. We identify the documents represented by the n top-ranked elements in this set. Then, using schemas representing their structures, the document trees are generated dynamically from the bottom up. At each level of the tree, element vectors are generated and their term weights are computed using Lnu term weighting.

The focus of our work in 2006 has been the generation of the ltu -weighted query for use in this process. This query weighting process requires access to global information that is normally available through indexing (in this case, the all-element index) and is not available here. (In [2005] we used an approximation to this query.) A methodology for producing this information was devised during the past year. Thus the ltu -weighted query is correlated with each of the Lnu -weighted element vectors as they are generated, and a rank-ordered list of elements is produced. This process is repeated for every document in the identified set. At the end, all such lists are merged, the elements sorted, and the final, rank-ordered list is returned.

This methodology was applied to the 2004 and 2005 INEX collections, and we were able to establish that the results produced by this method were virtually identical to those produced by the equivalent search of the all-element index. (For details, see [3].) Many interesting questions arose, some of which are still being addressed. We are currently applying our dynamic element retrieval methodology to the Wikipedia collection. Wikipedia lacks the strong structure typified by the earlier INEX collections. In particular, it contains untagged elements which must be incorporated in the tree-building process. And since our system must be tuned to determine appropriate values of slope and pivot (required in the $Lnu-ltu$ term weighting process), it requires time to properly utilize the relevance assessments in this process. At this time, our experiments are underway. We believe that our current methodology, without major changes in design, can be successfully applied to this new collection.

References

- [1] Crouch, C., Mahajan, A., and Bellamkonda, A. Flexible retrieval based on the vector space model. In Fuhr, et. al. (Eds): *Advances in XML Information Retrieval, Third International Workshop for the Initiative for the Evaluation of XML Retrieval (INEX 2004)*, LNCS 3493, Springer Verlag, 2005, 292-302.
- [2] Crouch, C., Khanna, S., Potnis, P., and Doddapaneni, N. The dynamic retrieval of XML elements. In Fuhr, et. al. (Eds): *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2005)*, LNCS 3977, Springer Verlag, 2006, 268-281.

- [3] Crouch, C. Dynamic element retrieval in a structured environment. *ACM Transactions on Information Systems*, 24(4), 2006 (to appear).

The University of Kaiserslautern at INEX 2006

Philipp Dopichaj

dopichaj@informatik.uni-kl.de

University of Kaiserslautern, Gottlieb-Daimler-Str.
67663 Kaiserslautern, Germany

Abstract. Digital libraries offer convenient access to large volumes of text, but finding the information that is relevant for a given information need is hard. The workshops of the Initiative for the Evaluation of XML retrieval (INEX) provide a forum for testing the effectiveness of retrieval strategies. In this paper, we present the current version of our search engine that was used for INEX 2006: Like at INEX 2005, our search engine exploits structural patterns in the retrieval results to find the appropriate results among overlapping elements. This year, we examine how we can change this method to work better with the Wikipedia collection, which is significantly larger than the IEEE collection used in previous years. We show that our optimizations both retain the retrieval quality and reduce retrieval time significantly.

1 Introduction

The Initiative for the Evaluation of XML Retrieval (INEX)¹ provides a testbed for comparing the effectiveness of content-based XML retrieval systems. The University of Kaiserslautern participated in the INEX workshop for the second time in 2006. Our retrieval approach is mostly unchanged from the approach we used in 2005: It is based on standard vector-space retrieval on the elements, enhanced with XML-specific additions (context patterns). This year, it was our aim to answer two questions:

- Do context patterns work well without using fuzzy logic? (Last year, we introduced the concept of context patterns and implemented the using fuzzy logic.)
- Can we improve the speed of our system without compromising quality by discarding a large fraction of the intermediate results? (The Wikipedia collection used this year is roughly eight times the size of the IEEE collection from previous years.)

Context patterns address the choice of a suitable result granularity, one of the central problems of element retrieval: Due to the tree structure of XML documents, retrieval results can overlap, so the search engine needs to decide which

¹ see <http://inex.is.informatik.uni-kl.de/>

of the overlapping results are more suitable to answering the query. Context patterns are based on the observation that the structural properties of retrieval results, like length and position, provide valuable hints about the importance of the retrieved elements.

Our paper is structured as follows: We first present a brief description of our baseline retrieval system in Section 2 and then proceed to explain context patterns in Section 3. Section 4 describes changes related to the scalability of our search engine. Finally, we discuss the performance of our baseline and enhanced results as evaluated in the INEX workshop in Section 5.

2 Baseline Search Engine

The basic structure of our retrieval system has not changed since last year [2, 3]; we repeat the description here for completeness. We use the Apache Lucene information retrieval engine² as the basis and add XML retrieval functionality. Instead of storing only the complete articles from the document collection in the index, we store each element’s textual contents as a (Lucene) document, enriched with some metadata (most notably, the enclosing XML document and the XPath within that document); see Fig. 1 for an example.

	XPath	Indexed contents
<code><sec>Hello, world!</code>	<code>/sec[1]</code>	Hello, world! How are you?
<code>How <i>are</i> you?</code>	<code>/sec[1]/b[1]</code>	world!
<code></sec></code>	<code>/sec[1]/i[1]</code>	are

Fig. 1: Source document and corresponding indexed documents as seen by Lucene.

Directly searching this index using Lucene would lead to bad results – overlap is not taken into account at all, and many elements on their own are useless because they are too small –, so we need to postprocess the Lucene results. We regard the results from different input documents as independent, so we can postprocess the results from each document separately (even concurrently). Overlapping results from the same document are arranged in a tree that mirrors the structure of the original XML document; this enables us to examine the relationships between the elements. Thus, retrieval is executed in the five steps depicted in Figure 2 (the context patterns from Section 3 are applied in step 3).

2.1 Query Processing

The queries in the INEX topics are formulated in NEXI, an XML query language derived from XPath with additional information retrieval functions [7].

² see <http://lucene.apache.org>

Operation	Output
1. Process query and send it to Lucene	Raw retrieval results (fragments)
2. Rearrange retrieval results	One result tree per document
3. Postprocess the result trees	One result tree per document
4. Merge the results	Flat list of results
5. Adjust scores of short elements	Flat list of results

Fig. 2: The retrieval process.

For content-only (CO) queries, we support the full syntax of NEXI with the following modifications to the interpretation of the Boolean operators:

- We discard query terms with the “-” qualifier (instead of asserting that they do not occur in the retrieved elements).
- Query terms prefixed with “+” are assigned a higher weight (instead of asserting that they occur in the retrieved elements).
- The modifiers “and” and “or” are ignored.

For content-and-structure (CAS) queries, only the last tag name in paths is used for searching (for example, given `//article//fm//at1`, we prefer all `at1` elements, not only those contained in `//article//fm`). Furthermore, we consider the structural parts of the query only as hints for the best elements to retrieve.

3 Context Patterns

The search engine we described in the previous section provides the basis for the implementation of our new approaches. On top of it, we implemented two different enhancements that are executed as a postprocessing step.

Fortunately, there are several telltale signs what the role of a given element in a text is, without having to examine the tag name.

We can achieve this by looking at *result contexts* of the retrieved nodes. For each non-leaf node, the result context consists of this node and its children, and the following data is stored for each node:

- The retrieval score of the node,
- the length of the node’s text (in words), and
- the position of the node in the parent node.

This information can be visualized in two dimensions, one for the lengths and positions of the text fragments and the other for the score. Fig. 3 shows an example XML fragment and how it can be visualized. The horizontal position of the left-hand side of each rectangle denotes the starting position in the text of the parent element, and its width corresponds to the length of the text it contains (this implies that the parent element occupies the width of the diagram). The

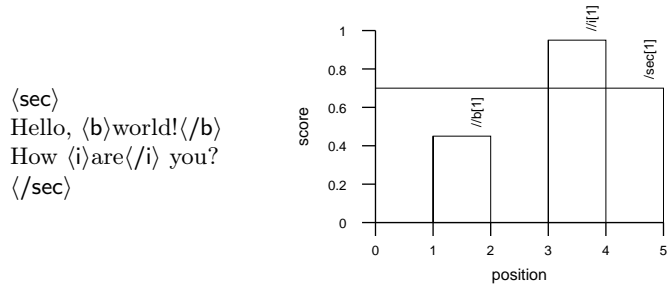


Fig. 3: XML text and corresponding context diagram. The horizontal axis denotes the positions and lengths of the text fragments, and the vertical axis shows the score (in this case random numbers).

parent element (in the Fig. 3, the root element `/sec[1]`) is the reference for the scale of the horizontal axis.

When we examined context graphs of some trial retrieval results, we realized that we could often determine what elements were section titles or inline elements, without referring to the original XML documents. Based on this observation, we defined a set of *context patterns* for formalizing the recognition of certain structures. A pattern looks like, “if the first child in the context is short and the parent is long, the first child is a title” (see Fig. 4 for an example). This is too vague for Boolean logic, but fuzzy logic is perfectly suited to this task. Fuzzy logic enables us to assign degrees of membership for the features, instead of Boolean values [6]. For example, a fragment containing only one word is definitely short, and a fragment containing 5000 words is definitely not short, but what about one containing 20 words? With fuzzy logic, we do not need to make a firm decision, but we can say that this fragment is short to a degree of (for example) 50%. Similarly, the Boolean operators like *and*, *or*, and *not* can be expressed in terms of these degrees.

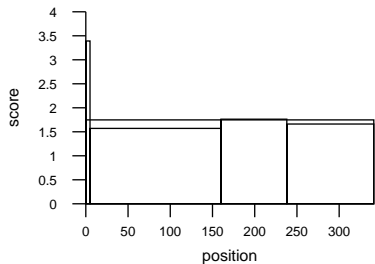


Fig. 4: Example context graph for the title pattern. The short peak at the left is the section title.

This year, in addition to evaluation the performance of patterns on the new collection, we also wanted to evaluate whether using fuzzy logic really helps and implemented a “crisp” version of the title pattern that simply regards all elements shorter than 30 words as short.

The patterns alone do not change the retrieval results in any way, so we need to take actions for modifying the relevant scores. For a match in a title, an appropriate action is to increase the parent’s score (because the match indicates that the corresponding section is highly relevant) and decrease the first child’s score (because the title itself contains too little information to be of any use).

4 Coping with the Large Collection

The switch from the INEX 2005 to the INEX 2006 data set was a challenge; the Wikipedia data collection [1] is roughly eight times as large as the IEEE collection that was used in previous years (5.8 gigabytes for Wikipedia compared to 742 megabytes for IEEE).

4.1 Keeping Metadata in Main Memory

Obviously, the search engine needs to have data about the precise location (document, XPath) of a result in order to return it to the user. In addition to that, our context patterns also require data about the word-based length and position of the fragment.

This data must be accessed as quickly as possible. Unfortunately, the size of the Wikipedia collection prohibits the straightforward storage of this information for each fragment indexed by ID in main memory, even in compressed form. Storing this data on disk is not an option, either, because there may literally be millions of intermediate results (step 2 from Figure 2), and even the best-case assumption of one seek per result would be way too slow.

Therefore, we needed to devise a memory-effective way of storing the metadata in main memory; we had to sacrifice direct access by fragment ID (now only the metadata for a whole document can be read at once), but avoiding the cost of disk access more than alleviates the additional cost of decompression.

The basic idea is to keep a compressed version of the original document structure with all text nodes replaced by the corresponding word count. For the Wikipedia collection, this results in a metadata structure size of roughly 2% of the size of the original XML files.

4.2 Discarding Intermediate Results

Even with the metadata compression scheme, obtaining the metadata for the results is still the most expensive operation in the retrieval process. The metadata includes the XPath and content length of the elements, and due to the way the data is stored, the system has to obtain the metadata for *all* elements in one go.

Many of the elements do not show up in the final results anyway (for INEX, only the 1,500 highest-ranking elements should be submitted), so it should be possible to simply discard the results from low-scoring documents after step 2 in the retrieval process (Figure 2); the metadata is needed for step 3. The problem is that “low-scoring documents” is not well-defined: There may be many elements from the same document with a variety of retrieval scores, so ranking the documents is not too straightforward (and indeed a task of its own in INEX). One could calculate the average retrieval score for all elements in a document and use that for sorting the documents. This would, however, be counterproductive for the “thorough” and “focused” tasks – a highly relevant element could be overshadowed by many elements with a much lower RSV and thus be discarded. Because of this, we use the maximum retrieval score in a document for sorting, and keep all fragments from the 5,000 best documents.

5 Evaluation

One important aspect of INEX is the comparison of XML search engines. The participating organizations submit runs consisting of up to 1,500 results for each topic, and the pooled results are then manually evaluated for relevance. This relevance information is used as input to several metrics [5] which provide a numerical value denoting the quality of a run’s retrieval results.

Although the currently available results³ are incomplete, three tentative conclusions can be drawn at this point (see Table 1 for details):

- Compared to other participants’ results, our results are on a low level.
- Discarding intermediate results does not appear to affect retrieval quality negatively; in fact, retrieval quality is in many cases slightly higher (though not significantly so). Retrieval time is reduced significantly.
- Whether or not fuzzy logic improves the performance for context-pattern-based runs remains unclear; for CO.Thorough and MAep, fuzzy logic is slightly better, for CO.Focused, it is slightly worse.

Table 1: INEX 2006 results.

Run	CO.Thorough		CO.Focused	
	MAep	Rank	nxCG@25	Rank
Fuzzy patterns	0.0184	30	0.1758	53
Crisp patterns	0.0160	41	0.1774	51
Crisp patterns (best 5,000 documents kept)	0.0161	39	0.1779	50

³ see <http://inex.is.informatik.uni-duisburg.de/2006/adhoc-protected/results.html>

References

1. Ludovic Denoyer and Patrick Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 40(1), 2006.
2. Philipp Dopichaj. The University of Kaiserslautern at INEX 2005. In Fuhr et al. [4].
3. Benedikt Eger. Entwurf und Implementierung einer XML-Volltext-Suchmaschine. Master's thesis, University of Kaiserslautern, 2005.
4. Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai, editors. *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005)*. Springer, 2006.
5. Gabriella Kazai and Mounia Lalmas. INEX 2005 evaluation metrics. In Fuhr et al. [4], pages 16–29.
6. Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*, chapter 13, pages 367–388. Springer, 2nd edition, 2004.
7. Andrew Trotman and Börkur Sigurbjörnsson. Narrowed extended XPath I (NEXI). In Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltán Szlávik, editors, *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2004)*. Springer, 2005.

Indexing "Reading Paths" for a Structured Information Retrieval at INEX 2006

Mathias Géry
Mathias.Géry@univ-st-etienne.fr

Laboratoire Hubert Curien - UMR CNRS 5516
Université Jean Monnet - Saint-Étienne

Abstract. We present in this paper our XML information retrieval experiments at INEX 2006 (ad-hoc track). We have developed a Structured Information Retrieval System based on an IR model considering Web documents as "reading paths", instead of a set of atomic and flat pages. Our algorithm is based on information propagation along the reading paths. We present some preliminary results at INEX 2006 (Ad-hoc track).

Keywords: Web IR, Indexation, Web structure, Reading Paths, Information Propagation

1 Introduction

The Web is obviously structured: it is composed of structured documents (the HTML pages can be structured), and it has also hypertext characteristics (the HTML pages can be linked together). The Wikipedia collection used by INEX is an example of such a structure, even if the structure of XML Wikipedia articles is homogeneous and not very representative of the Web structure. Document structure is an essential constituent of information description. One has to consider it during an IR process: the index should represent the semantic content of documents, including the structure. The IR model has to integrate links and their impact directly into the document model, instead of applying a simple re-ranking above a classical system.

2 Web structure and IR

Several research directions have been proposed to improve the use of Web structure in IR, especially in XML IR as studied by the INEX evaluation campaign. Some specific techniques propose to query explicitly the structure of documents (structured queries, [5]), that is not suitable for the heterogeneous Web. Two other approaches use the "global link information" (i.e. using structure independently from the query) for the indexing phase of an IRS: "information propagation" and "popularity propagation". Another approach uses the "local link

information” (i.e. using structure considering the query): “relevance propagation”. In both local and global cases, the idea is to propagate information along the links during the phase of querying and/or indexing.

Popularity propagation considers that “A good page is a page referenced by many other good pages”, typically with the calculation of a “prestige score” for each page. A popular implementation is Google *PageRank* algorithm, which calculates such a prestige score at indexing time, independently to the query [1].

Information propagation considers links by propagating *information* along them, in order to retrieve the structured documents considering their sub-parts, but also in order to retrieve the sub-parts considering the whole documents. For example, XFIRM system propagates words from sub-parts of a document to the top, and from the top to the bottom [8]. The search engine Google propagates words from context (link anchors, i.e. fragments of text, on which a user can click in order to activate a hypertext link) to a given page considering that “*anchors often provide more accurate descriptions of web pages than the pages themselves*” [1]. The anchors words are added to the index of the referenced page.

Based on the same principle than popularity propagation, relevance propagation calculates a “prestige score”, for a subset of pages that have been pre-selected considering the query. For example, the algorithm HITS calculates two “prestige scores” at query time: the *Hubs and Authorities*, assuming that “A good Hub points to many good Authorities, and a good Authority is pointed to by many good Hubs” [4]).

These links-based techniques show the interest of IR community in using structure for IR. Search engines *seem* to give good results on the Web exploiting the links, but the evaluation of these techniques is disappointing. [3] [2] show that there is no significant improvement of IR quality using the links network. Most of the systems are based on a Web model simplified to a directed graph with HTML pages as nodes and hyperlinks as edges (“triple-bag”: bag-of-words, bag-of-nodes (Web pages), bag-of-links). Few methods try to analyze what a link means regarding information, and how to consider them for IR.

We propose a Web model more structured than the uniform “triple-bag”, in order to really improve IR using links-based techniques. This model should describe the information as the authors have thought it, and index it as the readers read it.

3 Information propagation

Our IR model considers three aspects of information reading on the Web. Two of them consider the reading of a “document” (navigation inside a “document”), and are based on a tree structure and a reading path structure. The third aspect deals with the navigation outside a document, considering the concept of “context”.

INEX collection is composed by structured documents (XML), i.e. based on a hierarchical structure. Several works propose to propagate information and/or relevance from sub-parts of a document to the top, and from the top to the bottom. In this paper, information propagation is also based on a hierarchical structure, but we focus on the reading paths indexation. We propagate information along "reading paths links", i.e. from one node (section) to its brother (next section) in the document tree.

3.1 INEX Wikipedia collection

Ad-hoc collection at INEX 2006 is composed by 659.388 XML **articles** (Wikipedia entries). Each XML document is composed by tens of sections, s_j , paragraphs, etc.

$$\forall a_i \in WikipediaCollection, a_i = (s_1, s_2, \dots, s_j, \dots, s_n)$$

INEX collection test includes 125 queries, each query is composed by 5 fields : title, castitle, description, narrative, ontopics keywords. We use only the "title" field.

3.2 VSM indexation

The atomic document unit considered by our SIRS is a first level XML section (XML tag <s>). We don't use sub-sections, sub-sub-sections, etc., or any other XML element.

Our system is based on the Vector Space Model (VSM) [7] that has been well studied for atomic contents. Each article a_i and each section s_i is represented by a vector of weighted terms: $\mathbf{a}_i = (w_{i1}, w_{i2} \dots w_{ij} \dots w_{in})$; n is the number of terms in the collection.

3.3 Information propagation: reading paths for IR

We assume that a reader will read sections in their appearing order. Aiming to simulate human reading and consider this order, our algorithm is based on *reading memory* hypothesis: the reading of a s_i depends on the previous s_1, s_2, \dots, s_{i-1} that were read. For example, the information in the "introduction" section is generally used to understand the remaining of the reading path (section 2, section 3, ..., conclusion). Information that is read at the beginning of the reading path has more importance than the others, considering that it is reused afterward as reading memory. So, the reading memory benefits from an accumulation effect along the reading path.

Our choice is to propagate information from each section to the following sections. The terms appearing in a section s_i have to be considered while calculating the relevance of all sections s_j with $i < j$.

We assume that the importance of a section on its following sections decrease with the distance between sections.

```

(a)  $\forall a_i \in collection$ , BM25 produces vectors :
       $\mathbf{a}_i =, (w_{i1}, w_{i2} \dots w_{ip} \dots w_{in})$ 
(b)  $\forall a_i \in collection, \forall s_j \in a_i$ , BM25 produces vectors :
       $\mathbf{s}_j =, (w_{j1}, w_{j2} \dots w_{jp} \dots w_{jn})$ 
(c)  $\forall a_i \in collection$ , for  $j = 1$  to  $sizeof(a_i)$ 
       $\forall section_k, k \in [j + 1..sizeof(article_i)]$ 
      if  $(k - j) \leq d_{max}$  then
           $\mathbf{s}_k += \alpha \cdot \frac{\mathbf{s}_j}{(k-j)}$ 
      endif

```

Fig. 1. Information propagation

The parameters α and d_{max} (maximum distance) are fixed to 0.16 and 3 in our experiments.

4 Preliminary results

Wikipedia collection at INEX 2006 is composed by 659.388 XML articles. Our algorithm splits these articles in 1.909.598 first level sections. We use the BM25 weighting function [6], and a simple hand-made stopwords list.

We have submitted 2 runs for the Thorough sub-task of the Ad-Hoc track:

- Sections level 1, BM25 (baseline run): steps a) and b) of our algorithm.
- Section level 1, BM25 with information propagation: also step c) of our algorithm.

Our baseline run gives a score of 0.01, ranked 64th/106. Our second run (simple information propagation), is not conclusive, as the score obtained is below our baselin: 0.0091, ranked 69th/106. We have to investigate more deeply these experiments, in order to determine the impact of each one of the propagation parameters. We think that we can improve these results, tune the parameters and especially taking into account various document granularities, i.e. various XML elements instead of section level only.

5 Conclusion and future works

Our model considers the Web as a set of reading paths, instead of a set of flat, atomic and independent Web pages. Our INEX 2006 experiments implement only one aspect of our IR model: information propagation along reading path. However, we still have to investigate deeply the impact of our approach on IR.

Our objectives are to experiment with INEX test collection the other reading paths specificities. Our problematic covers a large set of questions: Is it interesting to index various documents granularities ? Take into account the nodes

ordering ? Accumulate information along reading path ? Combine reading path indexation and more classical hierarchical indexation ? Is it interesting to index a "document" (a reading path) that is composed by several nodes extracted from a XML document ? How can we extract these "documents" ? How can we identify reading links ? How can we extract these "documents" ? How can we identify reading links ?

References

1. S. Brin and L. Page. The anatomy of a large-scale Hypertextual Web Search Engine. In *7th World Wide Web Conference (WWW'98)*, Brisbane, Australia, April 1998.
2. D. Hawking and N. Craswell. Overview of the TREC-2001 Web Track. In *10th Text REtrieval Conference (TREC'01)*, pages 61–67, Gaithersburg, Maryland, USA, November 2001.
3. Jacques Savoy et Yves Rasolofo. Report on the TREC-9 Experiment: Link-based Retrieval and Distributed Collections. In *9th Text REtrieval Conference*, Gaithersburg, Maryland, United States, November 2000.
4. J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environnement. *Journal of the ACM*, 46:604–632, September 1999.
5. A. Mendelzon, G. A. Mihaila, and T. Milo. Querying the World Wide Web. *Journal of Digital Libraries*, 1:68–88, 1997.
6. S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *CIKM 2004*, pages 42–49, Washington, DC, November 2004.
7. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Janvier 1983.
8. K. Sauvagnat and M. Boughanem. Propositions pour la pondration des termes et l'evaluation de la pertinence des lments en recherche d'information structure. In *3th CONfrence en Recherche d'Informations et Applications (CORIA 2006)*, Lyon, France, March 2006.

GPX at INEX 2006
Shlomo Geva
Faculty of Information Technology
QUT
Australia
s.geva@qut.edu.au

Abstract – we describe the approaches taken at INEX 2006 in the ad-hoc retrieval track.

1. Introduction

The INEX 2006 Ad-hoc track consisted of 4 tasks, namely Thorough, Focused, All In Context and Best In Context retrieval. The reader is assumed to be familiar with the task definitions.

We have used the GPX search engine. The software was ported in 2006 from C# and MS-Access to Java and the IBM Cloudscape database. This was done to achieve speedup in searching, but the basic system remained almost unchanged. The entire set of 125 topics was processed in under 15 minutes on a 3MHz Pentium PC. We provide some brief details on the system in the next section. We also describe the approaches we took to the various tasks.

2. The GPX Search Engine

The search engine is based on XPath inverted lists. For each term in the collection we maintain an inverted list of XPath specifications. This includes the file name, the absolute XPath expression identifying a specific XML element, and the term position within the element. The actual data structure is design for efficient storage and retrieval and will be described in full details in the final post-proceedings paper.

Retrieval is performed by processing the NEXI expression and interpreting the query constraints to combine the inverted lists. In the simple case of a CO query we simply compute scores for all elements that contain at least one of the search terms. The score of an element is computed as

Equation 1: Calculation of a Leaf Element's Relevance Score

$$L = K^{n-1} \sum_{i=1}^n \frac{t_i}{f_i}$$

Here n is the count of unique query terms contained within the leaf element, K is a small integer (we used $K=5$). The term K^{n-1} scales up the score of elements having multiple distinct query terms. IT should be noted that the heuristics of rewarding multiple distinct

terms is essentially the same as taking into account query terms that do not appear in the elements. The system is not sensitive to the value of K and a value between 3 and 25 produces little difference in results. The sum is over all n terms that are found within the element 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. It should be noted that similar results are obtained if we use the *TF-IDF* to compute the sum, but it is not lead to significantly different results in our experience.

Equation 2: Calculation of a Branch Element Relevance Score

$$R = D(n) \sum_{i=1}^n L_i$$

Where:

n = the number of children elements

$D(n) = N1$ if $n = 1$

$N2$ Otherwise

L_i = the relevance score of the i^{th} 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 smaller $0 \leq N1 < N2 < 1$. A branch with only one relevant child will be ranked lower than its child. The decay factor $N2$ may be chosen large enough so that a branch with several 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 might be ranked higher than any of the paragraphs.

3. Thorough Retrieval

All the QUT runs were generated with GPX search engine, starting with Thorough retrieval. The query is parsed and interpreted as an XPath expression in a straight forward manner, albeit with NEXI loose interpretation. The CO topic runs are generated with a straight forward application of equations 1 and 2, with the elements sorted by score. With the COS queries it is possible to enforce results of a particular type (i.e. strict interpretation) or to return results by score, ignoring the element type. The search is also more specific in that it does take into account the structural cues for retrieval and the inverted lists are filtered according to the query specification. If, for instance, the query specifies that a template name should be about xyz, then the inverted list for the term xyz is filtered to eliminate all occurrences that are not in a template name context. We have experimented with several settings of the decay factor, with strict and loose interpretation of NEXI expressions, and with various query expansion techniques.

3. Focused Retrieval

Here we started from Thorough results and selected the highest scoring element on a path by filtering. We keep only elements that have a higher score than any descendent and ancestor.

4. Best in Context

We tested a trivial approach here – we simply kept the highest scoring element in each document. We made no attempt to choose more wisely and this is not necessarily a good strategy, and did not experiment with the strategy. In hindsight perhaps we should have, since as it turns out the mystery metric could be kinder to certain other strategies. Nevertheless this trivial approach seems to have produced reasonable results with the BEPD and EPRUM-BEP-Exh-BEPD metrics when using low values of the evaluation parameter A.

5. All in Context

Here the Thorough runs were filtered. The runs were sorted by file score and within each file the elements sorted by score.

6. Preliminary results

The results of retrieval were somewhat mixed. Very good results were obtained in the Thorough task, and the Best in Context task with low A values. Reasonable results were obtained in the Focused task, and slightly better when using the overlap off setting. Further analysis is required to determine what works, what doesn't, and why. It will probably be a lot easier with the mystery metrics finally revealed...

Robert Gordon University at INEX 2006: Adhoc Track

Fang Huang, Stuart Watt, David Harper, and Malcolm Clark

School of Computing, The Robert Gordon University, Scotland
{fah, sw, djh, mc}@comp.rgu.ac.uk

Abstract. This paper describes the participation of the Information Retrieval and Interaction group of Robert Gordon University in the INEX 2006 ad hoc track. We focused on two questions: “What potential evidence do human assessors use to identify relevant XML elements?” and “How can this evidence be used by computers for the same task?”. Our main strategy was to investigate evidence taken not only from the content, but also from the shallow features of how texts were displayed. We employed the vector space model and the language model combining estimates based on element full-text and the compact representation of the element. We analyzed a range of non-content priors to boost retrieval effectiveness.

1 Introduction

In this paper, we describe our participation in the INEX 2006 ad hoc track. We conducted experiments to retrieve XML elements that have similar features judged to be relevant by human assessors. The criteria used by a human assessor for judging relevance involve a complex variety of individual factors. However, it is evident that not every word of a given document catches the reader’s eye. In most cases, people judge a document’s relevance by skimming over the titles, section titles, figures, tables, and words emphasized in bold or larger fonts. We proposed, in this paper, to extract and put together all those most representative words to build a compact form of a document (or an XML element). We employed retrieval models that emphasized the importance of the compact form in identifying the relevance of a document (or an XML element). We also conducted preliminary experiments to investigate the potential features of human-retrieved elements regardless of their content, and introduced a range of priors to the language model to enhance retrieval effectiveness.

The remainder of this paper is organized as follows: Section 2 outlines our ideas for building a compact representation of a document (or an XML element). Section 3 describes the vector space model we used. The mixture language model is presented in section 4. Section 5 offers a detailed description and discussion of our INEX experiments and results. The final part, section 6, concludes with a discussion and possible directions for future work.

2 Compact Representation of an XML Element

Even a brief look at the procedure adopted by a human assessor when judging a document’s relevance to a query, shows quite clearly that not all the details of the document are taken fully into consideration. Some words attract more or less of the reader’s attention. It is our contention that words appearing in titles, figure captions, or printed in bold, italics, larger fonts, and different colors are frequently more representative of the document’s relevance. Figure 1 shows a sample text taken from a document named “Hercule Poirot” from the Wikipedia collection. Extracting words from the figure caption, and words that are underlined, or displayed in bold or larger size, we get a list of words containing “Hercule Poirot fiction character Belgium England World War I Private detective Arthur Hastings”. This list of words provides clues to the meaning and content of the original text. And we therefore believe it can be used to enhance retrieval effectiveness. Based on the this consideration, we proposed to extract

Hercule Poirot (pronounced) is a fiction character, the primary detective of Agatha Christie’s novels. He appears in over 30 books.



David Suchet as Poirot

The character was born in Belgium, and has worked as a Belgian police officer, but moved to England during World War I and started a second career as a private detective. Poirot is remarkable for his small stature and egg-shaped head, his meticulous moustache, his dandified dressing habits, his absolute obsession with

order and neatness, and his disdain for detective methods that include crawling on hands and knees and looking for clues. He prefers to examine the psychology of a crime, once even betting his best friend and sometime partner, Arthur Hastings, that he could solve a case simply by sitting in an easy chair and using his “little grey cells.”

Figure 1: a sample text

these representative words from the original text to build a compact form of a document and emphasize its importance in identifying the relevance of the document. In our experiments, retrieval units were XML elements. Consequently, the method was adapted to XML element-based (the whole document can be considered as an XML element as well), i.e., for each XML element, we built a compact representation of it by extracting words from titles, subtitles, figure captions, and words printed in bold, italics or larger fonts from the text nested by the element. The compact form was then used in our retrieval experiments based on vector space model and mixture language models.

3 Vector Space Model

We used the vector space model, the default similarity measure in Lucene[3], i.e., for a collection C , a document d and query q :

$$sim(q, d) = \sum_{t \in q} \frac{tf_{t,q} \cdot idf_t}{norm_q} \cdot \frac{tf_{t,d} \cdot idf_t}{norm_d} \cdot coord_{q,d} \cdot weight \quad (1)$$

where

$$tf_{t,x} = \sqrt{freq(t, X)} \quad (2)$$

$$idf_t = 1 + \log \frac{|C|}{freq(t, C)} \quad (3)$$

$$norm_q = \sqrt{\sum_{t \in q} tf_{t,q} \cdot idf_t^2} \quad (4)$$

$$norm_d = \sqrt{|d|} \quad (5)$$

$$coord_{q,d} = \frac{|q \cap d|}{|q|} \quad (6)$$

In our experiment, retrieval units were XML elements. An element's relevance was measured based on the element's full-text and the compact representation of the element, i.e.,

$$sim(q, e) = \frac{sim(q, e_{full}) + sim(q, e_{compact})}{2} \quad (7)$$

where e is an XML element, e_{full} is the full text nested in element e , and $e_{compact}$ is the compact form of element e .

4 Language Model

We present here a retrieval model based on the language model, i.e., an element's relevance to a query is estimated by

$$P(e|q) \propto P(e) \cdot P(q|e) \quad (8)$$

where e is an XML element; q is a query consisting of the terms t_1, \dots, t_k ; the prior, $P(e)$, defines the probability of element e being relevant in absence of a query; $P(q|e)$ is the probability of the query q , given element e .

4.1 Probability of the query

Assuming query terms to be independent, $P(q|e)$ can be calculated according to a mixture language model:

$$P(q|e) = \prod_{i=1}^k (\lambda \cdot P(t_i|C) + (1 - \lambda) \cdot P(t_i|e)) \quad (9)$$

where λ is the so-called smoothing parameter; C represents the whole collection. $P(t_i|C)$ is the estimate based on the collection used to avoid sparse data problem.

$$P(t_i|C) = \frac{\text{doc_freq}(t_i, e)}{\sum_{t' \in D} \text{doc_freq}(t', C)} \quad (10)$$

The element language model, $P(t_i|e)$, defines where our method differs from other language models. In our language model, $P(t_i|e)$ is estimated by a linear combination of two parts:

$$P(t_i|e) = \lambda_1 \cdot P(t_i|e_{full}) + (1 - \lambda - \lambda_1) \cdot P(t_i|e_{compact}) \quad (11)$$

where λ_1 is a mixture parameter; $P(t_i|e_{full})$ is a language model for the full-text of element e ; $P(t_i|e_{compact})$ is the estimate based on the compact representation of element e . Parameter λ and λ_1 play important roles in our model. Previous experiments[1,7] suggested that there was a correlation between the value of the smoothing parameter and the size of the retrieval elements. Smaller average sizes of retrieved elements require more smoothing than larger ones. In our experiments, the retrieval units, which are XML elements, are relatively small. Consequently, we set a large smoothing parameter $\lambda = 0.3$ and used equal weights for the full text and the compact representation, i.e., $\lambda_1 = 0.35$.

4.2 Element priors

The Prior $P(e)$ defines the probability that the user selects an element e without a query. Elements are not equally important even though their contents are ignored. Several previous studies[1,5] reported that a successful element retrieval approach should be biased toward retrieving large elements. Furthermore, we believe relevant elements are more likely to appear in certain parts of a document, e.g., the title, first paragraph, first section, etc.

We conducted a preliminary experiment to investigate potential non-content features that might be used to boost retrieval effectiveness. The features considered included size, type, location, and the path length of an element. Location was defined as the local order of an element ignoring its path. The path length of an element equals to the number of elements which nest it. For example, for an element `/article[1]/p[1]` (the first paragraph in the document), type of this element is 'paragraph', location is represented as '1' (the first paragraph), and the path length is 1. The main objective of our experiment was to find out the

distribution of the above features among the relevant elements. Two human assessors were asked to search the Wikipedia collection, retrieve relevant XML elements, and analyze retrieved results. Details of the procedure were: i) query creation: we created 216 queries. A query was a list of keywords or a one-sentence description of the information need which was written in natural language and without regard to retrieval system capabilities or document collection peculiarities. ii) element retrieval: in this step, each query created in the previous stage was used to explore the Wikipedia collection. The TopX[6] XML search engine, which is provided through the INEX website, was used for this task. Human assessors judged the top 100 results retrieved by TopX for each query, assessed the relevance of each of the retrieved elements, and recorded the path for each of the relevant elements. iii) feature distribution analysis: paths for relevant elements were analyzed automatically by a computer program. Results are shown in Table 1. Part (a) of the table shows that the total number of relevant elements is 9142. Among these elements, most of them are articles, sections, and paragraphs. The total number of elements of these three types is 8522, which accounts for 93.2% of the total amount. Part (b) shows the relevant elements tend to appear in the beginning parts of the text. The whole documents are excluded in this analysis, as the location feature is not applicable for the whole documents. The total number of the elements excluding whole documents is 3316. Elements with location value of ‘1’, ‘2’, ‘3’ account for 88.1%(2920 out of 3316) of the total amount. Part (c) shows relevant elements are not likely to be nested in depth. Again, whole documents are excluded. Elements that only nested by the whole article (path-length=1, e.g., /article/section, /article/p, etc.) constitute the majority (2089 out of 3316, i.e., 63.0%). Only 8.4% (280 out of 3316) of relevant elements are of path length longer than 3.

Table 1. Distribution of element shallow features

	(a)	(b)		(c)	
type	number	location-value	number	path-length	number
article	5826	1	1588	1	2089
section	2098	2	789	2	835
paragraph	598	3	543	3	112
others	620	≥ 4	396	≥ 4	280
total	9142	total	3316	total	3316

Our preliminary experiments indicated that relevant elements had some non-content features which could be used to boost retrieval effectiveness. We did not analyze the size of the elements in our experiment, because some studies[1,5] have already concluded that a successful element retrieval approach should be biased toward retrieving large elements.

Based on the above analysis, consider non-content feature set $F = \{|e|, |e_{path}|, e_{location}\}$, where $|e|$ is the size of element e measured in characters; $|e_{path}|$ is the path length of e ; and $e_{location}$ is the location of e . Assuming features are independent, we calculated the prior $P(e)$ by the following equation:

$$P(e) = \prod_{i=1}^3 P(e|F_i) \quad (12)$$

where F_i is i^{th} feature in set F . In the experiments, we chose a uniform length filter to ensure the retrieval of larger sized XML elements. The threshold used to filter out short elements was set to 120 characters, i.e.,

$$P(e| |e|) = \begin{cases} 1 & |e| \geq 120 \\ 0 & otherwise \end{cases} \quad (13)$$

The decision to measure in characters instead of words was based on the consideration that smaller segments like ‘‘I like it.’’ contains little information, while a sentence with three longer words tends to be more informative.

$P(e| |e_{path}|)$, the prior based on e_{path} in our experiments was calculated by:

$$P(e| |e_{path}|) = \frac{1}{1 + |e_{path}|} \quad (14)$$

$P(e| e_{location})$, the prior based on $e_{location}$ was calculated by:

$$P(e| e_{location}) = \frac{1}{e_{location}} \quad (15)$$

5 INEX Experiments

In this section, we present our INEX experiments in participating the Thorough task.

5.1 Index

We created inverted indexes of the collection using Lucene[3]. Indexes were word-based. All texts were lower-cased, stop-words removed using a stop-word list, but no stemming. For each XML element, all text nested inside it was indexed. In addition to this, we added an extra field which corresponded to the compact representation of the element. The indexing units could be any types of XML elements. However, due to the time restrictions placed on our experiments, we only indexed three types of elements: article, section, and paragraph.

5.2 Query processing

Our queries were created using terms only in the $\langle title \rangle$ or $\langle description \rangle$ parts of topics. Like the index, queries were word-based. The text was lower-cased and stop-words were removed, but no stemming was applied. ‘+’, ‘-’ and quotes in queries were simply removed. We process the $\langle description \rangle$ part of topics by identifying and extracting noun phrases[4] to form queries.

5.3 Runs and results

We submitted the following runs for the Thorough task.

1. SVM and nl-SVM: runs using vector space model based on full-text and compact representation of elements. For LM-1, queries were created using terms in the <title> field. And queries for nl-LM-1 were created from the <description> parts.
2. LM-2 and nl-LM-2: runs created using mixture language model based on full-text and compact representation of elements. Queries for the runs were created from <title> and <description> fields, respectively.
3. LM-1 and nl-LM-1: runs created using language model based on compact representation of elements only, i.e., equation (11) in section 4 is replaced by

$$P(t_i|e) = (1 - \lambda) \cdot P(t_i|e_{compact}) \quad (16)$$

where $\lambda = 0.3$. Queries for the runs were created from <title> and <description> fields, respectively.

Table 2 gives our overall results. Part (a) of the table shows the results of runs where the queries were created using <title> parts of the topics, and part (b) corresponds to queries created using <description> fields of topics. For runs using language models, estimates based on compact representation only (LM-1 and nl-LM-1) achieved comparable performances with estimates based on the combination of full-text and compact representation (LM-2 and nl-LM-2). This confirmed our hypothesis that the compact representation generated by extracting words from the original text is effective for element retrieval. The vector space model based on combination of full-text and compact representation outperformed our language model. However, due to the pressure of time, we did not submit baseline runs for the same retrieval model based on full-text or compact representation solely for comparison.

Results of each pair of runs using the same retrieval method (e.g., VSM and nl-VSM) show no significant difference. This prompts us to conclude that natural language queries work quite well after some shallow pre-processing.

Table 2. Results for the Thorough runs

(a)			(b)		
Run	MAep	iMAep	Run	MAep	iMAep
VSM	0.0157	0.0094	nl-VSM	0.0153	0.0092
LM-1	0.0095	0.0043	nl-LM-1	0.0091	0.0042
LM-2	0.0094	0.0043	nl-LM-2	0.0091	0.0041

6 Conclusions and Future Work

We have presented, in this paper, our experiments for the INEX 2006 evaluation campaign. We assumed important words could be identified according to the ways they were displayed in the text. We proposed to generate a compact representation of an XML element by extracting words appearing in titles, section titles, figure captions, tables, and words underlined or emphasized in bold, italics or larger fonts from the text the element nesting. Our retrieval methods emphasized the importance of these words in identifying relevance. Results of the Thorough task showed that estimates based solely on compact representation performed comparably with estimates using combinations of full-text and compact representation. This indicated that compact representation provided important clues to content of the original element, as we had assumed.

We also investigated a range of non-content priors. Our preliminary experiment indicated that relevant elements tended to be larger elements, such as whole articles, sections, paragraphs. Furthermore, relevant elements were more likely to appear in certain locations, such as the first element (e.g. first paragraph) of a document. And they were not likely to be deeply nested in the structure. We implemented priors in our language models, but the limited time at our disposal meant that we could not submit baseline runs for comparisons of how these priors work.

Our future work will focus on refining the retrieval models. Currently, the compact representation of an element is generated by words from certain parts of the text. However, the effectiveness of this method depends on the type of the documents. For example, in scientific articles, section titles (such as introduction, conclusion, etc) are not very useful for relevance judgement, whereas section titles in news reports are very informative. In the future, we will explore different patterns for generating compact representations depending on types of texts. This might involve genre identification techniques. We will investigate different priors' effectiveness and how different types of evidence can be combined to boost retrieval effectiveness.

7 Acknowledgments

We would like to thank Ulises Cervino Beresi for his help in indexing tasks.

References

1. Kamps J., Marx M., de Rijke M. and Sigurbjornsson B. XML retrieval: What to retrieve? *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2003*
2. Kamps J., de Rijke M. and M.Sigurbjornsson B. Topic field selection and smoothing for XML retrieval. *Proceedings of the 4th Dutch-Belgian Information Retrieval Workshop, 2003*
3. Lucene. The Lucene search engine, 2005. <http://jakarta.apache.org/lucene>

4. Ramshaw L. and Marcus M. Text chunking using transformation-based learning. *Proceedings of the Third ACL Workshop on Very Large Corpora, 1995*
5. Sigurbjornsson B., Kamps J. and de Rijke M. An element-based approach to XML retrieval. *INEX 2003 Workshop Proceedings, 2004*
6. Theobald M., Schenkel R. and Weikum G. An Efficient and Versatile Query Engine for TopX Search *Proceedings of the 31th International Conference on Very Large Databases (VLDB), Trondheim, Norway, 2005*
7. Zhai C. and Lafferty J. A study of smoothing methods for language models applied to ad hoc information retrieval. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001*

Tuning and evolving retrieval engine by training on previous INEX testbeds: preliminary work

Gilles Hubert

IRIT/SIG-EVI, 118 route de Narbonne, F-31062 Toulouse cedex 9
hubert@irit.fr

Abstract. This paper describes the retrieval approach proposed by the SIG/EVI group of the IRIT research centre at INEX'2006. This XML approach is based on direct contribution of the components constituting an information need. This paper focuses on the impact of corpus change between INEX'2005 and INEX'2006. This paper describes the search engine configurations and evolutions resulting from training on previous INEX testbeds and used to participate to INEX'2006. It presents also the first partial results of the experiments carried out at INEX'2006.

1 Introduction

Since the beginning of the INEX initiative, various XML retrieval systems and various evolutions of these systems were proposed. XML retrieval needs to take into account both content and structural aspects. XML retrieval systems can be separated into systems based on probabilistic models [7][10][11][14], systems based on vector space models [2][4][5][8] and systems based on databases [3][9].

A framework such as INEX is useful to try to estimate a global effectiveness of a system and to determine the contexts adapted to a system. Among the systems that participated to INEX previous years and that obtained globally good results there are approaches based on vector space models such as [4][8], probabilistic methods such as [10][14] and database systems such as [9] depending on task and quantisation.

In this paper, we present an IR method based on a vector space model. However, this approach is based on direct contribution of each component of the query and particularly on the presence of each term constituting the query. The method meets other proposals such as [4][12] in some principles but differs from heuristics, score aggregation principle or XML structure management. Different parameters are intended to provide configuration possibilities to adapt the method notably according to task and quantisation.

In the remainder of this paper, Section 2 summarizes the objectives of this year participation to the INEX'2006 round. Then, a presentation of the guiding principles on which relies the retrieval method is done in Section 3. Section 4 details the submitted runs and the partial obtained results. Section 5 concludes this paper.

2 Participation objectives

Participating to INEX this year had multiple objectives:

- On the one hand the interest was to estimate the influence of changes introduced in the INEX 2006 framework regarding corpus and tasks. The new Wikipedia corpus has features different from the past IEEE corpus notably regarding corpus size, document contents and document structures. Furthermore, new tasks have been defined i.e. Best in Context and All in Context. In addition, runs using queries on content only and runs using queries on content and structure were merged for evaluation,
- On the other hand, the interest was to study the behaviour of different configurations of our method resulting from learning on previous INEX testbeds. These configurations intend to be suited to the different tasks and quantisations defined in the INEX 2006 round.

3 Method principles

The IR method described in this paper is based on a vector space model. Document and query representations are comparable to vectors. However, the correspondence between documents and query is not estimated using a “usual” similarity measure. The method is based on a generic scoring function that can be adapted to different retrieval contexts. The current definition of the scoring function results from work on automatic document categorization [1] and work on XML retrieval at previous INEX rounds [5][6][13]. The scoring function is based on direct contribution of each query term appearing in an XML element. The contribution can be modulated according to other components of the query such as structural constraints. A principle of score aggregation completes the method with regard to the hierarchical structure of XML documents.

The scoring function is defined as a combination of three values. It can be globally defined as follows:

$$Score(Q, E) = \left(\sum_i f(t_i, E) \cdot g(t_i, Q) \right) \cdot h(Q, E)$$

where

Q is the query

t_i is a term representing the query Q

E is an XML element

$f(t_i, E)$	This factor estimates the importance of the term t_i in the XML element E .
$g(t_i, E)$	This factor estimates the importance of the term t_i in the query representation Q .
$h(Q, E)$	This factor estimates the global presence of the query Q in the XML element E .

On the one hand, the function is defined as an addition of contributions of the concepts constituting a query. This principle allows giving relevance to elements dealing about either only one concept or several concepts. The addition tends to promote elements containing several concepts. However, depending on the different chosen functions an element dealing strongly about one concept can be estimated higher than an element dealing lightly about many concepts. On the other hand, the function estimates globally the relevance of an element according to a query.

The function f that estimates the importance of a term in an XML element is based on the number of occurrences of the term in the element moderated by the number of XML elements of the corpus containing the term. Using this latter factor, the function increases the contributions of terms appearing in few XML elements of the corpus. This principle is similar to the $tf.idf$ principle. A coefficient related to structural constraints on content term intends to increase or reduce term contributions according to constraint matching.

The function g that estimates the importance of a term in the query representation is based on the frequency of the term in the topic. The frequency is moderated by the total number of occurrences of terms in the query. A coefficient related to term prefixes intends to increase or reduce term contributions according to sign '+' and '-' associated to terms in the query.

The function h that estimates the global presence of a query in an XML element is based on the proportion of terms common to the query and the element with respect to the number of distinct query terms. A function power is used to clearly distinguish the elements containing a lot of terms describing the query from the elements containing few terms of the query.

So, the scoring function is defined as follows:

$$Score_t(Q, E) = \left(\sum_i \frac{cc(t_i, E) \cdot Occ(t_i, E)}{NbE(t_i)} \cdot \frac{prf(t_i, Q) \cdot Occ(t_i, Q)}{Occ(Q)} \right) \cdot \varphi^{\left(\frac{NbT(Q, E)}{NbT(Q)} \right)}$$

where

t_i is a term representing the query Q

E is an XML element

$cc(t_i, E)$	Coefficient defined for the matching of constraint on content (associated to the term t_i) by the element E.
$Occ(t_i, E)$	Number of occurrences of the term t_i in the element E.
$NbE(t_i)$	Number of elements containing the term t_i
$prf(t_i, Q)$	Coefficient defined for the prefix associated to the term t_i in the query Q.
$Occ(t_i, Q)$	Number of occurrences of the term t_i in the query Q.
$Occ(Q)$	Total of occurrences of all the terms representing Q.
φ	Query presence coefficient, positive real
$NbT(Q, E)$	Number of terms of the query Q and that appear in the XML element E.
$NbT(Q)$	Number of distinct terms of the query Q.

The coefficients $cc(t_i, E)$ and $prf(t_i, Q)$ can be defined by functions. At the moment, these coefficients are defined as follows:

```

cc(t_i, E)  if E does not match the structural constraint defined on t_i
             then cc(t_i, E)=0.5
             else cc(t_i, E)=1.0

prf(t_i, Q) if prefix is '+'
             then prf(t_i, Q)=5.0
             else if prefix is '-'
                  then prf(t_i, Q)=-5.0
                  else prf(t_i, Q)=1.0

```

This solution allows attaching variable importance to structural constraints on content and prefixes. These definitions are resulting from experiments carried out on INEX'2003 and INEX'2004 testbeds.

The hierarchical structure of XML is taken into account through score aggregation. The hypothesis on which is based our method is that an element containing a component selected as relevant is also relevant and more if it has several relevant components. So, in our approach the score of an element is defined as the sum of its score computed according to its textual content (if it exists) and the scores of its descendant components that have a textual content (if they exist). The score of a component can be modulated (for example, according to distance between the component and the ascendant) when aggregating in the ascendant depending on the applied strategy. At the moment, the aggregation is defined as follows:

$$Score_a(Q, E) = Score_t(Q, E) + \sum_l \alpha^{\frac{d(E, E_l)}{d(E_r, E_l)}} \cdot Score_t(Q, E_l)$$

where

α (real) is the score aggregation coefficient

E, E_l and E_r are XML elements

E_l is a descendant component of the E structural hierarchy (document) such as E_l has textual content

E_r is the root element of the structural hierarchy (document) of which E is a descendant component

$d(X, Y)$ is the distance between an element X and its descendant element Y (for example in the path `/article/bdy/sec/p[2]`, $d(\text{bdy}, \text{p}[2]) = 2$).

The coefficient α allows varying the influence of scores of descendant components in the aggregated score of an XML element. Leaf components have no descendant thus for such components: $Score_a(T, E) = Score_t(T, E)$.

Two types of structural constraints can be used to define INEX topics:

- constraints on content (e.g. `about(//p,'+XML + "information retrieval")`),
- constraints on the granularity of target elements (e.g `//article[...]`).

As seen above, structural constraints on content are taken into account adding a coefficient $cc(t_i, E)$ in the scoring function $Score_t$.

Structural constraints on the granularity of target elements are handled adding a coefficient that modifies the aggregated score $Score_a$ (equal to $Score_t$ for leaf nodes). The general principle is that if the XML element does not verify the constraint on target granularity associated to the query, the score computed is reduced. The aggregated score including granularity coefficient is therefore defined as follows:

$$Score_a(Q, E) = gc(Q, E) \cdot \left(Score_t(Q, E) + \sum_l \alpha^{\frac{d(E, E_l)}{d(E_r, E_l)}} \cdot Score_t(Q, E_l) \right)$$

where

$gc(Q, E)$ Coefficient defined for the matching of constraint on target (associated to the query Q) by the element E .

At the moment, this coefficient is defined as follows:

if E does not match the structural constraint defined on Q
then $gc(Q, E) = 0.5$
else $gc(Q, E) = 1.0$

This definitions results from experiments carried out on INEX'2003 and INEX'2004 testbeds.

This solution allows attaching variable importance to structural constraints on result granularity. When $\gamma=0.0$ the structural constraints on result are strictly taken into account. When $\gamma=1.0$ the structural constraints on result are not taken into account.

The scoring function is completed by the notion of coverage. Coverage is a threshold corresponding to the percentage minimum of query terms that have to appear in an element to select it. It aims at ensure that only documents in which the query is represented enough will be selected for this topic. Coverage is combined to the scoring function as follows:

$$\text{If } \frac{NbT(Q, E)}{NbT(Q)} < CT \quad \text{Then } Score_a(Q, E) = 0.0$$

where

CT Coverage threshold, real positive, $0.0 \leq CT \leq 1.0$

$NbT(Q, E)$ Number of terms common to the query Q and the element E

$NbT(Q)$ Number of distinct terms describing the query Q

Coverage is currently combined to $Score_a$. Otherwise, it can be applied to $Score_t$, and then it has consequences on aggregated scores.

An additional process can be done to eliminate overlapping elements in the result. This process consists in filtering the result and keeping according to a defined strategy only one element when two overlapping elements are encountered. A strategy is for example to keep the element with the highest score.

4 Experiments

At least two runs based on our XML retrieval method was submitted to INEX 2006 for each subtask one using the title part of queries and the other using the castitle part. Depending on the subtask, an additional run using either title or castitle was submitted.

4.1 Experiment setup

Resulting from experiments and learning when participating to INEX'2003 and INEX'2004, all submitted runs shared the same values for the prefix coefficient $prf(t, Q)$ and the coverage threshold CT (cf section 3). The coverage threshold was fixed to 0.35 (i.e. more than a third of terms describing the topic must appear in the

text to keep the XML component). The values of prefix coefficient applied were fixed to +5.0 for the prefix '+', -5.0 for the prefix '-' and 1.0 for not prefixed terms.

For all the subtasks, CAS runs (i.e. having a label containing 'CAS') used the castitle part of each topic definition to define queries instead of the title part used for other runs. The coefficients taking into account structural constraints were fixed to 0.5 (i.e. the contribution of a query term is divided by 2 when the element does not meet the structural constraint) for all the subtasks. Structural constraints were handled so as vague conditions.

Furthermore, depending on the subtask we studied three combinations of score aggregation coefficient α and query presence coefficient ϕ (cf section 3) resulting from learning and experiments on INEX'2005 testbeds. The following combinations were tested:

- runs with labels containing 'CH01x1' used $\alpha=0.1$ and $\phi=1$. This combination obtained good results on INEX'2005 testbeds for the subtask Thorough and the quantisation strict. This combination includes weak score aggregation and does not consider global query presence.
- runs with labels containing 'CH06x50' used $\alpha=0.6$ and $\phi=50$. This combination obtained good results on INEX'2005 testbeds for the subtask Thorough and the quantisation generalised. This combination includes rather important score aggregation and considers moderately global query presence.
- runs with labels containing 'CH0x3000' used $\alpha=0.0$ and $\phi=3000$. This combination obtained good results on INEX'2005 testbeds for the subtask Focused. This combination does not include score aggregation and considers strongly global query presence.

4.2 Partial 'official' results

The partial 'official' results currently available and corresponding to different configurations of our method tested for the subtasks Thorough and Focused are detailed in the following tables.

Table 1. Partial results for the subtask Thorough

Task	Thorough	
Metric: ep/gr	<i>Quantisation: generalised, overlap=off</i>	
Run	V2006Cg35CH01x1tho	V2006CASCH01x1tho
Maep	0.0147	0.0161
Rank	47/106	40/106

The first observation is that results are slightly over average. Since the configuration results from experiments on strict quantisation with INEX'2005 testbeds, results for strict quantisation are awaited to establish final conclusions. Another observation is

that structural conditions seem to improve the results since the run using castitle parts of queries obtain higher average precision.

Table 2. Partial results for the subtask Focused

Task	Focused					
Metric: nxCG	<i>Quantisation: generalised</i>					
Run	V2006CH0x3000foc		V2006CASCH0x3000foc		V2006CH06x50foc	
<i>overlap=on</i>	precision	rank	precision	rank	precision	rank
nxCG@5	0.2899	43/85	0.2848	53/85	0.2630	61/85
nxCG@10	0.2472	42/85	0.2435	46/85	0.2198	62/85
nxCG@25	0.1905	43/85	0.1843	48/85	0.1759	52/85
nxCG@50	0.1558	36/85	0.1472	42/85	0.1471	43/85
<i>overlap=off</i>	precision	rank	precision	rank	precision	rank
nxCG@5	0.3151	41/85	0.3118	43/85	0.2666	63/85
nxCG@10	0.2841	35/85	0.2742	40/85	0.2270	62/85
nxCG@25	0.2255	34/85	0.2167	39/85	0.1826	54/85
nxCG@50	0.1801	28/85	0.1742	31/85	0.1466	47/85

The first observation is that results are average. Another observation is that treating only elements with textual content without including score aggregation leads to better results. To return leaf nodes seems to be better for Focused task than to return intermediate nodes. Another observation is that structural conditions do not improve results which can be explain by the fact that only a restricted set of XML elements are treated when score aggregation is not used.

5 CONCLUSIONS

Different changes have been introduced between the previous INEX'2005 round and the current INEX'2006 round. The changes occurred at different levels:

- The Wikipedia corpus has replaced the IEEE corpus introducing differences on corpus size, document contents and document structures,
- New tasks have been defined notably the 'Best in Context' task that asks systems to return one best entry per relevant article,
- There is no separate CAS task. Runs using topic 'titles' and runs using topic 'castitles' have been merged for evaluation. Furthermore, it was possible to make runs using other topic parts that title part or castitle part.

Participating to INEX this year had multiple objectives such as evaluating the impact of framework changes on our method effectiveness and to study the behaviour of

different configurations of our method resulting from learning on previous INEX testbeds.

The first partial ‘official’ results lead to mixed conclusions. Further results will give more information to establish final conclusions and future works.

References

- [1] Augé J., Englmeier K., Hubert G., Mothe J., Classification automatique de textes basée sur des hiérarchies de concepts. *Veille Stratégique Scientifique & Technologique (VSST'2001)*, Barcelone, 2001, p. 291-300.
- [2] Crouch C. J., Khanna S., Potnis P., Doddapaneni N., The Dynamic Retrieval of XML Elements An Approach to Structured Retrieval Based on the Extended Vector Model, *Advances in XML Information Retrieval, LNCS 3977, 4th International Workshop INEX, 2006*, p. 268-281.
- [3] Fuhr N., Großjohann K., XIRQL: An XML query language based on information retrieval concepts. *ACM Transactions on Information Systems (TOIS)*, vol. 22, Issue 2, 2004, p. 313-356.
- [4] Geva S., GPX – Gardens Point XML IR at INEX 2005, *Advances in XML Information Retrieval, Advances in XML Information Retrieval, LNCS 3977, 4th International Workshop INEX, 2006*, p. 240-253.
- [5] Hubert G., XML Retrieval Based on Direct Contribution of Query Components, *Advances in XML Information Retrieval, LNCS 3977, 4th International Workshop INEX, 2006*, p. 172-186.
- [6] Hubert G., A voting method for XML retrieval. *Advances in XML Information Retrieval, 3rd International Workshop INEX, LNCS 3493, 2005*, p. 183-196.
- [7] Larson R. R., Probabilistic Retrieval, Component Fusion and Blind Feedback for XML Retrieval, *LNCS 3977, 4th International Workshop INEX, 2006*, p. 225-239.
- [8] Mass Y., Mandelbrod M., Using the INEX Environment as a Test Bed for Various User Models for XML Retrieval, *Advances in XML Information Retrieval, LNCS 3977, 4th International Workshop INEX, 2006*, p. 187-195.
- [9] Mihajlović V., Ramírez G., Westerveld T., Hiemstra D., Blok H. E., de Vries A. P., TIJAH Scratches INEX 2005: Vague Element Selection, Image Search, Overlap, and Relevance Feedback, *Advances in XML Information Retrieval, LNCS 3977, 4th International Workshop INEX, 2006*, p. 72-87.
- [10] Ogilvie P., Callan J., Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval, *Advances in XML Information Retrieval, LNCS 3977, 4th International Workshop INEX, 2006*, p. 211-224.
- [11] Sigurbjörnsson B., Kamps J., de Rijke M., The Effect of Structured Queries and Selective Indexing on XML Retrieval, *Advances in XML Information Retrieval, LNCS 3977, 4th International Workshop INEX, 2006*, p. 104-118.
- [12] Sauvagnat K., Hlaoua L., Boughanem M., XFIRM at INEX 2005: Ad-Hoc and Relevance Feedback Tracks, *Advances in XML Information Retrieval, LNCS 3977, 4th International Workshop INEX, 2006*, p. 88-103.
- [13] Sauvagnat K., Hubert G., Boughanem M., Mothe J., IRIT at INEX 2003. 2nd Initiative for the Evaluation of XML Retrieval, *Dagstuhl, 2003*, p. 142-148.
- [14] Vittaut J.-N., Piwowarski B., Gallinari P., An Algebra for Structured Queries in Bayesian Networks, *Lecture Advances in XML Information Retrieval, LNCS 3493, 3rd International Workshop INEX, 2004*, p. 100-112.

The University of Amsterdam at INEX 2006

Jaap Kamps^{1,2}, Marijn Koolen¹, and Börkur Sigurbjörnsson²

¹ Archives and Information Science, Faculty of Humanities, University of Amsterdam

² ISLA, Faculty of Science, University of Amsterdam

Abstract. We describe the University of Amsterdam’s participation in the INEX 2006 Adhoc Track. We participated in all four Adhoc Track tasks, and report initial findings based on a single set of measure for all four tasks. Our main findings are the following. First, a complete element index outperforms a restricted index based on section-structure, albeit the differences are small. Second, grouping elements per article does not lead to performance degradation, but may improve scores. Third, all restrictions of the “pure” element runs (by removing overlap, by grouping elements per article, or by selecting a single element per article) lead to some but only moderate loss of precision.

1 Introduction

In this paper we document the University of Amsterdam’s participation in the INEX 2006 Adhoc Track. Our main aims for INEX 2006 were to investigate the effectiveness of our XML retrieval approaches on a new collection, the Wikipedia XML corpus [1], which has a different nature than the IEEE collection used in INEX 2002–2005. What are the characteristics of the new Wikipedia collection, and how do they affect the performance on our element retrieval system? We want to know which approaches transfer well to a new sort of collection, and which approaches don’t and why.

The rest of the paper is organized as follows. First, Section 2 describes the Wikipedia collection. Next, Section 3 documents the XML retrieval system used in the experiment. Then, in Section 4, we detail the setup of the experiments. The results of the experiments are reported in Section 5. Finally, in Section 6, we discuss our findings and draw some initial conclusions.

2 Wikipedia collection

In previous years, the IEEE collection was used in INEX. This year sees the introduction of a new collection, based on the English Wikipedia collection [2]. The collection has been converted from the wiki-syntax to an XML format [1]. Whereas the IEEE collection has somewhat over 12,000 documents, the Wikipedia collection has more than 650,000 documents. To get some idea of the characteristics of this new collection, we have gathered some statistics. Table 1 shows a few basic collection statistics. There are over 50,000,000 elements using 1241

different tag names. However, of these, 779 tags occur only once, and only 120 of them occur more than 10 times in the entire collection. On average, documents have almost 80 elements, with an average depth of 4.82.

Table 1. Wikipedia collection statistics

Description	Statistics
# of articles	659,388
# of elements	52,555,826
# of unique tags	1241
Avg. # of elements per article	79.69
Average depth	4.82

Next, we gathered tag statistics like collection frequency, document frequency and element length. Table 2 shows the 10 longest elements and their collection frequency. The length is the average number of words in the element. The `<article>` element is the longest element of course, since it always encompasses all other elements. However, after the `<body>` element, the other long elements occur only rarely in the entire collection, and contain only a few hundred words. Clearly, most of the elements are rather short. Even the average article length is short, containing no more than 415 words.

Table 2. Longest elements in Wikipedia collection

Element	Mean length	Collection freq.
<code><article></code>	414.79	659,388
<code><body></code>	411.20	659,388
<code><noinclude></code>	380.83	14
<code><h5></code>	253.18	72
<code><td_align></code>	249.20	4
<code><h4></code>	237.13	307
<code></code>	198.20	163
<code><timeline></code>	186.49	48
<code><number></code>	168.72	27
<code><h3></code>	163.80	231

In table 3 the most frequent tag names are listed. Column 2 shows the average document frequency of the tag name, column 3 shows the collection frequency. There are many links to other Wiki pages (`<collectionlink>`s), and many `<unknownlink>`s that are not really links (yet). Wiki pages have more than 4 paragraphs (indicated by `<p>` tags) and more than 2 sections on average.

As shown in table 4, the elements `<article>`, `<conversionwarning>`, `<body>` and `<name>` occur in every single document. Almost all documents have links to

Table 3. Most frequent tags in Wikipedia collection

Tag name	Document freq.	Collection freq.
<collectionlink>	25.80	17,014,573
<item>	8.61	5,682,358
<unknownlink>	5.98	3,947,513
<cell>	5.71	3,770,196
<p>	4.17	2,752,171
<emph2>	4.12	2,721,840
<template>	3.68	2,427,099
<section>	2.44	1,609,725
<title>	2.41	1,592,215
<emph3>	2.24	1,480,877

other Wiki pages (99.4%), and more than 70% have text tagged as <unknownlink> (indicating a topic that could have its own page). Together with the average frequency of the <collectionlink>s, this indicates a very dense link structure. Apart from that, the textual unit indicating elements <section> and <p> (paragraph) occur in 69.6% and 82.1% of the documents respectively.

Table 4. Elements with the highest document frequency in Wikipedia collection

Tag name	Document freq.	%
<article>	659,388	100.0
<conversionwarning>	659,388	100.0
<body>	659,388	100.0
<name>	659,388	100.0
<collectionlink>	655,561	99.4
<emph3>	587,999	89.2
<p>	541,389	82.1
<unknownlink>	479,830	72.8
<title>	459,253	69.6
<section>	459,252	69.6

The main observation is that elements are small on average. One important reason for this is the Wikipedia policy of splitting long articles into multiple new pages.³ The idea is that encyclopedia entries should be focused. If the article grows too long, it should be split into articles discussing the sub-topics. This is

³ As http://en.wikipedia.org/wiki/Wikipedia:Summary_style reads: “The length of a given Wikipedia entry tends to grow as people add information to it. This cannot go on forever: very long entries would cause problems. So we must move information out of entries periodically. This information should not be removed from Wikipedia: that would defeat the purpose of the contributions. So we must create new entries to hold the excised information.” (November 2006).

a policy that closely resembles the main purpose of element retrieval: a relevant results must be specific. The dense structure of the collection links should make it easy to navigate to other relevant pages.

3 XML Retrieval System

3.1 Indexing

Our indexing approach is based on our earlier work [3,4,5]).

- *Element index*: Our main index contains all retrievable elements, where we index all textual content of the element including the textual content of their descendants. This results in the “traditional” overlapping element index in the same way as we have done in the previous years [3].
- *Section index*: We built an index based on frequently retrieved elements. Studying the distribution of retrieved elements, we found that the <article>, <body>, <section> and <p> elements are retrieved far more often than other elements. The only exceptions are the <collectionlink> elements. However, since collection links contain only a few terms at most, and say more about the relevance of another page, we didn’t add them to the index.
- *Article index*: We also build an index containing all full-text articles (i.e., all wikipages) as is standard in IR.

For all indexes, stop-words were removed, but no morphological normalization such as stemming was applied. Queries are processed similar to the documents, we use either the CO query or the CAS query, and remove query operators (if present) from the CO query and the about-functions in the CAS query.

3.2 Retrieval

For all our runs we used a multinomial language model [6]. We use the same mixture model implementation as we used in earlier years [4]. We assume query terms to be independent, and rank elements according to:

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

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

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

where $P_{mle}(\cdot|e)$ is a language model for element e ; $P_{mle}(\cdot|d)$ is a language model for document d ; and $P_{mle}(\cdot)$ is a language model of the collection. The parameters

λ_e and λ_d are interpolation factors (smoothing parameters). Finally, we assign a prior probability to an element e relative to its length in the following manner:

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

where $|e|$ is the size of an element e . For a more thorough description of our retrieval approach we refer to [4].

4 Experimental Setup

In this section, we detail the experiments and runs submitted to the INEX 2006 Adhoc track tasks. None of our official submissions used the three layered mixture model, i.e., we use $\lambda_d = 0$ throughout.

4.1 Thorough

For the Thorough Task, we submitted two runs using the CO query (from the topic’s `<title>` field) and two runs using the CAS query (from the topic’s `<castitle>` field). We regard the Thorough Task as underlying all other tasks, and all other runs are based on postprocessing them in various ways.

The two Thorough CO runs are:

`thorough_element_lm` Language model ($\lambda_e = 0.15$) on the element index.

`thorough_section_lm` Language model ($\lambda_e = 0.15$) on the section index.

Our two CAS query runs are also based on postprocessing the CO run based on the element index. We extract all path-restrictions on the element of request in the CAS query, and filter the results for elements conforming on all or some of the location steps.

The two Thorough CAS query runs are:

`thorough_element_lm.cas.joined` Language model ($\lambda_e = 0.15$) on the element index, retaining elements that satisfy the complete path expression.

`thorough_element_lm.cas.seperate` Language model ($\lambda_e = 0.15$) on the element index, retaining element that satisfy at least the tagname of the element of request.

4.2 Focused

For the Focused Task we submitted two runs using the CO query and two runs using the CAS query. All our Focused Task submissions correspond to a Thorough Task submission, and are postprocessed by a straightforward list-based removal strategy. We traverse the list top-down, and simply remove any element that is an ancestor or descendant of an element seen earlier in the list. For example, if the first result from an article is the article itself, we will not include any further element from this article.

The resulting two Focused CO runs are:

`focused_element_lm` Language model ($\lambda_e = 0.15$) on the element index, with list-based removal of ancestor or descendant elements.

`focused_section_lm` Language model ($\lambda_e = 0.15$) on the section index, with list-based removal of ancestor or descendant elements.

The resulting two Focused CAS runs are:

`focused_element_lm.cas.joined` Language model ($\lambda_e = 0.15$) on the element index, retaining elements that satisfy the complete path expression, and with list-based removal of ancestor or descendant elements.

`focused_element_lm.cas.seperate` Language model ($\lambda_e = 0.15$) on the element index, retaining element that satisfy at least the tagname of the element of request, and with list-based removal of ancestor or descendant elements.

4.3 All in Context

For the All in Context Task, we only submitted runs using CO query. Here, we base our runs on the Thorough Task runs using the section index. We cluster all elements belonging to the same article together, and order the article clusters either by the highest scoring element, or by the combined scores of all elements belonging to the article.

The two All in Context CO runs are:

`all_section_lm.highest` Language model ($\lambda_e = 0.15$) on the section index, clustered by article and ranked according to the highest scoring element in an article, and with list-based removal of ancestor or descendant elements.

`all_section_lm.sum` Language model ($\lambda_e = 0.15$) on the section index, clustered by article and ranked according to the sum of element scores in an article, with list-based removal of ancestor or descendant elements.

4.4 Best in Context

Finally, for the Best in Context Task we submitted three runs, all based on the CO query. We use, again, the runs made against the section index, and postprocess them such that only a single result per article is kept in the result file.

`best_section_lm.highest_score` Language model ($\lambda_e = 0.15$) on the section index, selecting only the highest scoring element per article.

`best_section_lm.article` Language model ($\lambda_e = 0.15$) on the section index, selecting the article node of each element from an unseen article.

`best_section_lm.first` Language model ($\lambda_e = 0.15$) on the section index, selecting only the first element (in reading order) that is retrieved per article.

5 Results

At the time of writing, only partial results are available. We opt to compare all runs on equal grounds, focusing on two common measures that are available in the EvalJ package: a mean-average-precision measure (MAep), and early precision measure (nxCg at rank 5, 10, 25, and 50). This allows us to measure the effectiveness of various post-processing methods, such as overlap-removal or clustering by article, in terms of their relative impact on precision and recall. Before discussing the results for each of the Adhoc tasks, we first give some detail about the topics and resulting relevance judgments.

5.1 Topics and Judgments

Assessments are available for 111 topics (numbered 289–298, 300–306, 308–369, 371–376, 378–388, 390–392, 395, 399–407, 409–411, and 413). There is a total 8,737 relevant passages for these 111 topics in 5,483 different articles. Table 5 shows some statistics of the relevant passages (i.e., the text highlighted as relevant by the assessors). It is interesting to see that most relevant passages are

Table 5. Relevant passage statistics

Description	Statistics
# articles with relevance	5,483
# relevant passages	8,737
avg. rel. pass. length	1,098
median rel. pass. length	56

very short. The lengths of the elements are measured in characters (text offset). The length distribution is skewed: the difference between mean and median relevant passage length is enormous. Apparently, most relevant passages contain only a few words or a sentence, indicating that even though the average article is rather short, there is still a lot of irrelevant text that can be filtered out.

Table 6 looks at the judgments from the vista point of elements containing only relevant text. We see that there are many `<collectionlink>` elements that contain relevant text. This is not very surprising, because the `<collectionlink>` element is by far the most frequent element in the entire collection (see Table 3). Other short elements, like `<cell>`, `<emph2>` and `<unknownlink>` are also found often in relevant passages. The lengths mentioned are the average lengths of the *relevant* elements of that type. Longer elements containing relevant text are mostly `<section>` and `<p>` elements.

The shorter elements often contain only a few words, and often are only a small part of the entire passage. However, there are a lot of `<collectionlink>` elements that encompass an *entire* relevant passage. Table 7 shows the frequency

Table 6. Relevant element statistics

Tag name	Frequency	Avg. length
<collectionlink>	171,766	14
<item>	35,107	285
<cell>	29,711	17
<p>	29,199	470
<emph2>	28,260	22
<unknownlink>	24,893	17
<section>	20,667	2,434
<emph3>	11,867	16
<row>	10,148	57
<title>	9,082	25

Table 7. Elements encompassing entire relevant passages

Tag name	Frequency	Avg. length
<p>	2,813	509
<collectionlink>	1,592	16
<name>	886	21
<title>	715	21
<emph3>	699	19
<item>	532	140
<emph2>	216	60
<body>	209	5,227
<unknownlink>	202	18
<caption>	191	72

of elements that are the shortest element to encompass an entire relevant passage. Apparently, topic authors often consider a link to another page to be a relevant passage. However, the <p> element now surfaces as the most frequent shortest element to encompass an entire relevant passage. This gives support to our Section index as a viable indexing strategy. The focus of this year’s relevance metrics is in specificity, though, so these results might point us in the wrong direction.

5.2 Thorough

We’ll now discuss the results for the four Adhoc tasks, starting with the Thorough Task. The Thorough Task puts no restriction on XML elements to return. Table 8 shows the results for the Thorough Task. We first discuss the top two

Table 8. Results for the Thorough Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
<code>thorough_element_lm</code>	0.4120	0.3789	0.3262	0.2790	0.0343
<code>thorough_section_lm</code>	0.3948	0.3721	0.2977	0.2503	0.0227
<code>thorough_element_lm_cas_joined</code>	0.1872	0.1642	0.1410	0.1100	0.0116
<code>thorough_element_lm_cas_seperate</code>	0.2124	0.1761	0.1511	0.1208	0.0129

runs using the keyword or CO query. We make a few observations: First, we see that the index containing all XML elements in the collection is more effective on all measures. Second, we see that difference with the section index (containing only article, body, section, and paragraph nodes) is relatively small, especially in terms of precision. This is in line with earlier results on the IEEE collection, and shows the potential of the much smaller section index. We now zoom in on the bottom two runs using the structured or CAS query. We see that the joined run (using the element of request’s full path as a Boolean filter) performs less than the separate run (filtering only for the tagname of the element of request). When comparing the results for the CO and CAS queries, we see that the CO query runs are more effective for both mean average precision, and for early precision. While the loss of mean average precision can be expected, the structural hints hold the potential to improve precision. We should note, however, that the CAS processing was done naively, resulting in many topics with very few or no results left.

5.3 Focused

For the Focused Task, none of the retrieved elements was allowed to contain text that overlaps with another retrieved element. We evaluate runs here using the same measures as the Thorough Task above, but since the elements judged relevant in recall base may overlap, performance can never obtain perfect scores.

Table 9. Results for the Focused Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
<code>focused_element_lm</code>	0.3571	0.3245	0.2560	0.2116	0.0105
<code>focused_section_lm</code>	0.3386	0.2868	0.2212	0.1825	0.0080
<code>focused_element_lm.cas.joined</code>	0.1522	0.1310	0.0975	0.0740	0.0033
<code>focused_element_lm.cas.seperate</code>	0.1781	0.1418	0.1060	0.0789	0.0037

Table 9 shows the results for the Focused Task. Since we use the same post-processing method—the list-based, top-down removal of elements overlapping with earlier seen text—we see the same relative behavior as for the Thorough Task runs above. The complete element index is still more effective than the smaller section index. This signals that the effectiveness of the element index is not due to the fact that it contains all potentially overlapping elements (which could be exploited in theory). When comparing the Focused Task results to the Thorough Task results, we note that, as expected, the scores are lower. There is a moderate decline for the precision scores, but the recall (and mean average precision) drops dramatically.

5.4 All in Context

For the All in Context Task, there is the further restriction that retrieved elements must be grouped per article (and still may not overlap). Again, we evaluate runs here using the same measures as the Thorough Task above, so optimal performance will result in still imperfect scores. Table 10 shows the results for the

Table 10. Results for the All in Context Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
<code>all_section_lm.highest</code>	0.3357	0.3082	0.2290	0.1889	0.0082
<code>all_section_lm.sum</code>	0.2454	0.2135	0.1804	0.1470	0.0059

All in Context Task. We see that for ranking the groups of elements from the same article, the best scoring element is a more useful criterion than the sum of all element scores. When comparing the All in Context Task results to the Focused Task results, we note that the precision scores are in the same league. In fact, considering that our All in Context runs are all based on the section index, the clustering by article improves performance for all measures except for the nxCG at rank 5.

5.5 Best in Context

For the Best in Context Task, we may only retrieve a single result per article. For this task, a best-entry-point was obtained from the human judge during the assessment procedure. At the time of writing, the measure corresponding to this best-entry-point judgment is unavailable, and hence we evaluate the retrieved element in terms of perceived topical relevance. For ease of comparison, we evaluate runs here using the same measures as the Thorough Task above, so optimal performance will result in still grossly imperfect scores. Table 11 shows

Table 11. Results for the Best in Context Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
<code>best_section_lm.highest_score</code>	0.3290	0.2796	0.2082	0.1670	0.0069
<code>best_section_lm.article</code>	0.2451	0.1983	0.1423	0.1106	0.0045
<code>best_section_lm.first</code>	0.3290	0.2796	0.2082	0.1670	0.0069

the results for the Best in Context Task. It is comforting to note that the element selection strategies outperform the strategy that simply backs off to the whole article. Our Best in Context runs where based on postprocessing the All in Context run, result in the same performance for selecting the first or highest scoring element. This may be due to the layered processing of the runs, where the Thorough Task’s section index run is processed by removing overlap, then clustered by article, and then, finally, we selecting our final element to retrieve. When comparing the Best in Context Task results to the All in Context Task results, we note that there is only a moderate loss of precision for the Best in Context Task.

The main aim of the Best in Context task is to investigate how the chosen best entry point related to the perceived relevance in the article. There is obvious value in comparing the results above to the results based on the selected best entry point.

6 Discussion and Conclusions

This paper documents the University of Amsterdam’s participation in the INEX 2006 Adhoc Track. We participated in all four Adhoc Track tasks. At the time of writing, only partial result are available. Hence we decide on a single set of measures for all four Tasks, focusing on both precision and mean average precision. This will allow for straightforward comparison between the tasks, but may not reflect best each individual Task. In particular, we evaluate the Best in Context Task not in terms of the best entry point as provided in the assessments, but in terms of the perceived relevance.

Our main findings so far are the following. First, for the Thorough Task, we see that a complete element index was more effective than a restricted index

based on the sectioning structure, although the difference is not large. We also see that the keyword or CO query was more effective than the structured or CAS query. Second, for the Focused Task, we observe a very similar pattern as for the Thorough Task. This is a reassuring result, because it signals that the superior performance of the element index is not due to the fact that it contains many overlapping elements. Third, for the All in Context Task, we find that the clustering per article is in fact improving the performance when compared to the corresponding overlap-free Focused Task runs. Fourth, for the Best in Context Task, we see that element selection outperforms backing off to the whole article, and obtain—perhaps suprisingly—still agreeable precision scores in terms of perceived relevance.

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (NWO, grants # 612.066.302, 612.066.513, 639.072.601, and 640.001.501), and by the E.U.'s 6th FP for RTD (project MultiMATCH contract IST-033104).

References

1. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. *SIGIR Forum* **40** (2006) 64–69
2. Wikipedia: The free encyclopedia (2006) <http://en.wikipedia.org/>.
3. Sigurbjörnsson, B., Kamps, J., de Rijke, M.: An Element-Based Approach to XML Retrieval. In: *INEX 2003 Workshop Proceedings*. (2004) 19–26
4. Sigurbjörnsson, B., Kamps, J., de Rijke, M.: Mixture models, overlap, and structural hints in XML element retrieval. In: *Advances in XML Information Retrieval: INEX 2004*. Volume 3493 of LNCS 3493. (2005) 196–210
5. Sigurbjörnsson, B., Kamps, J.: The effect of structured queries and selective indexing on XML retrieval. In: *Advances in XML Information Retrieval and Evaluation: INEX 2005*. Volume 3977 of LNCS. (2006) 104–118
6. Hiemstra, D.: *Using Language Models for Information Retrieval*. PhD thesis, University of Twente (2001)

Using Language Models and the HITS Algorithm for XML Retrieval*

Benny Kimelfeld, Eitan Kovacs, Yehoshua Sagiv, and Dan Yahav

The Selim and Rachel Benin School of Engineering and Computer Science
The Hebrew University of Jerusalem
Edmond J. Safra Campus, Jerusalem 91904, Israel
{bennyk,koveitan,sagiv,dyahav}@cs.huji.ac.il

Our goal for INEX 2006 was to assess a variety of methods for XML and Web retrieval. The nature of the Wikipedia collection, which contains XML elements and links (XPointer and XLink), led us to believe that combining methods of XML and Web retrieval could be a promising approach. Thus, we have developed an information-retrieval system that is extensible, in the sense that it is composed of several modules that were designed so that each one can be easily reimplemented in order to test different retrieval methods. We first conducted thorough experiments on the INEX 2005 collection, using the automatic assessment tool, thereby gaining knowledge on which method performed best on a specific ad-hoc sub-task (e.g., CO Thorough, CO Focused).

We have experimented with three different methods for ranking XML elements. All our methods are based on a linear interpolation of three language models [1]: corpus, document and element. We also use element-length cut-off and keyword proximity. In addition, we apply the HITS algorithm [2] (using the JUNG library [3]) in combination with the language-modeling approach.

Our general approach starts by ranking documents, according to some criteria, and then applying each of the three methods in order to rank elements of the documents that are deemed relevant. One criterion prefers documents that have more keywords from the query. Another one chooses documents that are highly ranked by the HITS algorithm.

We apply the structural constraints by considering the support keywords with a reduced weight. In our results for INEX 2006, there is only a minor difference between the CO and the COS tasks. Overall, the combination of the HITS algorithm and the language-model approach gives the best results.

References

1. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR, ACM (1998) 275–281
2. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. In: SODA, ACM (1998) 668–677
3. The JUNG Framework Development Team: JUNG java universal network/graph framework. (2006) <http://jung.sourceforge.net>.

* This research was supported by The Israel Science Foundation (Grant 893/05).

CSIRO's participation in INEX 2006

Alexander Krumpholz and David Hawking

CSIRO ICT Centre, Canberra, Australia
{Alexander.Krumpholz, David.Hawking}@csiro.au

Extended Abstract

In this year's participation of INEX, CSIRO participated in the Ad-hoc Track, contributing to three of the four given tasks, namely the Thorough Task, the Focussed Task and the Best in Context Task.

In order to use CSIRO's plain text search engine PADRE we had to preprocess the collection in a similar manner to the approach taken by our group in 2002. We split the documents in subdocuments according to the elements that we need to retrieve and indexed the files with PADRE.

In a first step we extracted query elements from the INEX topics. Then the query processor generated PADRE queries and post-processed the results according to specifications for each run.

The following runs have been submitted:

CSIRO-CAS1-A Using the Content-And-Structure-Title as a query to PADRE, using the results as they are retrieved.

CSIRO-CAS2-A Using the Content-And-Structure-Title as a query to PADRE, restricting results to match the query.

CSIRO-CO1-A Using the Content-Only-Title as a query to PADRE, using the results as they are retrieved.

CSIRO-CO1-B Using the Content-Only-Title as a query to PADRE, but suppressing overlapping results.

CSIRO-CO1-D Using the Content-Only-Title as a query to PADRE, restricting results to articles.

CSIRO-CO2-A Using the Content-And-Structure-Title without structural references as a query to PADRE, using the results as they are retrieved.

CSIRO-CO2-B Using the Content-And-Structure-Title without structural references as a query to PADRE, but suppressing overlapping results.

CSIRO-CO2-D Using the Content-And-Structure-Title without structural references as a query to PADRE, restricting results to articles.

EXTIRP: baseline retrieval from Wikipedia

Miro Lehtonen¹ and Antoine Doucet²

¹ Department of Computer Science
P. O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

`firstname.lastname@cs.helsinki.fi`

² IRISA-INRIA
Campus de Beaulieu
F-35042 Rennes Cedex
France

`firstname.lastname@irisa.fr`

Abstract. The Wikipedia XML documents are considered an interesting challenge to any XML retrieval system that is capable of indexing and retrieving XML without prior knowledge of the structure. Although the structure of the Wikipedia XML documents is highly irregular and thus unpredictable, EXTIRP manages to handle all the well-formed XML documents without problems. Whether the high flexibility of EXTIRP also implies high performance concerning the quality of IR has so far been a question without definite answers. The initial results do not confirm any positive answers, but instead, they tempt us to define some requirements for the XML documents that EXTIRP is expected to index. The most interesting question stemming from our results is about the line between high-quality XML markup aids accurate IR and noisy “XML spam” that misleads flexible XML search engines.

1 Introduction

The experimental XML retrieval system of University of Helsinki — EXTIRP — needed only slight modification when adapted to indexing and retrieving information from the Wikipedia document collection. Application of the existing methods to a new set of documents was especially interesting: EXTIRP has previously been tested on the IEEE article collection only, although it can handle documents of arbitrary document types. The previous test results could be explained by alleged fine-tuning to a single document collection because we were not able not show how EXTIRP worked on other collections. Therefore, the Wikipedia documents added another valuable dimension to the testing history of EXTIRP.

Partly because of our low resources and partly because of our desire to keep our system from 2005 pristine, we did not analyse the Wiki documents before they were indexed and queried for the official submissions. There certainly was no fine-tuning one way or another. We also have left out many of the characteristic

features that have been part of EXTIRP during its short history. These features include query expansion, intra-document reference analysis, as well as weighting schemes for titles and inline elements. The remaining system has come close to a baseline retrieval model based on the vector space model and cosine similarity.

2 Background

The history and basic structure of EXTIRP.

3 Selective indexing

Examples of XML that was left out, and examples that were included.

Demonstration of how our methods work and how the Wikipedia XML documents challenge them.

Looking at our observations raises an interesting question: What is the validity of this evaluation where the test documents can only be used in the evaluation because the structure is completely useless elsewhere?

4 Related work

5 Results

The results from INEX 2005 showed that the official evaluation metrics [1] do not favour systems like EXTIRP because there is no reward for returning “too small” answers. The 2005 version could not adjust the granularity of the answers according to the query, but the granularity came directly from the indexed document fragments [2]. The 2006 version of EXTIRP comes with the same drawback even though some “near misses” are rewarded.

6 Conclusion

As a simple implementation of an XML retrieval system, the 2006 version of EXTIRP serves as a baseline that other more advanced implementations can be compared with. However, according to the official evaluation metric (XCG), the performance of this baseline is so poor that other metrics with better results are necessary for a meaningful comparison.

References

1. Kazai, G., Lalmas, M.: INEX 2005 Evaluation Measures. [3] 16–29
2. Lehtonen, M.: When a few highly relevant answers are enough. [3] 296–305
3. Fuhr, N., Lalmas, M., Malik, S., Kazai, G., eds.: Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, November 2005, Revised Selected Papers. In Fuhr, N., Lalmas, M., Malik, S., Kazai, G., eds.: INEX. Volume 3977 of Lecture Notes in Computer Science., Springer (2006)

CISR at INEX 2006

Wei Lu¹, Stephen Robertson^{2,3}, Andrew Macfarlane³

¹ Center for Studies of Information Resources, School of Information Management
Wuhan University, China and City University
sa713@soi.city.ac.uk

² Microsoft Research, Cambridge, U.K
ser@microsoft.com

³ Centre for Interactive Systems Research, Department of Information Science
City University London
andym@soi.city.ac.uk

Abstract. In this paper, we describe the Centre for Interactive Systems Research's participation in the INEX 2006 adhoc track. Rather than using field-weighted BM25 model in 2005, we revert back to using the traditional BM25 weighting function. Our main research aims in this year is to investigate the effects of document filtering by result record cut-off, element filtering by length cut-off and the effects of using phrases. The initial results show the latter two methods did not do well, while the first one did well on FOCUSED TASK and RELEVANT IN CONTEXT TASK. We also derived a novel method for BEST IN CONTEXT TASK which requires more investigation.

1. Introduction

This is the second year that the CISR has participated in INEX. In INEX 2005, we used a field-weighted BM25 model and submitted runs for two adhoc CO tasks [1]. Our results show the method is promising. Subsequent to this, we investigated XML retrievable units and element inheritance in [2] and the average element length in [3]. This year, rather than exploiting the field-weighted method further, our work mainly focuses on investigating the effects of document filtering, element filtering and using phrases.

In traditional text retrieval system, a document is usually treated as independent unit. But for XML element retrieval, elements in the same document are usually semantically relevant and can be independent units themselves, e.g., article title, abstract, and section title to section text in IEEE's data collection. This raises an issue of how context elements affect the effectiveness of XML element retrieval. Some work has been done in this area. Lu et al [1] and Robertson et al [2] used a field-weighted method to exploit the inheritance from context elements; Abolhassani et al [4], Geva et al [5] and Ogilive et al [6] used various methods to compute the parent element's weight by merging its sub-element's weight. Both of these two methods consider the

element weight inheritance from context elements but without evidence from the whole document. Sigurbjornsson et al [7] and Mass et al [8] investigated document weight's contribution to element retrieval by using a interpolation method of merging the document weight into element weight. The results show this method is beneficial and has yielded good results at INEX 2004 and INEX 2005.

In this paper, we use a different method to exploiting the document affects to element retrieval. That is, we divided element retrieval into two phases: firstly, we conduct document level retrieval and set a cut-off for the retrieved results; secondly, we use the filtered results to further execute element level XML retrieval. Our aim is to investigate whether using top weighted documents can produce better results than the method we have utilized previously.

In order to avoid the too-small element problem, we use two methods. We restrict our set of retrievable units to article, body, section and paragraph and we set a cut-off for element length to abandon those elements which are shorter than the cut-off value. We also use phrases instead of single words to see if it could improve retrieval effectiveness.

In section 2, we simply introduce the BM25 model used in our experiment. In section 3, we discuss our submitted runs. Section 3 further illustrates the experiment of this method on INEX 06 and Evaluation results are reported in section 4. A conclusion and further work to be undertaken are given at the end.

2. BM25 model

Different to that in 2005, we use BM25 model only. For ad-hoc retrieval, and ignoring any repetition of terms in the query, BM25 can be simplified to:

$$wf_j(d, C) = \frac{(k_1 + 1)tf_j}{k_1((1 - b) + b\frac{dl}{avdl}) + tf_j} \log \frac{N - df_j + 0.5}{df_j + 0.5} \quad (1)$$

where C denotes the document collection, tf_j is the term frequency of the j th term in document d , df_j is the document frequency of term j , dl is the document length, $avdl$ is the average document length across the collection, and k_1 and b are tuning parameters.

Formula (1) uses a logarithmic function to compute term's collection weight. For frequently occurring terms, this function will produce negative weight values. To avoid this, we used an alternative weight function (command "w fn=3" in Okapi bss system) instead of the logarithmic function.

3. Description of the Experiments

Within the ad-hoc XML retrieval task there are four sub-tasks: THOROUGH TASK, FOCUSED TASK, RELEVANT IN CONTEXT TASK and BEST IN CONTEXT TASK. For each sub-task, we submitted 3 or 4 runs only for CO queries but not for CAS queries. The details of these experiments are as follows:

3.1 THOROUGH TASK

We submitted 3 runs for THOROUGH TASK. They are THOR-BM25-nobody, THOR-BM25-nobody-cutoff400 and THOR-BM25-400-1500-phrase:

- THOR-BM25-nobody directly uses BM25 to compute the element weight score;
- THOR-BM25-nobody-cutoff400 is much the same as THOR-BM25-nobody except an element length cut-off 400 is set, which filters out elements shorter than 400 characters;
- THOR-BM25-400-1500-phrase uses the same element length cut-off, and it also set document result cut-off (1500) and uses phrases instead of single words.

3.2 FOCUSED TASK

The 3 submitted runs for FOCUSED TASK are as follows:

- FOCU-BM25-cutoff400 uses 400 characters length as element length cut-off;
- FOCU-BM25-cutoff400-filter1500 uses the same element length cut-off and also uses document result cut-off (1500);
- FOCU-BM25-cutoff400-filter1500-phrase is similar to the above one except using phrases rather than single terms.

3.3 RELEVANT IN CONTEXT TASK

For this task, We submitted runs All-BM25-cutoff400, All-BM25-cutoff400-filter1500 and All-BM25-cutoff400-filter1500-phrase. These runs use the same conditions as the ones for FOCUSED TASK. The difference is that the results in the runs are grouped by articles and without overlap elements.

3.4 BEST IN CONTEXT TASK

BEST IN CONTEXT TASK is a new ad-hoc task which aims at locating the best entry point of XML retrieval. We used two methods for this task and submitted 4 runs. In the first method, we just take the element with the highest weight score (best-match

element) in each document as the best entry point. The 2 submitted runs BEST-BM25-cutoff400 and BEST-BM25-filter1500 used this method.

In the second method, we proposed novel way of selecting the best entry point. The distribution of element weight scores in the document is considered. Our basic idea is that, given an element, if more than one of its sub-elements has a good score, then this element should be chosen as the candidate best entry point rather than using its sub-element as the candidate best entry point. A problem for this particular method is how to determine a good score. In implementation, we set half the score of the best-match element in each document as the cut-off for determining a good score and use a bottom up method for selecting the best entry point. For each document, we firstly find the best-match element in the document. Then we consider all other elements which do not overlap with this one. If any of these elements scores higher than half the score of the best-match element, then it should be included in the scope implied by the entry point. That is, we move the entry point up to the start of a higher-level element, such that the higher-level element includes all the high-scoring elements.

For example, in Fig. 1, the best-match element is E and the best-match element weight score is 2.11, so the cut-off value is 1.055. Using this method, we get the best entry point B.

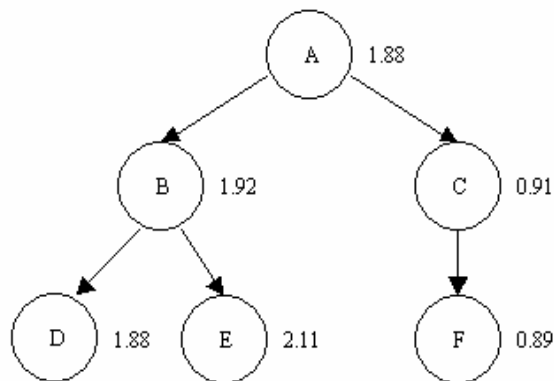


Fig.1 XML element tree with element weight score

In INEX 2006, the 2 submitted runs BEST-BM25-400-1500-level-p and BEST-BM25-Level-filter1500 used this method.

4. Evaluation

The evaluation results of our runs are shown in Table 1, Table 2, Table 3 and Table 4. From the table 1, we can see that the run using the basic BM25 model does best, and in table 2 and 3, the runs using element length cut-off do best (They also rank at the top of all INEX 2006's corresponding submitted runs), while the runs using document filtering and phrases does not do well. Table 4 shows the results of our runs for

locating document’s best entry point. Although the run BEST-BM25-cutoff400 does best among the 4 runs, we can see that the 2 runs BEST-BM25-400-1500-level-p and BEST-BM25-Level-filter1500 using the derived novel method do better than the run BEST-BM25-filter1500. These 3 runs use the same document result set for locating the best entry point. This may suggest us for more experiment on this method.

Table 1: Evaluation results for THOROUGH Task

Runs	Metric: ep/gr
THOR-BM25-nobody	0.0228
THOR-BM25-nobody-cutoff400	0.0215
THOR-BM25-400-1500-phrase	0.0118

Table 2: Evaluation results for FOCUSED Task

Runs	Metric: nxCG			
	5	10	25	50
FOCU-BM25-cutoff400	0.3961	0.3428	0.2638	0.2001
FOCU-BM25-cutoff400-filter1500	0.3054	0.2557	0.1873	0.1335
FOCU-BM25-cutoff400-filter1500-phrase	0.2849	0.2452	0.1836	0.1332

Table 3: Evaluation results for RELEVANT IN CONTEXT Task

Runs	Metric: nxCG			
	5	10	25	50
All-BM25-cutoff400	0.4176	0.3629	0.2783	0.2126
All-BM25-cutoff400-filter1500	0.3216	0.2710	0.2018	0.1448
All-BM25-cutoff400-filter1500-phrase	0.3048	0.2650	0.1994	0.1398

Table 4: Evaluation results for BEST IN CONTEXT Task

Runs	Metric: BEPD				
	A=0.01	A=0.1	A=1	A=10	A=100
	Metric: EPRUM-BEP-Exh-BEPDistance				
	A=0.01	A=0.1	A=1	A=10	A=100
BEST-BM25-cutoff400	0.0860	0.1311	0.1984	0.3175	0.4532
	0.0221	0.0435	0.0760	0.1431	0.2349
BEST-BM25-filter1500	0.0655	0.1071	0.1706	0.2621	0.3441
	0.0139	0.0311	0.0547	0.0956	0.1384
BEST-BM25-400-1500-level-p	0.0664	0.1071	0.1710	0.2626	0.3429
	0.0147	0.0319	0.0567	0.0989	0.1420
BEST-BM25-Level-filter1500	0.0610	0.1087	0.1749	0.2632	0.3413
	0.0153	0.0367	0.0629	0.1014	0.1395

5 Conclusion

Rather than using field-weighted BM25 model in 2005, we reverted back to using the basic BM25 model. We exploited the effects of element filtering by length cut-off, document filtering by result record cut-off and the effects of using phrases. The results show the latter two methods did not do well, while the first one did very well on FOCUSED TASK and RELEVANT IN CONTEXT TASK and for THOROUGH TASK the effectiveness is not quite obvious. We also utilized a novel method for BEST IN CONTEXT TASK. However we did not consider the number of sub-elements and the adjacency of the relevant elements. These issues need to be investigated further.

Acknowledgements

This work is supported in part by National Social Science Foundation of China 06CTQ006.

References

- [1] W. Lu, S. Robertson, A. Macfarlane. Field-Weighted XML Retrieval Based on BM25. Proceedings of INEX 2005. LNCS. 2006 126-137
- [2] S. Robertson, W. Lu, A. Macfarlane. XML-structured documents: retrievable units and inheritance. FQAS 2006. Springer LNCS. 2006 121-132
- [3] W. Lu, S. Robertson, A. Macfarlane. Investigating Average Element Length for XML Retrieval by Using BM25. (submitted to review)
- [4] M. Abolhassani, N. Fuhr, S. Malik. HyREX at INEX 2003. Proceedings of the Second Workshop of the Initiative for The Evaluation of XML Retrieval (INEX), 15-17 December 2003, Schloss Dagstuhl, Germany.
- [5] S. Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. INEX 2004, Lecture Notes in Computer Science, Springer-Verlag GmbH Volume 3493 / 2005.
- [6] P. Ogilvie, J. Callan. Hierarchical Language Models for XML Component Retrieval. INEX 2004, Lecture Notes in Computer Science, Springer-Verlag GmbH Volume 3493 / 2005.
- [7] 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.
- [8] Y. Mass, M. Mandelbrod, Component Ranking and Automatic Query Refinement for XML Retrieval, INEX 2004, Lecture Notes in Computer Science, Springer-Verlag GmbH Volume 3493 / 2005.

A scalable XML component ranking algorithm

Yosi Mass

IBM Haifa Research Lab
Haifa 31905, Israel
yosimass@il.ibm.com

Abstract. In previous INEX years we presented an XML component ranking algorithm that was based on separation of elements to different indices. This worked fine for the IEEE collection which has a small number of potential component types that can be returned such as sections and paragraphs. However, such an assumption doesn't scale to this year Wikipedia collection where there is a large set of potential component types that can be returned. We show a new version of the Component ranking algorithm that does not assume any knowledge on the set of component types. We then show how we exploited the connectivity of the Wikipedia collection to use Page Rank and Anchor Text in the component ranking algorithm.

1 Introduction

The challenge in XML retrieval is to return the most relevant components that satisfy the user needs. Of most interest is the class of CO (Content Only) queries where the user doesn't know anything about the collection structure and issue her query in free text. The search engine then exploits the XML structure to return the most relevant XML components that satisfy the user needs.

The main challenge in component ranking is how to adapt ranking methods from classical IR [3] that rank full documents to rank components inside a document. The main problem is that classical IR methods work on statistics such as term frequency and document frequency at the document level. This does not perform well at the component level due to component nesting in XML as explained in [4,5,6]

In previous INEX workshops we described a component ranking algorithm [4,5,6] that solved the component nesting problem by running each query against different indices where each index contains elements of the same type. The idea is to build different indices for the most informative component types where each index contains elements of the same type. This worked fine for the IEEE collection where the components we choose were {article, abs, bdy, sec, ss1, ss2 and p+ ip1}. This leaves us with 7 indices which is a manageable solution. However in this year Wikipedia collection we can not pre identify such small number of the most informative collection as there are at least hundreds of such potential component types.

In this paper we describe a modified version of the component ranking algorithm that does not assume any set of known component types, but still uses the same idea of a small number of separate indices with no nested elements. We further show how we exploited the rich connectivity of the Wikipedia collection to improve component ranking by applying PageRank and anchor text algorithms.

The rest of the paper is organized as follows: In section 2 we describe the modified component ranking algorithm and in section 3 we describe the addition of PageRank and Anchor Text algorithms. In section 4 we describe our runs and results in the adhoc track. We conclude in section 5 with summary and some conclusions.

2 Component ranking (indexing) algorithm

The basic idea in the Component ranking algorithm[4,5,6] is to build different indices for the most informative component types where each index contains elements of the same type. The indices we used for the IEEE collection in previous years were {article, abs, bdy, sec, ss1, ss2 and p+ ip1}.

The component ranking algorithm is described in Fig 1 below. We give here a short summary while full details can be found in [4,5,6]. Given a query Q , we run the query in parallel on each index (step 1) and then optionally apply an Automatic Query Refinement (AQR) algorithm such as LARefinement[2] or similar (step 2) on each result set. Then in step 3, the scores of elements in each result set are normalized to $\text{score}(Q,Q)$ which as described in details in [6], normalizes scores from the different indices to the same range so that they can be compared. In step 4 we apply a document pivot scaling where scores of elements from each index are scaled by the score of their parent article. Finally all the results sets are merged into a single result set of all element types.

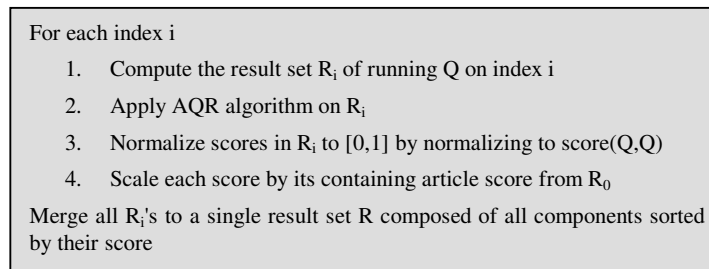


Fig. 1. Component ranking algorithm

The above algorithm requires some prior knowledge on the collection such as the decision on which element types to index in the separate indices. This doesn't work

for heterogeneous collection that may have different element types. Moreover even for the Wikipedia collection it doesn't work since a simple analysis of the collection shows that there can be hundreds of possible element types that can be returned. Table 1 below shows distribution of element types by their size. The table shows the top 20 element types sorted by their max direct size in tokens. A direct size of an element is the content directly under it and a total size of an element is the sum of all elements in the tree below the element. All those top 20 elements and much more are potential to be returned so the method of selecting a small number (e.g. 7) of the most informative elements does not scale for such collections.

	A	B	C	D	E	F	G	H
1	Element Name	count	avgDirect Size	avgTotal Size	minDirect Size	maxDirect Size	minTotal Size	maxTotal Size
2	div	13117	47	280	0	103990	0	110055
3	body	659361	190	2555	1	70312	1	297611
4	section	1609969	48	840	0	41360	0	141502
5	cadre	149342	22	88	0	36759	0	38652
6	template	2427489	6	18	0	17645	0	47914
7	gallery	2527	277	390	0	12801	0	39063
8	p	2752784	304	353	0	10737	0	50884
9	emph2	2722601	15	21	0	10367	0	10367
10	normallist	1110093	3	263	1	8504	2	122629
11	blockquote	4830	327	496	0	8210	0	18306
12	td	370975	8	104	0	8180	0	80652
13	item	5683252	28	50	0	8033	0	46820
14	indentation1	135404	66	113	0	5550	0	26942
15	i	17935	21	44	0	4815	0	18143
16	small	61132	16	41	0	3943	0	29194
17	title	1592497	14	15	0	3716	0	61533
18	b	11298	12	37	0	2974	0	14712
19	indentation2	14065	54	87	0	2576	0	17513
20	cell	3770300	7	13	0	2491	0	30990

Table 1. Distribution by element type size

We still want to use the component ranking algorithm so we use a different approach for creating the separate indices. Going back to the roots of the component ranking algorithm, the motivation for creating the separate indices was two fold: first to solve the problem of nested elements and second to compare elements of same nature. We describe below a “Component indexing algorithm” for creating a small number of indices for any given collection and we show that it satisfies the above two objectives. The indexing algorithm gets two parameters -

1. **minCompSize** – the minimum component size (in tokens) to index from each document
2. **numIndices** - the number of indices to create.

The indexing algorithm parses (using XML SAX parser) each document in the collection and finds all minimal elements that are larger than *minCompSize*. A minimal element is either a leaf element whose size is larger than *minCompSize* or the deepest

element in a path whose direct size is larger than *minCompSize* and it has no descendant with a direct size larger than *minCompSize*.

We mark the indices with $\text{Index}_0 \dots \text{Index}_{n-1}$ where $n = \text{numIndices}$. For each such minimal element we extract all its ancestors and if its depth is less than the number of indices then each level are indexed in a separate index. For example assume $\text{numIndices} = 7$ then for example the minimal element `/article[1]/bdy[1]/p[1]` will be indexed as follows:

Index 0: `/article[1]`
Index 1: `/article[1]/bdy[1]`
Index 2: `/article[1]/bdy[1]/p[1]`

If the depth of a minimal element is greater than the number of indices then we take the first elements in the path into the first indices and the last elements in the path into the last indices skipping some elements in between. For example

`/article[1]/body[1]/section[7]/table[1]/tr[1]/td[2]/tr[1]/td[2]/tr[1]/td[2]`

which has depth 10 will be split into the 7 indices as follows:

index 0 : `/article[1]`
index 1 : `/article[1]/body[1]`
index 2 : `/article[1]/body[1]/section[7]`
index 3 : `/article[1]/body[1]/section[7]/table[1]/tr[1]/td[2]/tr[1]`
index 4 : `/article[1]/body[1]/section[7]/table[1]/tr[1]/td[2]/tr[1]/td[2]`
index 5 : `/article[1]/body[1]/section[7]/table[1]/tr[1]/td[2]/tr[1]/td[2]/tr[1]`
index 6 : `/article[1]/body[1]/section[7]/table[1]/tr[1]/td[2]/tr[1]/td[2]/tr[1]/td[2]`

If a document does not have any minimal element namely it has not element whose direct size greater than **numIndices** then we index the all document (e.g. `/article[1]`) into index_0 .

So far we showed how to split a single path into the separate indices. However a document may have several minimal elements that may have common ancestors so while building higher level indices we make sure that elements do not repeat. For example the following minimal set of elements –

`/article[1]/body[1]/p[1],`
`/article[1]/body[1]/section[1]/p[1],`
`/article[1]/body[1]/section[1]/p[2],`
`/article[1]/body[1]/section[2]/section[1]/p[1],`
`/article[1]/body[1]/section[2]/section[1]/p[2],`
`/article[1]/body[1]/section[2]/section[4]/p[2]`
`/article[1]/body[1]/p[5],`

are split to indices as follows:

Index 0: /article[1]]

Index 1: /article[1]/body[1]]

Index 2: /article[1]/body[1]/p[1],
/article[1]/body[1]/section[1],
/article[1]/body[1]/section[2],
/article[1]/body[1]/p[5],

Index 3: /article[1]/body[1]/section[1]/p[1],
/article[1]/body[1]/section[1]/p[2],
/article[1]/body[1]/section[2]/section[1],
/article[1]/body[1]/section[2]/section[4]]

Index 4: /article[1]/body[1]/section[2]/section[1]/p[1],
/article[1]/body[1]/section[2]/section[1]/p[2],
/article[1]/body[1]/section[2]/section[4]/p[2]

It's easy to verify that this separation to indices satisfies the first requirements that elements in an index are not nested. The second requirement of having in each index elements of same nature is not very intuitive. It does exist in higher level indices for example in $Index_0$ we always index the whole document. Its true that in the last index due to the cutoff of long depths, some elements can be from dept $n-1$ but some may be of level $m > n-1$ so we need more investigation if a different separation to indices is more appropriate.

3 PageRank and AnchorText

The Wikipedia collection is highly connected through internal <collectionlink> links. The number of incoming such links into a document is called the document's PageRank. A document with a higher PageRank is deemed to be more relevant for a topic than a similar document with a lower PageRank. Each such link has further a *href* attribute which we call AnchorText. The AnchorText is actually the way other pages describe the pointed page so it can improve ranking of the pointed document.

In Wikipedia the <collectionlink> that generates the PageRank and AnchorText is pointing to a full document and not to its elements. Therefore in our indexing schema we infer the same document's PageRank and AnchorText to all elements of the document.

AnchorText terms are then added to the pointed document text. They can be weighted differently than regular document terms and this is something we need to explore in future work. Finally the element score is a weighted sum of its calculated score and its PageRank. In next section we describe different PageRank weights we used in our runs.

4 Runs and results

We describe our submissions and results for the CO & COS thorough and BestInContext runs.

Thorough runs

In all runs we ignored phrases (namely phrase terms were treated as simple terms) and we treat “+” terms as regular terms. We did respect “-“terms namely we never return a result which has a “-“term.

For all runs we picked minCompSize=20 and numIndices=7 and we added the AnchorText to the pointed documents with same weight as regular terms. We tried combinations of various PageRank weights with and without the “ontopic_keywords” part of the topic. We submitted 3 CO runs and 3 COS runs. For the COS runs we translated the topics to XML Fragments[1] and applied the component ranking algorithm as described in [6]. The results of our 3 CO runs (Table 2) using the Metric:ep-gr, Quantization: gen were superior to the COS runs. This strengthens our findings from last year that structural hints do not seem to contribute to the results.

Rank	Run id	MAep
8	PR=0.1, with keywords	0.0345
11	PR=0.1	0.0326
12	No PR	0.0324

Table 2. CO results for the thorough run

Comparing the 3 CO runs it seems that using PageRank with weight 0.1 (2nd run) did not improve our MAep over no PageRank (3rd run). The ontopic_keywords (1st run) did improve a bit (from 0.0326 to 0.0345) but we are more interested in the runs without the keywords.

BestInContext runs

The BestInContext run is based on a filtering step over results of a thorough run. We run a thorough run and then we go over the returned ranked list of elements and for each element if it is the first element from its document, we pick it as the best entry point for that document and remove all other elements from same document.

Our 3 CO runs used PR=0, 0.1, 0.2 respectively and for the published BEPD metric they were ranked first for some cutoff values (e.g. at A=100).

5 Discussion and summary

We described a scalable & robust component ranking algorithm that does not require any pre existing knowledge on the element types that are potential to be returned. This algorithm can work on the Wikipedia collection as well as on any heterogeneous collection. We further tried to improve the algorithm by incorporating PageRank and AnchorText but it doesn't seem to improve much.

As future work we should investigate the coverage of indexed elements out of all relevant elements in the assessment pool. We should explore other indexing algorithm that uses combination of direct size and total size for picking minimal elements. And finally we should try to tune the algorithm with more weights of PageRank and AnchorText terms.

References

- 1 Broder A.Z., Maarek Y., Mandelbrod M. and Y. Mass (2004): "Using XML to Query XML – From Theory to Practice". In Proceedings of RIAO'04, Avignon France, Apr , 2004.
- 2 Carmel D., Farchi E., Petruschka Y., Soffer A.: Automatic Query Refinement using Lexical Affinities with Maximal Information Gain. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002.
- 3 Carmel D., Maarek Y., Mandelbrod M., Mass Y., Soffer A.: Searching XML Documents via XML Fragments, In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada, Aug. 2003
- 4 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, Dagstuhl, Germany, pg 53-58
- 5 Y. Mass, M. Mandelbrod, Component Ranking and Automatic Query Refinement for XML Retrieval, Advances in XML Information Retrieval, LNCS 3493, INEX 2004, Dagstuhl Germany, December 2004, pg. 73-84
- 6 Y. Mass, M. Mandelbrod, Using the INEX Environment as a Test Bed for various User Models for XML Retrieval, LNCS 3977, INEX 2005, Dagstuhl Germany, November 2005, pg. 187-195

Indian Statistical Institute at INEX 2006 Adhoc track: A Preliminary VSM Approach

Sukomal Pal, Mandar Mitra, and Prasenjit Majumder

Information Retrieval Lab, CVPR Unit,
Indian Statistical Institute
203, B.T. Road, Kolkata -700 108, India
{sukomal_r, mandar, prasenjit_t}@isical.ac.in

Abstract. This paper describes the preliminary work that Indian Statistical Institute did towards XML retrieval for INEX 2006. As a beginner, we applied the Vector Space Model and made minor modifications to a widely used and popular text retrieval system (SMART) to retrieve XML documents against the INEX Adhoc queries. This year, we considered Content-Only(CO) queries and submitted two runs with retrieval at the document level. The result is not-at-all satisfactory, but it gives us enough confidence to explore the potential of the model and the system towards XML element retrieval which we aim to do in the coming years.

1 Introduction

Traditional Information Retrieval systems return whole documents in response to queries, but the challenge in XML retrieval is to return the most relevant parts of XML documents which meet the given information need. INEX [1] classified the adhoc retrieval task into two sub-tasks: Content-Only(CO) task and Content-And-Structure(CAS) task. In the CO task, the user poses the query in free text and the retrieval system is supposed to return the most relevant elements. A CAS query can provide explicit or implicit indications about what kind of element the user requires along with a textual query. Thus, a CAS query contains structural hints expressed in XPath [2] with an *about()* predicate.

Our retrieval approach this year was based on the Vector Space Model which sees both the document and the query as bags of words, and uses their *tf-idf* based weight-vectors to measure the *similarity* between the document and the query as angular distance (cosine-distance, to be specific) between these vectors. The documents are retrieved and ranked in decreasing order of the similarity-value.

We used the SMART system for our experiments at INEX 2006 and submitted two runs for the *thorough* task of the Adhoc track considering CO queries only. In the following section we describe our approaches for these two runs, and discuss results and further work in Section 3.

2 Approach

To extract the useful parts of the given documents, we shortlisted about thirty tags that contain useful information: $\langle p \rangle$, $\langle ip1 \rangle$, $\langle it \rangle$, $\langle st \rangle$, $\langle fnm \rangle$, $\langle snm \rangle$, $\langle atl \rangle$, $\langle ti \rangle$, $\langle p1 \rangle$, $\langle h2a \rangle$, $\langle body \rangle$, etc. Documents were parsed using the LIBXML2 parser, and only the textual portions included within the selected tags were used for indexing. Similarly, for the topics, we considered only the *title* and *description* fields for indexing, and discarded the *inex-topic*, *castitle* and *ontopic-keywords* tags. No structural information from either the queries or the documents was used.

The extracted portions of the documents and queries were indexed using Salton's blueprint for automatic indexing [5]. Stopwords were removed in two stages. First, we removed frequently occurring common words (like *know*, *find*, *information*, *want*, *articles*, *looking*, *searching*, *return*, *documents*, *relevant*, *section*, *retrieve*, *related*, *concerning*, etc.) from the INEX topic-sets. Next, words listed in the standard stop-word list included within SMART were removed from both documents and queries. Words were stemmed using a variation of the Lovin's stemmer implemented within SMART. Documents and queries were weighted using the *Lnu.ltu* [4] term-weighting formula. For each of 125 adhoc queries, we retrieved 1500 top-ranked XML documents.

2.1 Phrases

Documents and queries were indexed using single terms and a controlled vocabulary (or pre-defined set) of statistical phrases. For the baseline run, we used the default set of phrases used by Smart to index documents/queries. This phrase-list consists of all word-pairs that occur side-by-side 25 times or more in disk 1 of the TIPSTER collection. For the *isical-wiki-phrases* run, we used a set of phrases constructed from the Wikipedia corpus. We used the N-gram Statistics Package (NSP)¹ on the English Wikipedia text corpus and selected the most frequent 160,000 bi-grams as the list of possible phrases.

3 Result

The results reported for the two runs are shown below:

Table 1. Metric:ep-gr, Quantization: gen,Overlap=off

RunId	MAep Value
isical_baseline	0.0052
isical-wiki-phrase	0.0054

¹ <http://www.d.umn.edu/~tpederse/nsp.html>

Since an off-the-shelf IR system was used for the task with only the minimal changes required to make it run on the given data, and with little or no tuning for the task, it is not surprising that the results are fairly poor. However, with access to the relevance assessment data, we are now in a position to investigate how structural information may be used to implement effective element-level retrieval.

4 Conclusion

This was our first attempt at INEX. Our main objective this year was to check whether SMART can be customized for XML retrieval and to obtain a representatively large XML corpus. In subsequent years, we plan to incorporate important structural information into the indexing and retrieval stage.

References

1. INEX, Initiative for the Evaluation of XML Retrieval.
<http://inex.is.informatik.uni-duisburg.de/>
2. XPath-XML Path Language(XPath) Version 1.0.
<http://www.w3.org/TR/xpath>
3. Gerard Salton, editor. The SMART Retrieval System - Experiments in Automatic Document Retrieval. Prentice Hall Inc., 1988.
4. Chris Buckley, Amit Singhal, and Mandar Mitra. Using Query Zoning and Correlation within SMART: TREC5. E.M. Voorhees and D.K. Harman, editors. Proceedings of the Fifth Text Retrieval Conference(TREC-5). NIST Special Publication 500-238, 1997.
5. Gerard Salton. A Blueprint for Automatic Indexing. ACM SIGIR Forum, 16(2), pages 22–38. Fall 1981.

SIRIUS XML IR System at INEX 2006: Approximate Matching of Structure and Textual Content

Eugen Popovici, Gildas M  nier, Pierre-Fran  ois Marteau

VALORIA Laboratory, University of South-Brittany
BP 573, 56017 Vannes Cedex, France
{Eugen.Popovici, Gildas.Menier, Pierre-Francois.Marteau}@univ-ubs.fr

Abstract. This paper reports on the retrieval approach taken by the VALORIA laboratory of the University of South-Brittany while participating at INEX 2006. We describe the extensions made to the SIRIUS XML IR system to address each of the four subtasks of the INEX 2006 ad-hoc retrieval track (i.e. Thorough, Focused, Relevant In Context and Best In Context). Finally, we present and analyze the SIRIUS retrieval performance evaluation obtained within the INEX 2006 campaign.

1 Introduction

This preliminary study reports on the second year of experiments conducted by the VALORIA laboratory at the University of South-Brittany with the SIRIUS XML IR system [20] within the framework of the INEX evaluation campaigns.

The main contributions brought relatively to our last year participation are: i) the evaluation of the retrieval approach against a new collection and a new set of topics, ii) the implementation of the approximate search and indexing process using a distributed inverted file architecture; iii) the use of selective indexing profiles defining how the structure and the content of XML tags should be indexed, and iv) the addressing of new requirements for the Relevant In Context and Best In Context tasks. As for the last year we continue to investigate if and how the approximate match of the structural constraints in the queries may help retrieval and to experiment with different methods for removing overlapping elements.

The paper is organized as follows. In Section 2 we present the main functionalities and characteristics of the SIRIUS XML IR system. In Section 3 we introduce our retrieval approach for the INEX 2006 ad-hoc task. In Section 4 we present and analyze the SIRIUS retrieval evaluation results for the Thorough, Focused, and Best In Context tasks. Finally, in Section 5 we conclude the paper.

2 SIRIUS XML IR System

Approximate matching in XML provides the ability of querying the information managed by a system having an incomplete or imprecise knowledge about both the

structure and the content of the XML documents [14, 15]. In this context, we propose and experiment a lightweight model for indexing and querying XML documents implemented in the SIRIUS XML IR system.

SIRIUS [6, 7, 20] is a lightweight indexing and search engine for XML documents developed at the VALORIA laboratory of the University of South-Brittany. The retrieval approach implemented in SIRIUS is document oriented. It involves an approximate matching scheme of the structure and textual content. Instead of managing the matching of whole XML trees, SIRIUS splits the documents object model in a set of paths. This set is indexed using optimized data structures. In this view, the request is a path-like expression with conditions on the attribute values. For instance `/document(> date "1994")/chapter(= number 3)/John` is a request aiming to extract the documents (written after 94) with the word John in the chapter number 3. We designed a matching process that takes into account mismatched errors both on the attributes and on the XML elements. The matching process uses a weighted editing distance on XML paths: this provides an approximate matching scheme able to manage jointly the request on textual content and on document structure. The search scheme is extended by a set of IR retrieval operators, and features a set of thesaurus rewriting rules. Recently the system has been extended with a specialized set of operators for extracting, indexing and searching heterogeneous sequential and time series data embedded in heterogeneous XML documents [8].

2.1 Indexing Scheme

Each element in an XML document may be composed of a set of possible nested XML elements, textual pieces of information (TEXT or CDATA), unordered <attribute, value> pairs, or a mixture of such items. XML documents are generally represented as rooted, ordered, and labeled trees in which each node corresponds to an element and each edge represent a parent-child relationship.

XML Context. According to the tree structure, every node n inherits a path $p(n)$ composed with the nodes that link the root to node n . This path is an ordered sequence of XML elements potentially associated to unordered <attribute, value> pairs $A(n_i)$, that determines the XML context in which the node is occurring. A tree node n , containing textual/mixed information can be decomposed into textual sub-elements. Each string s (or word, lemma, ...) of a leaf node is also linked to $p(n)$. This XML context characterizes the occurrence of s within the document and can be represented as follows:

$$p(n)=\langle n_0, A(n_0)\rangle \langle n_1, A(n_1)\rangle \dots \langle n, A(n_n)\rangle \quad (1)$$

Index Model. The indexing process involves the creation of an enriched inverted list designed for the management of these XML contexts. For this model, the entries of the inverted lists are the textual sub-elements s of a tree node. For a sub-element s of a node n , four pieces of information are attached:

- a link to the URI of the document `<fileId>`,

- the *<preorder>* and *<postorder>* positions of the node n in the XML tree,
- an index specifying the positions of s within the document *<wordOffset>*,
- a link toward its XML context $p(n)$ *<ctxId>*.

2.2 Searching Scheme

Most of the time, for large heterogeneous databases, one cannot assume that the user knows all of the structures – even in the very optimistic case, when all of the structural properties are known. Some straightforward approaches (such as the XPath search scheme [16]) may not be efficient in these cases. As the user cannot be aware of the complete XML structure of the data base due to its heterogeneity, efficient searching should involved exact and approximate search mechanisms.

The main structure used in XML is a tree: It seems acceptable to express a search in term of tree-like requests and approximate matching. We proposed [6], to focus on path matching rather than on tree matching – in a similar way with the XML fragment approach [15]. The request should be expressed as a set of path $p(r)$ that is matched with the set of sub-path $p(n)$ in the document tree. This breaks the algorithmic complexity of tree matching techniques while still providing high precision results [5]. This ‘low-level’ matching only manage subpath similarity search with conditions on the elements and attributes matching. This process is used to design a more higher-level request language: a full request is a tree of low-level matching goals (as leafs) with set operators as nodes. These operators are used to merge leaf results. The whole tree is evaluated to provide a set of ranked answers. The operators are classical set operators (intersection, union, difference) or dedicated fuzzy merging processors. The system analyzes a request and produces a set of weighted results. Let $\{ (e_i, v_i) \}$ the set of weighted results produced by the system, where e_i is a an element of the result and $v_i \in [0..1]$ a weight showing the relevance of the returned element to the request.

Textual Content Ranking Scheme. We compute the relevance value $v_i \in [0..1]$ for all the XML elements e_i containing at least one researched term τ_k of a content only request CO . The ranking scheme takes into account the number and the discriminating power of the retrieved terms in the collection. We used a dedicated TFIDF [18] function for this purpose:

$$v_i(e_i, CO) = \xi \cdot \sum_k \lambda_k \cdot \tau_k,$$

where k is the number of terms τ_k in the CO request, λ_k is an IDF weighting factor specifying the discriminating power of the term τ_k in the collection : $\lambda_k = 1 - \log((1 + |D \tau_k|) / (1 + |D|))$; where $|D \tau_k|$ is the number of documents in which τ_k is occurring; $|D|$ the total number of documents in the collection; and ξ a normalization constant $\xi = 1 / \sum_k (\lambda_k)$;

Approximate Path Search. Let p^R be a structural constraint, expressed as a path goal with conditions or constraints to be fulfilled on the attributes. We investigate the similarity between a p^R (coding a path with constraints) and p_i^D (a root/./terminal(r) path of the tree T^D associated to an index document D) as follow:

$$\sigma(p^R, p_i^D) = 1/(1 + \delta_L(p^R, p_i^D)) \quad (2)$$

where δ_L is a dedicated editing distance (see [12]).

The search complexity is $O(l(p^R).deep(T^D).| \{ p_i^D \} |)$ with $| \{ p_i^D \} |$ the size of the set $\{ p_i^D \}$ (i.e. the number of different paths in D , starting at the root and leading to the last element of the p^R request – terminal(r)), $l(p)$ the length of the path p and $deep(T)$ the deepest level of T . This complexity remains acceptable for this application as 99% of the XML documents have fewer than 8 levels and their average depth is 4 [13]. We designed [6] an editing pseudo-distance using a customised cost matrix to compute the match between a path p_i^D and the request path p^R . This scheme, also known as modified Levenshtein distance, computes a minimal sequence of elementary transformations to get from p_i^D to p^R . The elementary transformations are:

- **Substitution:** a node n in p_i^D is replaced by a node n' for a cost $C_{subst}(n, n')$.
- **Deletion:** a node n in p_i^D is deleted for a cost $C_{del}(n)$,
- **Insertion:** a node n is inserted in p_i^D for a cost $C_{ins}(n)$.

Weighting Scheme for INEX. The NEXI language [3] allows only the descendant relationship between the nodes in a path. Therefore the XML path expressed in the request is interpreted as a *subsequence* of an indexed path, where a subsequence need not consist of contiguous nodes. To model this, we relaxed in [20] the weights of the path editing distance in order to allow node deletions in the indexed paths without any penalty: $C_{del}(n) = 0$, $C_{ins}(n) = \xi$, and $C_{subst}(n, n') = \xi$. Since a node n not only stands for an XML element but also for attributes or attributes relations, we compute $C_{subst}(n, n')$ as follows: $C_{subst}(n, n') = \{ \xi \text{ if } (n \neq n'); \frac{1}{2}\xi \text{ if } (n = n') \ \& \ (\neg attCond(n')) \}; 0 \text{ if } (n = n') \ \& \ (attCond(n')) \}$, where $attCond$ stands for a condition stated in the request that should apply to the attributes.

For a sequence $Seq(p_i^D, p^R)$ of elementary operations, the global cost $GC(Seq(p_i^D, p^R))$ is computed as the sum of the costs of elementary operations. The Wagner&Fisher algorithm [12] computes the best $Seq(p_i^D, p^R)$ (i.e. minimizes $GC()$ cost) with a complexity of $O(length(p_i^D) * length(p^R))$ as stated earlier. Let

$$\delta_L(p^R, p_i^D) = Min_k GC(Seq_k(p^R, p_i^D)). \quad (3)$$

Given p^R and p_i^D , the value for $\sigma(p^R, p_i^D) \rightarrow 0$ when the number of mismatching nodes and attribute conditions between p^R and p_i^D increases. For a perfect match $\sigma(p^R, p_i^D) = 1$, i.e. all the elements and the conditions on attributes from the request p^R match correspondent XML elements in p_i^D .

The weights used to compute the structural similarity relate to an end user having precise but incomplete information about the XML tags of the indexed collection and about their ancestor-descendant relationships. The structural similarity takes into account the order of occurrence of the matched nodes and the number of nodes with no matching in the request. It heavily penalizes any mismatch relatively to the

information provided by the user but it is independent to mismatches/extra information extracted from the indexed paths.

Merging Structure and Content Matching Scores. We add structural matching information to the set of solutions returned by the system using a weighted linear aggregation between the conditions on structure $\sigma(p^R, p_i^D)$ and the initial/textual ranking score v_i as follows:

$$v'_i = \beta \cdot \sigma(p^R, p_i^D) + (1 - \beta) \cdot v_i.$$

The value of the $\beta \in [0..1]$ parameter may be used to emphasize the importance of the structural versus textual content matching scores.

3 SIRIUS Approach for the INEX 2006 Ad-hoc Task

The retrieval task we are addressing at INEX 2006 is the ad-hoc retrieval of XML documents. This involves the searching of a document collection of 4.6 GB made of 659,388 English articles from Wikipedia using a set of 125 topics. The structural part of the collection corresponds to the Wikipedia templates (about 5000 different tags). The topics may contain both content and structural conditions and, in response to a query, arbitrary XML elements may be retrieved by the system. An example of an INEX 2006 topic with the *title* and *castitle* expressed in NEXI language [3] is given in Fig. 1.

```
<inex_topic topic_id="406" ct_no="198">
  <title>book architecture</title>
  <castitle>//template[about(./@name,book reference)]/*[about(.,architecture)]</castitle>
  <description>Show me books about architecture</description>
  <narrative>After coming home from a trip to venice...</narrative>
  <ontopic_keywords>+house</ontopic_keywords>
</inex_topic>
```

Fig. 1. An excerpt of the INEX 2006 topic 406.

Content only (CO) queries contain just search terms (see the *title* part in Fig. 1) while the content and structure (CAS) queries (see the *castitle* part in Fig. 1) are topic statements that contain explicit references to the XML structure, and explicitly specify the contexts of the user's interest (e.g. target elements) and/or the context of certain search concepts (e.g. support elements).

3.1 Indexing the Wikipedia Collection

This year we added to the SIRIUS system the capability of using indexing profiles for a specific collection. The *indexing profiles* are composed of rules defining how the structure and the content of each specified XML tag should be indexed. By default, all the non empty XML tags are fully indexed. Using these profiles we may decide or not to index the attributes associated to a given tag, to index only the content of the

presentation tags or *jump tags* [9], or to completely ignore some *logical tags* for a specific collection. The use of indexing profiles may reduce significantly the volume of the requested disk space for the index and improves the system performances both in indexing and retrieval time.

We use the rules shown in Table 1. to index the Wikipedia collection. This indexing profile was manually defined as we assumed that the jump and presentation tags contained information that should not be retrieved out of their context. The logical tags `<name>`, `<title>` and `<caption>` are of a particular importance for the Wikipedia collection, as this will ensure that the `<title>` of a `<section>` will always be retrieved with the `<section>` itself, that the `<name>` of an `<article>` will be retrieved with the whole `<article>`, and that the `<caption>` of a `<figure>` or `<table>` will be retrieved only associated to the element to which they are referring to.

Table 1. Indexing rules for the Wikipedia collection.

	Ignore tags	Ignore tag attributes
Presentation tags	emph2, emph3, emph4, sup	table, tr, td, font
Jump tags	Collectionlink, unknownlink, outsidelink, languagelink	
Logical tags	title, name, image, caption	

The Wikipedia collection is processed using an XML SAX parser and standard methods for stop words removal and stemming. At indexing time, the most frequent words are eliminated using a stop list. The XML elements containing no valid textual content after stop words removal are not indexed. The index terms are stemmed using the Porter algorithm [17]. The index model (Section 2.1) is implemented on top of the Berkeley DB¹ library using a combination of BTrees and Hashtables structures. The inverted file index is constructed in parallel by using a *Physical Document Partitioning* approach [4]. The total size of the index is about 86% of the initial database size – i.e. 4GB.

3.2 Processing NEXI Requests

Processing CO requests. CO queries are INEX topics containing only textual search terms (i.e. see the *title* part in Fig. 1). We compute the relevance score for all the leaves elements of the XML tree containing at least one of the researched terms using a variant of the TF-IDF ranking scheme (i.e. IDF) (see Section 2.2). In our approach we consider the XML element containing a researched term as the basic and implicitly valid unit of retrieval regardless of its size.

Processing CAS requests. For CAS topics, we have two cases: simple queries of the form `//A[B]` – i.e. the request specifies only the target elements, and complex queries of the form `//A[B]//C[D]` – i.e. the request specifies both target (i.e. `//C[D]`) and support (i.e. `//A[B]`) elements.

¹ <http://www.sleepycat.com/>

Processing the Support and Target Elements. For simple type queries of the form $//A[B]$ like $//template//*[about(.,architecture)]$ (see topic in Fig. 1), we rank the textual content of the nodes using the same ranking scheme as for the CO requests. The structural constraints from the requests are interpreted as structural hints [3]. We compute the similarity between the structural constraints expressed in the request – i.e. $//template//*$ – and the XML paths of the candidate fragments using a modified editing distance [20] involving specific heuristics for attributes and attributes values. Finally we merge the content and structural match scores using a weighted linear aggregation method (see Section 2.2).

Processing the Containment Conditions. To process complex queries of the form $//A[B]//C[D]$ (see the *castitle* part in Fig. 1) we compute the relevance for both the support elements $//A[B]$ and target elements $//C[D]$. Next, we select only the target elements that have at least a relevant support element occurring in the same document. The logic behind this is that if a relevant support element exists in a document, its weight should be propagated using a *max* function to the root node of the XML tree that is an ancestor – i.e. support element – for all the elements of the tree. This applies inclusively to the target elements.

The similarity computation for a complex request involves modifications of the relevance associated with a result element. The relevance of a result element is computed as the arithmetic average between the relevance of the target element and the maximum relevance of its support elements.

Formally, let $\{(e_i, v_i)\}$ the set of target results, $\{(e_j, v_j)\}$ the set of support elements, where e_i is an element of the result and $v_i \in [0..1]$ its relevance weight. Let e^D a descendant of document D . The set of weighted results produced by the system is $\{(e_i^D, v'_i)\}$ with $v'_i = (v_i + \text{Max}_j(v_j)) / 2$ where $\exists e_j^D \in \{(e_j, v_j)\}$.

Using this approach, the target elements that have no support elements are discarded from the final answers, while the ones supported by highly relevant elements are boosted in the final ranking. The final results are sorted by relevance values and the top N results returned.

4 Experimental Results

We submitted a total of 20 runs to all of the four tasks of the ad-hoc retrieval track: Thorough, Focused, Relevant In Context and Best In Context tasks. In all the submitted runs we used the same basic retrieval approach:

- To answer INEX 06 topics, we use automatic transformation of the *title* and *castitle* part of the topics expressed in NEXI [3] to SIRIUS recursive query language as described in [20].

CO runs

- The XML elements directly containing the research terms are considered as independent and the only valid units of retrieval;
- IDF weighting for textual content of the leaf nodes containing the researched terms (i.e. **IDF**) (see Section 2.2);

- Strict and vague search for phrase matching. In the strict sequence matching runs the researched terms must occur in sequence and belong to the same XML element. This is not required for the vague phrase matching runs (i.e. **noSEQ**) that rank as best results the XML elements containing all the researched terms without taking into account their order of occurrence.

*CAS runs (*cas*)*

- The structural constraints on both the support elements (where to look) and on the target elements (what to return) are interpreted vaguely, as structural hints. The vague interpretation of the structural constraints is implemented using a modified editing distance (**EDs**) on the XML paths with conditions on attributes and attributes values (see Section 2.2) .
- We use weighted linear aggregation for content and structure matching scores. (see Section 2.2) The runs (**W0_1**, **W0_5**) use different values for the β parameter to emphasize the importance of the structural versus textual content matching (i.e. $\beta = 0.1$ biases the ranking towards the textual content while $\beta = 0.5$ uses equal weights for merging the structural and content matching relevance scores).
- We use boolean (**BOOL**) merging operators at document level.

4.1 Thorough Task

At the Thorough task, the system estimates the relevance of elements in the collection. We submitted five runs identified by runId's using combinations of the abbreviations introduced above. We report in Fig. 2. the evaluation curves for the ep/gr metric and in Table 2. the official system-oriented MAep measure and the ranks obtained by all the submitted runs . Details of the evaluation metrics can be found in [2].

Table 2. Task: Thorough, Metric:ep-gr, Quantization: gen, Overlap=off

<i>RunId</i>	<i>MAep</i>	<i>Rank</i>
IDF_BOOL_noSEQ	0.0158	42/106
IDF_BOOL	0.0151	45/106
casEDsW0_1_IDF_BOOL_noSEQ	0.0146	48/106
casEDsW0_5_IDF_BOOL_noSEQ	0.0134	50/106
casEDsW0_5_IDF_BOOL	0.0130	51/106

We were ranked on the 42, 45, 48, 50 and 51 places from 105 submissions. This is not surprising as the implementation of our approach is biased towards focused retrieval. A rather surprising result is the fact that using the structural hints does not improve the retrieval quality of the results. Rather the opposite. This is contradictory with the evaluation results obtained for the INEX2005 ad-hoc collection and topics [20]. The best overall performance is obtained by the run using only the textual content and no phrase constraints (IDF_BOOL_noSEQ) with a MAep value of 0.0158.

4.2 Focused Task

The aim of the *Focussed* retrieval strategy is to find the most exhaustive and specific element in a path. In other words, the result list should not contain any overlapping elements. In our approach for the Thorough task we consider the XML element containing a researched term as the basic and implicitly valid unit of retrieval regardless of its size. This approach “naturally” implements a focused strategy as it returns the most focused elements containing the research terms. However, cases where nested/overlapping XML elements could be returned as valid results may occur.

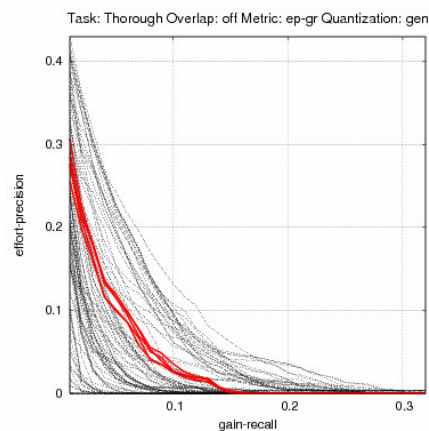


Fig. 2. INEX'2006 Result Summary: Task: Thorough, Metric: ep/gr, Quantization: gen, Overlap: off.

For the Focused runs we use a similar approach as for the Thorough task and implement a two steps post filtering process to remove the overlapping elements from the results [20]: i) we recalculate the relevance of the elements in the answer list in order to reflect the relevance of their descendant elements (if any); and ii) we select non overlapping elements from the list.

The weights are calculated recursively starting at leafs to the highest non overlapping nodes composing the answer by using two strategies:

- *MAX* - the max relevance value is propagated recursively to the highest non overlapping elements; and
- *AVG* - the relevance of a node is computed as the arithmetic average of all its descendant relevant nodes including its own relevance.

To select the non overlapping elements we compared the following strategies:

- *HA* - the highest ancestor from the answer list is selected;
- *MR* - the most relevant answer is selected recursively from the answer list as long as it not overlaps with an already selected element – i.e. for equally relevant

overlapping elements we choose either the descendant (*MRD*) or the ancestor (*MRA*).

We experimented with different settings for computing the elements relevance and selecting the non overlapping answers for the Focused tasks within the framework of the INEX 2005 campaign [20]. This year we selected only the MAX_MRD and MAX_HA strategies for the focused task as they obtained the best results during the INEX 2005 evaluation.

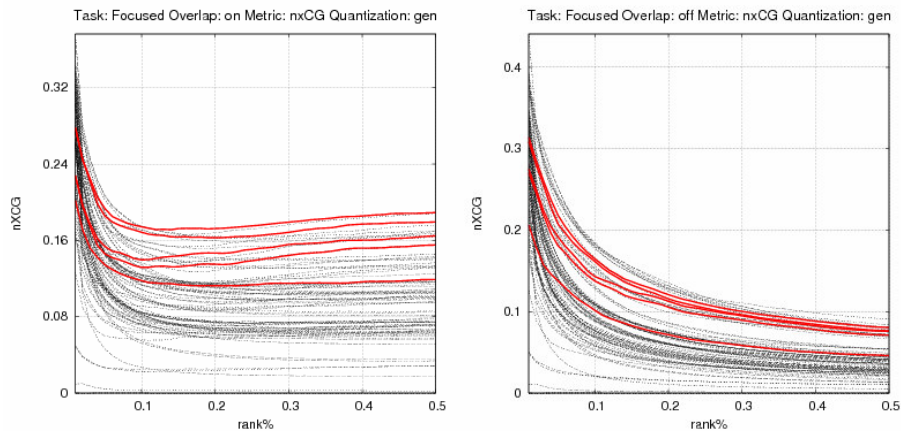


Fig. 3. INEX'2006 Result Summary: Task: Focused Metric:nxCG, Quantization: generalised, Overlap=on (left fig.) ; and Overlap=off (right fig.)

Table 3. Task: Focused, Metric: nxCG, Quantization: generalised, Overlap=on.

RunId	nxCG@5	nxCG@10	nxCG@25	nxCG@50				
IDF_BOOL_noSEQ_MAX_MRD	0.2882	47	0.2759	24	0.2393	13	0.2095	6
IDF_BOOL_MAX_MRD	0.2889	45	0.2695	28	0.2391	14	0.2022	9
casEDsW0_5_IDF_BOOL_noSEQ_MAX_MRD	0.2335	66	0.2215	60	0.1965	35	0.1638	29
casEDsW0_5_IDF_BOOL_MAX_MRD	0.2338	65	0.2202	61	0.1933	40	0.1572	35
casEDsW0_1_IDF_BOOL_noSEQ_Foc_MAX_HA	0.2055	73	0.1996	65	0.1693	54	0.1436	46

Table 4. Task: Focused, Metric: nxCG, Quantization: generalised, Overlap=off.

RunId	nxCG@5	nxCG@10	nxCG@25	nxCG@50				
IDF_BOOL_noSEQ_MAX_MRD	0.3227	38	0.3238	16	0.2807	12	0.2424	9
IDF_BOOL_MAX_MRD	0.3180	40	0.3093	21	0.2768	14	0.2339	12
casEDsW0_5_IDF_BOOL_noSEQ_MAX_MRD	0.2770	60	0.2829	36	0.2475	21	0.2071	20
casEDsW0_5_IDF_BOOL_MAX_MRD	0.2719	62	0.2735	41	0.2418	27	0.2002	21
casEDsW0_1_IDF_BOOL_noSEQ_Foc_MAX_HA	0.2073	73	0.2063	66	0.1779	59	0.1477	46

We report here the nxCG values @5, @10, @25 and @50 (see [2] for metric descriptions) for all the submitted focused runs, along with their official ranks in the INEX06 campaign (see Table. 3, Table 4. and Fig. 3.).

For the Focused task, the system is better ranked than on the Thorough task regardless if it is evaluated with the overlap 'on' or 'off'. SIRIUS has three results in

in the top ten runs: a 6/85 rank for IDF_BOOL_noSEQ_MAX_MRD with $nxCG@50=0.2095$ and overlap on, a 9/85 rank for IDF_BOOL_MAX_MRD with $nxCG@50=0.2022$ with overlap ‘on’, and a 9/85 rank for IDF_BOOL_noSEQ_MAX_MRD with $nxCG@50=0.2424$ with overlap ‘off’.

By analyzing the $nxCG$ curves of Fig. 3 we observe that the SIRIUS runs have a good recall. This may be explained by the fact that in the approach taken for the INEX 2006 ad-hoc task we return mostly leaf elements which are shown to be effective retrieval units for the focussed strategy [19]. We also observe a slightly decrease in the system retrieval performance for the first ranked results relative to the INEX 2005 evaluation [20]. This may be determined by the indexing configuration settings (see Table 1.). The indexing profile did not allowed for a large number of small/possibly relevant focused elements (i.e. jump tags & presentation tags) to be retrieved. We observe that as for the Thorough task, the runs involving structural conditions performed worse than their content only pairs.

4.3 Relevant in Context Task,

The INEX 2006 Relevant In Context task has to find a set of elements that corresponds well to (all) relevant information in each article. The relevant elements must be clustered per article and ordered in their original document order when returned to the user. The assumption is that users consider the article as the most natural unit, and prefer an overview of relevance in their context.

For this task, we used as starting point the approach used for the Focused runs. We propagate the max relevance obtained for the non overlapping elements at the file level. The files are ranked by their relevance. We clustered the non overlapping results by file and ranked them according to their relevance inside each file. We returned the top N relevant results for each file, where $N=\{5, 10\}$ until reaching the INEX 2006 max results limit per topic (i.e. 1500 results).

The evaluation results for this task were not yet available at the moment of writing this paper.

4.4 Best in Context Task

Best In Context task returns a ranked list of articles. For each article, it returns a single element, representing the best entry point for the article with respect to the topic of request. For this task we used the same approach as for the Relevant In Context Task with N set to 1. The official results evaluated with BEP-D (see Table 5.) and EPRUM-BEP-Exh-BEPDistance [10] (see Table 6.) were ranked several times in the top ten positions out of 77 submitted runs. The top ten results are highlighted while the best obtained values are emphasized.

Table 5. Task: Best In Context. Metric: BEPD.

<i>RunId</i>	<i>A=0.01</i>	<i>A=0.1</i>	<i>A=1</i>	<i>A=10</i>	<i>A=100</i>					
IDF_BOOL_noSEQ_AVG_MRD	0.1959	1	0.2568	2	0.3642	6	0.5596	6	0.7556	7
IDF_BOOL_MAX_HA	0.1722	2	0.2753	1	0.4095	1	0.5847	3	0.7542	8
casEDsW0.1_IDF_BOOL_noSEQ_MAX_HA	0.1394	16	0.2303	8	0.3580	7	0.5239	18	0.6853	27
casEDsW0.5_IDF_BOOL_noSEQ_MAX_HA	0.1346	17	0.2222	12	0.3447	12	0.5048	24	0.6631	36
casEDsW0.5_IDF_BOOL_MAX_HA	0.1322	19	0.2114	17	0.3222	23	0.4691	36	0.6170	45

Table 6. Task: Best In Context. Metric:EPRUM-BEP-Exh-BEPDistance.

<i>RunId</i>	<i>A=0.01</i>		<i>A=0.1</i>		<i>A=1</i>		<i>A=10</i>		<i>A=100</i>	
IDF_BOOL_noSEQ_AVG_MRD	0.0407	1	0.0579	8	0.0873	13	0.1489	16	0.2193	35
IDF_BOOL_MAX_HA	0.0304	4	0.0607	6	0.1069	7	0.1770	8	0.2536	14
casEDsW0.1_IDF_BOOL_noSEQ_MAX_HA	0.0233	24	0.0478	15	0.0881	12	0.1480	19	0.2180	36
casEDsW0.5_IDF_BOOL_noSEQ_MAX_HA	0.0218	31	0.0444	24	0.0812	20	0.1363	34	0.2031	42
casEDsW0.5_IDF_BOOL_MAX_HA	0.0214	34	0.0435	29	0.0785	23	0.1323	38	0.1969	44

The Best In Context task results confirmed that the runs using structural hints (**cas**) are ranked lower than the ones using only the textual content. We have a single content and structure run in the top ten results *casEDsW0.1_IDF_BOOL_noSEQ_MAX_HA* for $A=0.1$ when evaluated with the BEPD metric (see Table 5.). The AVG_MRD method for overlap removal gives better results than the MAX_HA technique. This may be considered with care, as may also be an effect of relaxing the constraints on phrase searching for the *IDF_BOOL_noSEQ_AVG_MRD* run – ranked on 1st /77 position by both the BEPD and EPRUM-BEP-Exh-BEPDistance metrics.

5 Conclusions

This year, at INEX 2006, we have pursued the evaluation of the retrieval performances of the SIRIUS XML IR system [6, 7, 8] started last year within the INEX 2005 campaign [20]. SIRIUS retrieves relevant XML elements by approximate matching both the content and the structure of the XML documents. A modified weighted editing distance on XML paths is used to approximately match the documents structure while the IDF of the researched terms are used to rank the textual contents of the retrieved elements. A number of extensions were brought to the system in order to cope with the requirements of the Thorough, Focused, Relevant In Context and Best In Context tasks.

We have submitted and evaluated 20 valid runs in all the INEX 2006 ad-hoc tasks, and showed the system ability to retrieve relevant non overlapping XML elements within the Focused and Best In Context tasks. SIRIUS obtained average rankings for the Thorough task and good quality results in the range of the 50 first ranked answers for the Focused task (see Fig. 3.). For the Best In Context task the results were quite encouraging as the system was ranked several times within the top ten runs out of 77 submissions. (see Table 5., and Table. 6.).

The runs using structural constraints were consequently outperformed by the runs using content only conditions, while the runs using strict constraints for phrase searching were outperformed by their variants using no constraints for phrases.

Our experiments at INEX 2005 showed that taking into account the structural constraints improved the retrieval performances of the system and jointly showed the effectiveness of the proposed weighted editing distance on XML paths for this task. This observation was not confirmed by any of the tasks evaluated at INEX 2006. More experimental studies are necessary to better understand the reasons for this behaviour.

References

1. Clarke C., Kamps J., Lalmas M., INEX 2006 Retrieval Task and Result Submission Specification, In INEX 2005 Workshop Pre-Proceedings, Dagstuhl, Germany, December 18–20, 2006.
2. Kazai, G., Lalmas, M., INEX 2005 Evaluation Metrics. In *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005)*, LNCS, Vol 3977, 2006.
3. Trotman A., & Sigurbjörnsson B., Narrowed Extended XPath I (NEXI) In *Proceedings of the INEX 2004 Workshop*, p. 16-40, 2004 .
4. Baeza-Yates R. and Ribeiro-Neto B. *Modern Information Retrieval*. ACM Press. Addison-Wesley, New-York, 1999.
5. Amer-Yahia S., Koudas N., Marian A., Srivastava D., and Toman D., Structure and Content Scoring for XML, VLDB 2005 Trondheim, Norway, pages 361–372, 2005.
6. Ménier G., Marteau P.F., Information retrieval in heterogeneous XML knowledge bases, The 9th International Conference on Information Processing and Magement of Uncertainty in Knowledge-Based Systems, 1-5 July 2002, Annecy, France, 2002.
7. Ménier G., Marteau P.F., PARTAGE: Software prototype for dynamic management of documents and data, ICSSEA, 29 Nov-1 Dec. 2005, Paris, France, 2005.
8. Popovici E., Marteau P.F., Ménier G., Information Retrieval of Sequential Data in Heterogeneous XML Databases, AMR 2005, 28-29 July 2005, Glasgow, UK, 2005.
9. Tannier X., Girardot J.-J., and Mathieu. M., Classifying XML Tags through “Reading Contexts”. In P. R. King, editor, *Proceedings of the 2005 ACM Symposium on Document Engineering*, pages 143–145, Bristol, United Kingdom, Nov. 2005. ACM Press, New York City, NY, USA.
10. Piwowarski B., Dupret G., Evaluation in (XML) information retrieval: expected precision-recall with user modelling (EPRUM). SIGIR 2006, pp 260-267.
11. Levenshtein A., Binary Codes Capable of Correcting Deletions, Insertions and Reversals, *Sov. Phy. Dohl*. Vol.10, P.707-710, 1966.
12. Wagner R., Fisher M., The String-to-String Correction Problem, *Journal of the Association for Computing Machinery*, Vol.12, No.1, p.168-173, 1974.
13. Mignet L., Barbosa D., Veltri P., *The XML Web: A First Study*, WWW 2003, May 20-24, Budapest, Hungary, 2003.
14. Carmel D., Maarek Y. S., Mandelbrod M., Mass Y. and Soffer A., Searching XML documents via XML fragments, SIGIR 2003, Toronto, Canada p. 151-158, 2003.
15. Fuhr N., Großjohann K., XIRQL: An XML query language based on information retrieval concepts, TOIS, v.22 n.2, p.313-356, April 2004.
16. Clark J., DeRose S., XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999, <http://www.w3.org/TR/xpath.html>, 1999.
17. Porter M.F., An algorithm for suffix stripping, *Program*, 14(3):130-137, 1980.
18. Salton G. and Buckley C., Term-weighting approaches in automatic text retrieval, *Information Processing and Management*, 24, p. 513-523, 1988.
19. Mihajlovic V., Ramirez G., Westerveld T., Hiemstra D., Blok H. E., de Vries A., *TIJAH Scratches INEX 2005: Vague Element Selection, Overlap, Image Search, Relevance Feedback, and Users'*, INEX 2005 Workshop Pre-Proceedings, Dagstuhl, Germany, November 28–30, 2005, p. 54-71, 2005.
20. Popovici E., Ménier G., and Marteau P.-F., SIRIUS: A Lightweight XML Indexing and Approximate Search System at INEX 2005, In *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005)*, LNCS, Vol 3977, pages 321-335, 2006.

A Method of Preferential Unification of Plural Retrieved Elements for XML Retrieval Task

Hiroki Tanioka

Innovative Technology R&D, JustSystems Corporation,
Brains Park Kawauchi-cho Tokushima-shi Tokushima, Japan
hiroki.tanioka@justsystem.co.jp

Abstract. We developed a passage retrieval system for XML documents with the vector space model. We also developed a method of unification of plural retrieved elements using XML Path Language (XPath) and Fragment Indexing System (FIS) we called. Therefore our system, which composed of the fragmentary inverted file and the simplified XML database, was confirmed the validity of the method from the results of Adhoc Track in the Initiative for the Evaluation of XML Retrieval (INEX) 2006.

1 Introduction

In the research field of document information retrieval, the unit of retrieval results returned by information retrieval systems is a whole document or a document fragment, like a paragraph in passage retrieval. Generic information retrieval systems based on the vector space model compute feature vectors of the units and calculate the similarities between the units and the query. However, the unit of retrieval results is unfit yet for document information retrieval, since it is not what users looking for.

Therefore, the unit of retrieval results should be a portion of XML document, such as a chapter, section, or subsection. This is undeniable that all XML documents consist of several portions and contain some portions which are meaningful to the users. It is easier to construct the appropriate portion of XML documents as the unit of retrieval results than unstructured plaintexts, because XML is a standard document format consists of contents and document structures, and also XML format is widely used on the Internet.

But nonetheless the appropriate fragmentation problem remains still unsolved, which is difficulties of XML passage retrieval task, like the automatic text summarization in document processing and the object segmentation in image processing. Therefore, we propose an efficient index structure and flexible information retrieval system for XML documents.

In the objective described above, we examine the “applicability” and the “scalability” of fragmentary indexing approach for the generic information retrieval model, by developing an XML retrieval system and testing with INEX corpus. This article describes the points and the validation results of our proposed index structure and the information retrieval system called Fragment Indexing System (FIS). Then, our motivations in INEX 2006 are listed bellow.

1. To verify the “applicability” of fragmentary indexing approach for generic information retrieval model, we check the accuracy of retrieving results.

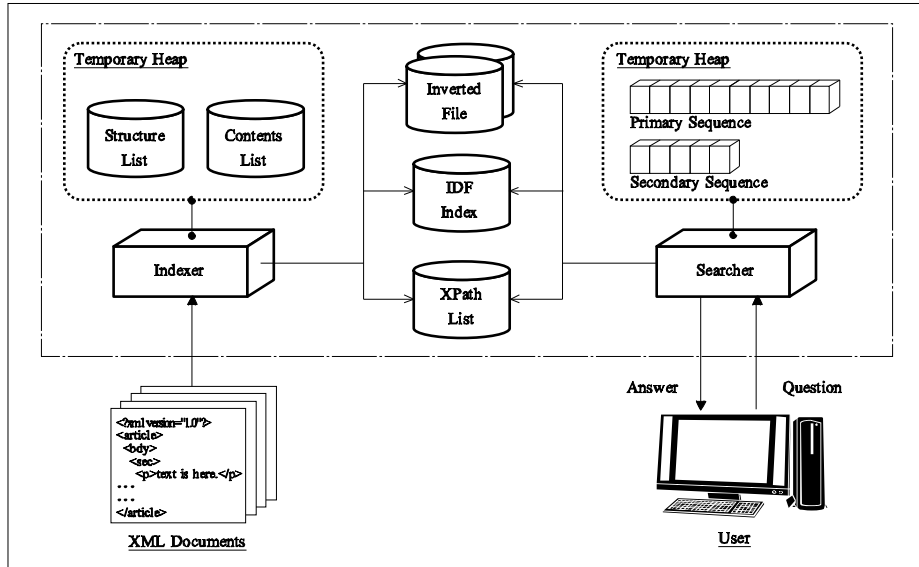


Fig. 1. System description

2. To test the “scalability” of the XML retrieval system, we measure the following size and time.
 - (a) The size of an inverted file – spatial scalability (*space complexity*)
 - (b) The processing time of retrieving – temporal scalability (*time complexity*)

The rest of the article is divided into three sections. Section 2, we describe an architecture of our indexing and retrieving system for XML documents. Section 3, we describe experimental results. And Section 4, we discuss about results and future works.

2 System Description

In this section we describe the architecture of our distributed search system as shown in Figure 1 and information retrieval models including an indexing method with fragment indexing algorithm and an scoring method with preferential unification algorithm.

2.1 Index

We develop a distributed search system which is based on the vector space model using term (as word) partitioning with an inverted file-based system, while a single inverted file is created for the document collection and the inverted lists are spread across the processors[6]. During query evaluation, the query is decomposed into indexing items and each indexing item is sent to the processor that holds the corresponding inverted file[2].

And our distributed search system bring in the simplified XML database. Hence we create our runs using two types of inverted files, one for XML articles and the others for all XML elements.

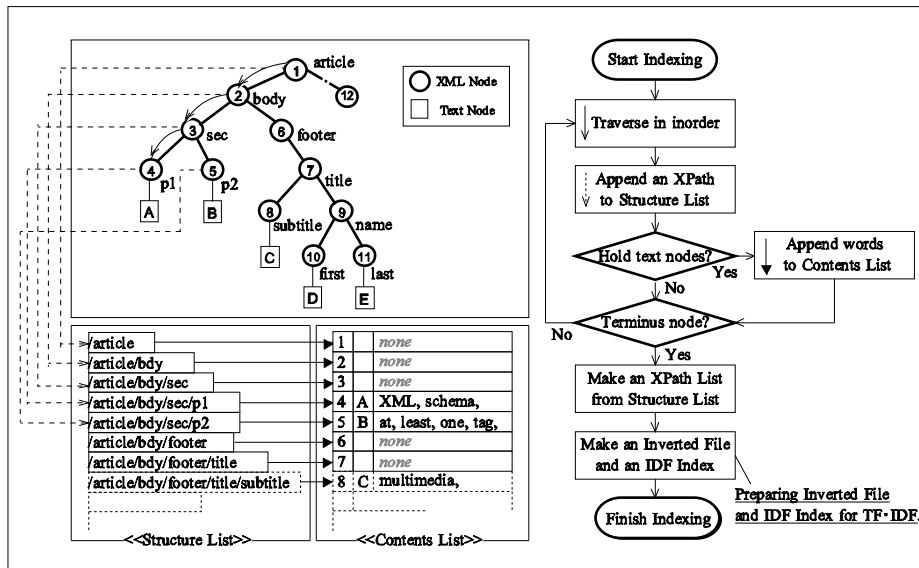


Fig. 2. Fragment indexing algorithm

Article Index. For XML articles, the unit is a whole XML document containing all the terms appeared at any nesting level within the <article> tag. This method is commonly used by a traditional inverted file for document retrieval.

Element Index. For XML elements, the unit is some XML elements included <article> tag. For each element containing only text located directly below. Hence every units are no overlap and flat. In this particular case, all units need not only identification but also addressing of XML elements (Figure 2).

XPath List. XPath is a language for addressing parts of an XML document[1]. In our scheme, each element in an XML document is identified by 2 parts, a relative path of XML document as an article and an absolute path of an element in XML document. Though our system does not support XML tag attributes.

The relative path identifies documents in the collection. And the absolute XPath expression identifies XML elements within the document, indicates the position of the element relative to the root element; for instance:

File path : C:/INEX/ex/2001/x0321.xml
Absolute XPath : /article[1]/bdy[1]/sec[5]/p[3]

Fragmental Indexing. Figure 2 shows an algorithm of making a fragment inverted file and a coordinate XPath list. The inverted file composed of three members: a word, a node identifier, a frequency of the word in the node. While the XPath list must numerate all nodes of XML trees in inorder traversal. When we select the sequence to sort, we can expect that the retrieval time reduced. As for details, please refer to the next section.

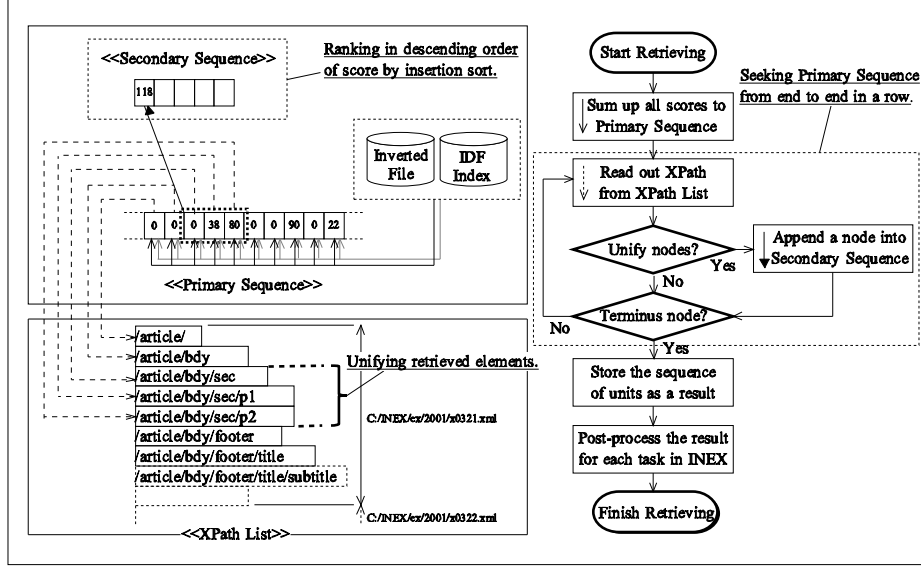


Fig. 3. Preferential unification algorithm

2.2 Retrieval Model

Figure 3 shows an overview of this retrieving framework. This system is based on the *tf-idf* (term frequency-inverse document frequency) weight.

Tf-idf Weighting. Let N be the total number of documents(as elements) in the system and n_i be the number of documents in which the index term k_i appears. Let $f_{i,j}$ be the raw frequency of term k_i in the document d_j . Then, the normalized frequency $tf_{i,j}$ of term k_i in document d_j and inverse document frequency idf_i for k_i are given by

$$tf_{i,j} = \frac{f_{i,j}}{\max_l f_{l,j}}, \quad idf_i = \log \frac{N}{n_i}$$

where the maximum is computed over all terms which are mentioned in the text of the document d_j . If the term k_i does not appear in the document d_j then $tf_{i,j} = 0$. The best known term-weighting schemes and a simple *similarity* use *tf-idf* weights, which are given by

$$sim(d_j, q) = \sum_{i=1}^t w_i \cdot tf_{i,j} \cdot idf_i$$

$$sim(D_k, q) = \sum_{d_j \in D_k} sim(d_j, q)$$

let w_i is the weight in a query $q = (w_1, w_2, \dots, w_t)$, where $w_i \geq 0$ and t is the total number of index terms in the system. Additionally, to consolidate fragmented index, let D_k is a set of fragmented elements, which will become a meaningful unit to the users by preferential unification.

Table 1. THOROUGH TASK.

Run ID	Description	MAep	Rank
VSM_1	-	0.0031	90/106
VSM_2	normalized TF	0.0047	89/106
VSM_3	unit=10	0.0064	79/106

* ep-gr(Quantization:gen,Overlap=o) shows a case of results.

Table 3. BEST IN CONTEXT TASK.

Run ID	Description	At A=1.0	Rank
VSM_10	unit=10	0.2376	50/77
VSM_11	unit=50	0.2912	33/77
VSM_12	unit=100	0.3074	28/77

* BEPD shows a case of results.

Table 2. FOCUSED TASK.

Run ID	Description	nxCG@5	Rank
VSM_4	unit=10	0.1945	74/85
VSM_5	unit=50	0.1672	77/85
VSM_6	unit=100	0.2305	68/85

* nxCG(Quantization:gen,Overlap=on) shows a case of results.

Table 4. Processing time.

	Searching time [s]
Article index	2.01
Element index	66.2

* Searching time is the average time of retrieving par query.

3 Experimental Results

The system was only designed for content-only conditions (CO queries). Then each post-processing dealt with differences between all four sub-tasks within the ad-hoc XML retrieval task.

3.1 THOROUGH TASK

THOROUGH TASK asks systems to estimate the relevance of elements in the collection. Hence the system produced 2 runs with the Article Index and 1 run in the Element Index with unit¹. THOROUGH TASK: are no further restrictions. Overlap is even permitted. Then, the system has no use for post-processing.

3.2 FOCUSED TASK

FOCUSED TASK asks systems to return a ranked list of elements to the user. Hence the system produced 3 runs in the Element Index. FOCUSED TASK: for the same topic, results may not be overlapping. That is, overlap is not permitted in the submitted run. Then, the system can apply the post-processing to reduce overlapping elements.

3.3 BEST IN CONTEXT TASK

BEST IN CONTEXT TASK asks systems to return articles with one best entry point to the user. Hence the system produced 3 runs by retrieving in the Element Index. BEST IN CONTEXT TASK: only single result per article is allowed. Then, the system can adopt the post-processing to filter that only one single element per article allowed.

¹ The unit in description which is one unit, consists of the number of retrieved elements in Element Index, as upper limit.

3.4 Time and Size

As Element Index, the results showed the searching time was markedly increased as compared with Article Index, as shown in Table 4. Also, the sizes of inverted files as *Article* and *Element* are 936 [MB] and 2,100 [MB] excluding XPath list: 1,749 [MB], respectively.

4 Conclusions

We proposed a method of fragmental indexing and an unification algorithm of retrieved elements. Even the system was not trained at all, the evaluation results showed more acceptable result from the unification approach than from the standard approach. Although, the system have one research issue, that is, the preferential unification algorithm takes longer than no unification. The reason is that the time of seeking in XPath list expends time. Then, further research has the improvement in seeking speed for XPath, before the accuracy improvement.

References

1. XML Path Language (XPath) Version 1.0. <http://www.w3.org/TR/xpath>
2. Ricardo Baeza-Yates, Berthier Ribeiro-Neto: Modern Information Retrieval (Acm Press Series). Addison-Wesley. (1999) 1–69, 141–162
3. Salton G., Wong A., Yang C. S.: A vector space model for automatic indexing. *Communications of the ACM*. **18** (1975) 613–620
4. D.Evans, R.Lererts: Design and Evaluation of the CLARIT-TREC-2 system. In *D.K.Harman editor, Proceedings of the Second Text REtrieval Conference (TREC-2)*. NIST Special Publication. (1994) 500–548
5. Andrew Trotman, Shlomo Geva: Passage Retrieval and XML-Retrieval Tasks. In *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology*. (2006) 43–50
6. Hiroki Tanioka, Kenichi Yamamoto, Takashi Nakagawa: A Distributed Retrieval System for NTCIR-5 WEB Task. In *The 5th NTCIR Workshop Meeting, 2005*
7. Hiroki Tanioka, Kenichi Yamamoto: A Distributed Retrieval System for NTCIR-5 Patent Retrieval Task. In *The 5th NTCIR Workshop Meeting, 2005*
8. Torsten Grabs, Hns-Jorg Schek: Flexible Information Retrieval on XML Documents. In *Intelligent Search on XML Data. Applications, Languages, Models, Implementations, and Benchmarks*. Springer Verlag, Berlin. **2818** (2003) 95–106
9. G. Kazai and N. Gvert and M. Lalmas, and N. Fuhr: The INEX evaluation initiative. In *Intelligent Search on XML Data. Applications, Languages, Models, Implementations, and Benchmarks*. Springer Verlag, Berlin. **2818** (2003) 279–293
10. Kenji Hatano, Hiroko Kinutani, Masahiro Watanabe, Masatoshi Yoshikawa, and Shunsuke Uemura: Determining the Unit of Retrieval Results for XML Documents. In *INEX Workshop*. (2002) 57–64
11. Borkur Sigurbjornsson, Jaap Kamps, Maarten de Rijke: An element-based approach to XML retrieval. In *INEX 2003 Workshop Proceedings*. (2003) 19–26
12. Shlomo Geva, Murray Leo-Spork: XPath Inverted File for Information Retrieval. In *INEX 2003 Workshop Proceedings*. (2003) 110–117
13. Wayne Kelly, Shlomo Geva, Tony Sahama, Wengkai Loke: Distributed XML Information Retrieval. In *INEX 2003 Workshop Proceedings*. (2003) 126–133
14. Vojkan Mihajlovic, Georgina Ramirez, Thijs Westerveld, Djoerd Hiemstra, Henk Ernst Blok, and Arjen P. de Vries: TIJAH Scratches INEX 2005: Vague Element Selection, Image Search, Overlap, and Relevance Feedback. In *INEX*. (2005) 72–87

TopX – AdHoc and Feedback Tasks

Martin Theobald, Andreas Broschart, Ralf Schenkel, Silvana Solomon, and
Gerhard Weikum

Max-Planck-Institut für Informatik
Saarbrücken, Germany
<http://www.mpi-inf.mpg.de/departments/d5/>
{mtb,abrosch,schenkel,solomon,weikum}@mpi-inf.mpg.de

Abstract. This paper describes the setup and results of our contributions to the INEX 2006 AdHoc and Feedback tasks.

1 System Overview

TopX [10, 11] aims to bridge the fields of database systems (DB) and information retrieval (IR). From a DB viewpoint, it provides an efficient algorithmic basis for top- k query processing over multidimensional datasets, ranging from structured data such as product catalogs (e.g., bookstores, real estate, movies, etc.) to unstructured text documents (with keywords or stemmed terms defining the feature space) and semistructured XML data in between. From an IR viewpoint, TopX provides ranked retrieval based on a relevance scoring function, with support for flexible combinations of mandatory and optional conditions as well as text predicates such as phrases, negations, etc. TopX combines these two aspects into a unified framework and software system, with emphasis on XML ranked retrieval.

Figure 1 depicts the main components of the TopX system. It supports three kinds of front-ends: as a servlet with an HTML end-user interface (that was used for the topic development of INEX 2006), as a Web Service with a SOAP interface (that was used by the Interactive track), and as a Java API (that was used to generate our runs). TopX currently uses Oracle10g as a storage system, but the JDBC interface would easily allow other relational backends, too.

The *Indexer* parses and analyzes the document collection and builds the index structures for efficient lookups of tags, content terms, phrases, structural patterns, etc. An *Ontology* component manages optional ontologies with various kinds of semantic relationships among concepts and statistical weighting of relationship strengths; we used WordNet [2] for some of our runs.

At query run-time, the *Core Query Processor* decomposes queries and invokes the top- k algorithms. It maintains intermediate top- k results and candidate items in a priority queue, and it schedules accesses on the precomputed index lists in a multi-threaded architecture. Several advanced components provide means for run-time acceleration:

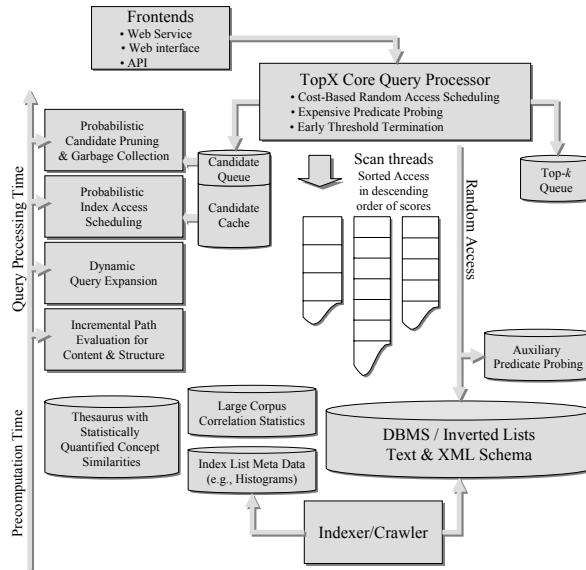


Fig. 1. TopX architecture.

- The *Probabilistic Candidate Pruning* component [12] allows TopX to drop candidates that are unlikely to qualify for the top- k results at an early stage, with a controllable loss and probabilistic result guarantees.
- The *Index Access Scheduler* [1] provides a suite of scheduling strategies for sorted and random accesses to index entries.
- The *Incremental Path Evaluation* uses additional cost models to decide when to evaluate structural conditions like XML path conditions, based on specialized indexes for XML structure.
- The *Dynamic Query Expansion* component [9] maps the query keywords and/or tags to concepts in the available ontology and incrementally generates query expansion candidates.

2 Data Model and Scoring

We refer the reader to [11] for a thorough discussion of the scoring model. This section shortly reviews important concepts.

2.1 Data Model

We consider a simplified XML data model, where idref/XLink/XPointer links are disregarded. Thus every document forms a tree of nodes, each with a *tag* and a related *content*. We treat attributes nodes as children of the corresponding

element node. The content of a node is either a text string or it is empty; typically (but not necessarily) non-leaf nodes have empty content. With each node, we associate its *full-content* which is defined as the concatenation of the text contents of all the node’s descendants in document order.

2.2 Content Scores

For content scores we make use of element-specific statistics that view the full-content of each element as a bag of words:

- 1) the *full-content term frequency*, $ftf(t, n)$, of term t in node n , which is the number of occurrences of t in the full-content of n ;
- 2) the *tag frequency*, N_A , of tag A , which is the number of nodes with tag A in the entire corpus;
- 3) the *element frequency*, $ef_A(t)$, of term t with regard to tag A , which is the number of nodes with tag A that contain t in their full-contents in the entire corpus.

The score of an element e with tag A with respect to a content condition of the form $*[\text{about}(\cdot, \tau)]$ is then computed by the following BM25-inspired formula:

$$\text{score}(e, *[\text{about}(\cdot, \tau)]) = \frac{(k_1 + 1) ftf(t, e)}{K + ftf(t, n)} \cdot \log \left(\frac{N_A - ef_A(t) + 0.5}{ef_A(t) + 0.5} \right) \quad (1)$$

$$\text{with } K = k_1 \left((1 - b) + b \frac{\sum_{s \in \text{full content of } e} ftf(s, e)}{\text{avg}\{\sum_{s'} ftf(s', e') \mid e' \text{ with tag } A\}} \right)$$

For a query content condition with multiple terms, the score of an element satisfying the tag constraint is computed as the sum of the element’s content scores for the corresponding content conditions, i.e.:

$$\text{score}(e, *[\text{about}(\cdot, t_1 \dots t_m)]) = \sum_{i=1}^m \text{score}(e, *[\text{about}(\cdot, t_i)]) \quad (2)$$

TopX provides the option to evaluate queries either in conjunctive mode or in “andish” mode. In the first case, all terms (and, for content-and-structure queries, all structural conditions) must be met by a result candidate, but still different matches yield different scores. In the second case, a node is already considered a match if it satisfy at least one content condition.

Orthogonally to this, TopX can be configured to return two different granularities as results: in *document mode*, TopX returns the best documents for a query, whereas in *element mode*, the best target elements are returned, which may include several elements from the same document.

2.3 Structural Scores

Given a query with structural and content conditions, we transitively expand all structural query dependencies. For example, in the query `//A//B//C[about(. , t)]` an element with tag `C` has to be a descendant of both `A` and `B` elements. Branching path expressions can be expressed analogously. This process yields a *directed acyclic graph* (DAG) with tag-term conditions as leaves, tag conditions as inner nodes, and all transitively expanded descendant relations as edges.

Our structural scoring model essentially counts the number of navigational (i.e., tag-only) conditions that are satisfied by a result candidate and assigns a small and constant score mass c for every condition that is matched. This structural score mass is combined with the content scores. In our setup we have set $c = 1$, whereas content scores are normalized to $[0, 1]$, i.e., we emphasize the structural parts.

3 AdHoc Track Results

There were two major changes in this year's AdHoc Track: The queries were run against the Wikipedia collection instead of the old IEEE collection, and there was only a single dimension of relevance (i.e., specificity) instead of both exhaustivity and specificity. As a consequence of this, smaller elements should be favored over larger elements (e.g., complete articles) at least for the Thorough subtask. Our scoring functions do not take this into account as they are still tuned towards the old twodimensional relevance (with exhaustivity and specificity).

For each subtask, we submitted at least the following four types of runs:

- `CO_{subtask}_baseline`: a CO run that considered the terms in the title of a topic without phrases and negations, limiting tags of results to `article`, `section`, and `p`.
- `CO_{subtask}_exp`: a CO run that considered terms as well as phrases and negations (so-called *expensive predicates*), again limiting tags of results to `article`, `section`, and `p`.
- `CAS_{subtask}_baseline`: a CAS run that considered the castitle of a topic if it was available, and the title otherwise. The target tag was evaluated strictly, whereas support conditions were optional; phrases and negations were ignored.
- `CAS_{subtask}_exp`: a CAS run that additionally considered phrases and negations.

3.1 Thorough Task

We submitted six runs to the Thorough task. In addition to our four standard runs, we submitted

- `TOPX_CO_Thorough_all`: a CO run that allowed all tags in the collection instead of limiting the tags to `article`, `section`, and `p`

- `TOPX_CAS_Thorough_ex_incr`: a CAS run that included expanding terms using WordNet

Table 1 shows the results for our runs. It turns out that all CO runs outperform the CAS runs that suffer from the strict evaluation of the target tag. Among the CO runs, the run that allows all result tags is best; this is not surprising as the other runs exclude many relevant results that have the ‘wrong’ tag. We see a slight advantage for runs that include phrases and negations, and a slight disadvantage for the run that expanded terms with WordNet. Overall, the performance of TopX is good (with a peak rank of 20), taking into account the limited amount of tuning that we did. Being a top- k engine, we expect that TopX would, like last year, perform even better for early cutoff points; however, they were unfortunately not measured this year.

run	rank	MAep
TOPX_CO_Thorough_all	20	0.0253
TOPX_CO_Thorough_ex	26	0.0190
TOPX_CO_Thorough_baseline	32	0.0178
TOPX_CAS_Thorough_ex	61	0.0103
TOPX_CAS_Thorough_baseline	62	0.0101
TOPX_CAS_Thorough_ex_incr	75	0.0081

Table 1. Results for the Thorough Task

3.2 Focused Task

Our runs for the focused task were produced by postprocessing our AdHoc runs to remove any overlap. For each such AdHoc run, we kept an element e if there was no other element e' from the same document in the run that had a higher score than e and had a path that overlapped with e 's path. This simple, syntactic postprocessing yielded good results (shown in Table 2). Especially for the early cutoff points, TopX performed extremely well with peak ranks 3 and 4. Interestingly, the CO run that considered phrases and negation did slightly better than its counterpart without expensive predicates.

3.3 BestInContext Task

To produce the runs for the BestInContext task, we ran TopX in document mode. This yielded a list of documents ordered by the highest score of any element within the document, together with a list of elements and their scores for each document. To compute the best entry point for a document, we simply selected the element with highest score from each document and ordered them by score. The results (Tables 3 and 4) show that this gave good results, with a peak rank of 1.

run	nxCG [5]	nxCG [10]	nxCG [25]	nxCG [50]
TOPX_CO_Focused_ex	0.3769 (3)	0.3154 (4)	0.2431 (10)	0.1916 (14)
TOPX_CO_Focused_baseline	0.3723 (4)	0.3051 (10)	0.2432 (9)	0.1913 (15)
TOPX_CAS_Focused_baseline	0.3397 (16)	0.2792 (21)	0.2017 (31)	0.1524 (39)
TOPX_CAS_Focused_ex	0.3339 (20)	0.2790 (22)	0.1985 (33)	0.1501 (41)
TOPX_CAS_Focused_ex_incr	0.2909 (40)	0.2341 (50)	0.1640 (59)	0.1232 (59)

Table 2. Results for the Focused Task with the nxCG metric at different cutoffs (ranks are in parentheses), with overlap=on

run	A=0.1	A=1	A=10	A=100
TOPX-CO-BestInContext-baseline	0.1280 (22)	0.2237 (11)	0.3685 (5)	0.5715 (5)
TOPX-CO-BestInContext-exp	0.1189 (28)	0.2074 (20)	0.3451 (11)	0.5384 (11)
TOPX-CAS-BestInContext-baseline	0.0718 (54)	0.1361 (53)	0.2272 (53)	0.3780 (53)
TOPX-CAS-BestInContext-exp	0.0653 (57)	0.1254 (56)	0.2131 (57)	0.3594 (54)

Table 3. Results for the BestInContext Task with the BEPD metric (ranks are in parentheses)

run	A=0.1	A=1	A=10	A=100
TOPX-CO-BestInContext-exp	0.0260 (13)	0.0604 (7)	0.1241 (3)	0.2081 (1)
TOPX-CO-BestInContext-baseline	0.0258 (17)	0.0607 (5)	0.1231 (4)	0.2050 (3)
TOPX-CAS-BestInContext-exp	0.0163 (42)	0.0394 (38)	0.0764 (26)	0.1422 (29)
TOPX-CAS-BestInContext-baseline	0.0160 (44)	0.0388 (40)	0.0748 (33)	0.1380 (32)

Table 4. Results for the BestInContext Task with the EPRUM-BEP-Exh-BEPDistance metric (ranks are in parentheses)

4 Structural Query Expansion

Our feedback framework aims at generating a content-and-structure query from a keyword query, exploiting relevance feedback provided by a user for some results of the keyword query. This section gives a very brief summary of our approach; for a more detailed and formal description, see [8].

We consider the following classes of candidates for query expansion from an element with known relevance:

- all terms of the element’s content (C candidates),
- all tag-term pairs of descendants of the element in its document (D candidates),
- all tag-term pairs of ancestors of the element in its document (A candidates),
- and
- all tag-term pairs of descendants of ancestors of the element in its document, together with the ancestor’s tag (AD candidates).

To weight the different candidates c , we apply an extension of the well-known Robertson-Sparck-Jones weight [5] to element-level retrieval in XML, applying it to elements instead of documents:

$$w_{RSJ}(c) = \log \frac{r_c + 0.5}{R - r_c + 0.5} + \log \frac{E - ef_c - R + r_c + 0.5}{ef_c - r_c + 0.5}$$

Here, for a candidate c , r_c denotes the number of relevant elements which contain the candidate c in their candidate set, R denotes the number of relevant elements, E the number of elements in the collection, and ef_c the element frequency of the candidate.

To select the candidates to expand the query, we use the Robertson Selection Values (RSV) proposed by Robertson [4]. For a candidate c , its RSV has the form $RSV(c) = w_{RSJ}(c) \cdot (p - q)$, where $p = r_c/R$ is the estimated probability of the candidate occurring in a relevant element’s candidate set and q is the probability that it occurs in a nonrelevant element’s set. We ignore candidates that occur only within the documents of elements with known relevance as they have no potential to generate more relevant results outside these documents, and we ignore candidates that contain a query term. We choose the top b of the remaining candidates for query expansion (b is a configurable parameter).

Using these top- b candidates, we generate a content-and-structure query from the original keyword query, where each additional constraint is weighted with the normalized RSJ weight of its corresponding candidate (see [8]). The expansion itself is actually rather straightforward; the generated query has the following general structure:

```
//ancestor-tag[A+AD constraints]//*[keywords+C+D constraints]
```

As an example, if the original query was "XML" and we selected

- the A candidate `//ancestor::article[about(., IR)]`,

- the AD candidate `//ancestor::article[about(../bib, index)]`,
- the D candidate `//descendant:p[about(., index)]`, and
- the C candidate `about(., database)`,

the expanded query (omitting the weights) would be

```
//article[about(., IR) and about(../bib, index)]/*[about(., XML)
and about(., database) and about(../p, index)].
```

5 Feedback Task Results

INEX 2006 introduced a new relevance measure, specificity, that replaced the two dimensions of relevance, exhaustivity and specificity, used before. This happened mainly for two reasons: First, to make assessments easier, and second, because correlation analyses had shown that comparing systems in the AdHoc track yields a result when using specificity only that is sufficiently similar to the result with specificity and exhaustivity.

However, this new measure does not reflect the relevance of an element from a user’s point of view. It is unlikely that a user would greatly appreciate seeing a single `collectionlink` element or, even worse, an isolated `xlink:href` attribute in a result list. It is therefore questionable if specificity alone can be used for automated feedback.

5.1 Evaluation of Feedback Runs

We discussed different evaluation modes in our paper at last INEX [7]. There is still no common agreement on one mode that should give the ‘best’ results. We shortly review the modes here and introduce a new mode, *resColl-path*.

- Simply comparing the results of the baseline run with the results generated from feedback (we denote this as *plain*) is commonly considered as illegal, as feedback includes the advantage of knowing some relevant results and hence can yield a better performance.
- With rank freezing, the rank of results with known relevance is frozen, thus assessing only the effect of reranking the results with unknown relevance. We label this approach *freezeTop* as usually the top-*k* results are used for feedback and hence frozen. This has been the standard evaluation mode for the INEX relevance feedback task.
- With the residual collection technique, all XML elements with known relevance must be removed from the collection before evaluation of the results with feedback takes place. Depending on which elements are considered as having known relevance, a variety of different evaluation techniques results:
 - *resColl-result*: only the elements for which feedback is given are removed from the collection,

- *resColl-desc*: the elements for which feedback is given and all their descendants are removed from the collection,
- *resColl-anc*: the elements for which feedback is given and all their ancestors are removed from the collection,
- *resColl-doc*: for each element for which feedback is given, the whole document is removed from the collection, and
- *resColl-path*: for each element for which feedback is given, the element itself, its ancestors and its descendants are removed from the collection.

The most natural evaluation mode is *resColl-path*, as it removes all elements for which the feedback algorithm has some knowledge about their potential relevance. We evaluate our approach with all seven evaluation techniques in the following section and try to find out if there are any differences.

5.2 Preliminary Results

We measured only MAP and precision at different cutoffs (the other measurements will be part of the official evaluation). Due to time constraints, we consider only the first 49 topics that have assessments (topics 289-339, excluding topics 299 and 307), runs with 100 elements, and feedback for the top-20 results of our baseline run `TOPX_CO_Thorough_all` with the *Generalised* quantization. Our experiments use the top-10 candidates for feedback, where different classes of candidates are considered. We tested the significance of our results with the t-test and the Wilcoxon signed-rank test [6].

Due to time constraints, we could only consider a limited number of combinations of candidate classes, see Table 5 for the results. Unlike our results from last year with the IEEE collection, content-only feedback outperformed all other combinations, and A and AD candidates alone often could not improve result quality significantly. At this time, we do not have a well-founded explanation for this behaviour. However, there are some major differences of the new Wikipedia collection to the old IEEE collection:

- Wikipedia documents do not have a clear structure with front and back matter. For the old IEEE collection, especially A and AD candidates could exploit things like authors of a document, authors of a cited document, or journal names.
- The one-dimensional relevance measure penalizes large elements towards the root of a document. This is a natural disadvantage for using D candidates that tend to add results near the root element.
- Our old experiments used only the *Strict* quantization where the best elements typically were sections or paragraphs. With the new relevance measure and quantization, the best elements and attributes are small (like `collectionlink` or `xlink:href`) which do not contribute many candidates to the candidate pool. We will rerun the experiments with the *Strict* quantization for feedback to see if this assumption is true.

evaluation	baseline	C	D	C+D	A	AD	A+AD
plain	0.0188	<i>0.0364</i>	<i>0.0328</i>	<i>0.0344</i>	0.0187	0.0164	<i>0.0228</i>
freezeTop	0.0188	<i>0.0284</i>	<i>0.0248</i>	<i>0.0256</i>	0.0189	0.0181	<i>0.0216</i>
resColl-result	0.0108	<i>0.0264</i>	<i>0.0212</i>	<i>0.0218</i>	0.0106	0.0106	0.0171
resColl-anc	0.0101	<i>0.0246</i>	<i>0.0194</i>	<i>0.0201</i>	0.0102	0.0102	0.0169
resColl-desc	0.0049	<i>0.0087</i>	0.0078	<i>0.0081</i>	0.0048	0.0045	0.0056
resColl-doc	0.0041	<i>0.0077</i>	0.0067	<i>0.0072</i>	0.0040	0.0038	0.0048
resColl-path	0.0046	<i>0.0085</i>	0.0072	<i>0.0079</i>	0.0044	0.0043	0.0056

Table 5. MAP values for different configurations and different evaluation modes. Runs shown in **bold** are significantly better than the baseline under the WSR test ($p < 0.01$), runs shown in *italics* are significantly better than the baseline under the t-test ($p < 0.01$).

Our future work in this area will focus on using other measures of relevance like the one proposed for HiXEval [3]. This may additionally pave the way for feedback that exploits the granularity of results (e.g., to derive tags for a keyword-only query). We will additionally examine how to choose a threshold for the element frequency of candidates that are considered, and which other candidate classes could be used.

References

1. H. Bast, D. Majumdar, M. Theobald, R. Schenkel, and G. Weikum. IO-Top- k : Index-optimized top- k query processing. In *VLDB*, pages 475–486, 2006.
2. C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
3. J. Pehcevski and J. A. Thom. Hixeval: Highlighting xml retrieval evaluation. In *INEX*, pages 43–57, 2005.
4. S. Robertson. On term selection for query expansion. *Journal of Documentation*, 46:359–364, Dec. 1990.
5. S. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society of Information Science*, 27:129–146, May–June 1976.
6. J. Savoy. Statistical inference in retrieval effectiveness evaluation. *Inf. Process. Manage.*, 33(4):495–512, 1997.
7. R. Schenkel and M. Theobald. Relevance feedback for structural query expansion. In *INEX*, pages 344–357, 2005.
8. R. Schenkel and M. Theobald. Structural feedback for keyword-based xml retrieval. In *ECIR*, pages 326–337, 2006.
9. M. Theobald, R. Schenkel, and G. Weikum. Efficient and self-tuning incremental query expansion for top- k query processing. In *SIGIR*, pages 242–249, 2005.
10. M. Theobald, R. Schenkel, and G. Weikum. An efficient and versatile query engine for TopX search. In *VLDB*, pages 625–636, 2005.
11. M. Theobald, R. Schenkel, and G. Weikum. TopX & XXL @ INEX 2005. In *INEX*, pages 282–295, 2005.
12. M. Theobald, G. Weikum, and R. Schenkel. Top- k query evaluation with probabilistic guarantees. In *VLDB*, pages 648–659, 2004.

Supervised and Semi-supervised Machine Learning Ranking

Jean-Noël Vittaut and Patrick Gallinari

Laboratoire d'Informatique de Paris 6
8, rue du Capitaine Scott, F-75015 Paris, France
{vittaut, gallinari}@poleia.lip6.fr

Abstract. (draft) We present a Machine Learning based ranking model which can automatically learn its parameters using a training set of labeled and unlabeled examples composed of queries and relevance judgments on a subset of the document elements. Our model improves the performance of a baseline Information Retrieval system by optimizing a ranking loss criterion and combining scores computed from doxels and from their local structural context. We analyze the performance of our supervised and semi-supervised algorithms on CO-Focussed and CO-Thorough tasks using a baseline model which is an adaptation of Okapi to Structured Information Retrieval.

1 Introduction

Different studies and developments have been recently carried out on ranking algorithms in the machine learning community. In the field of textual documents, they have been successfully used to combine features or preferences relations in tasks such as meta search [1] [2] [3], passage classification, automatic summarization [4] and recently for the combination of different sources of evidence in Information Retrieval (IR) [5]. One of the challenges of this paradigm is to reduce the complexity of the algorithms which is in the general case quadratic in the number of samples. This is why most real data applications of ranking are based on two-classes problems. Nevertheless, some linear methods has been proposed [3] [4] and under some conditions, fast rates of convergence are achieved with this class of methods [6].

Ranking algorithms work by combining features which characterize the data elements to be ranked. In our case, these features will depend on the doxel itself and on its structural context. Ranking algorithms will learn to combine these different features in an optimal way according to a specific loss function using a set of examples. It is hoped that ranking algorithms may help to improve the performance of existing techniques.

The semi-supervised paradigm has emerged already been used in classification [?]. It has showed its efficiency on a class of tasks where there are only a few labeled samples available. This is the case in SIR and more generally in IR:

there are very few labeled databases, and the labeling of relevant units for particular queries is time consuming. There is also tasks where only a few samples are labeled such as relevance-feedback.

The paper is organized as follows, in section 2 we present the ranking model, in section 3 we show how we adapted it to CO-Focussed and CO-Thorough tasks. In section 4 we comment the results reached by our model and compare it to a baseline Okapi method adapted for Structured Information Retrieval (SIR).

2 Ranking model

We present in this section a general model of ranking which can be adapted to IR or SIR. The idea of the ranking algorithms proposed in the machine learning community is to learn a total order on a set \mathcal{X} , which allows to compare any element pair in this set. Given this total order, we are able to order any subset of \mathcal{X} in a ranking list. For instance in IR, \mathcal{X} can be the set of couples (document, query), and the total order is the natural order on the document scores.

As for any machine learning technique, one needs a training set of labeled examples in order to learn how to rank. This training set will consist in ordered pairs of examples. This will provide a partial order on the elements of \mathcal{X} . The ranking algorithm will use this information to learn a total order on the elements of \mathcal{X} and after that will allow to rank new elements. For plain IR, the partial ordering may be provided by human assessments on different documents for a given query.

2.1 Notations

Let \mathcal{X} be a set of elements with a partial order \prec defined on it. This means that some of the element pairs in \mathcal{X} may be compared according to the \prec relation. For Structured Information retrieval \mathcal{X} will be the set of couples (doxel, query) for all doxels and queries in the document collection. This set is partially ordered according to the existing relevance judgments for each query.

2.2 Ranking

Let f be a function from \mathcal{X} to the set of real numbers. We can associate a total order \prec_T to f such that:

$$x \prec_T x' \Leftrightarrow f(x) < f(x') . \tag{1}$$

Clearly, learning the f function is the same as learning the total order. In the following, we will extend the partial order \prec to a total order \prec_T , so we will use the same notation for both relations.

An element of \mathcal{X} will be represented by a real vector of features:

$$x = (x_1, x_2, \dots, x_d).$$

In our case, the features will be local scores computed on different contextual elements of a doxel. In the following, f will be a linear combination of x 's features:

$$f_\omega(x) = \sum_{j=1}^d \omega_j x_j \quad (2)$$

where $\omega = (\omega_1, \omega_2, \dots, \omega_d)$ are the parameters of the combination to be learned.

Ranking loss. f_ω is said to respect $x \prec x'$ if $f_\omega(x) < f_\omega(x')$. In this case, couple (x, x') is said to be well ordered by f_ω . The ranking loss [3] measures how much f_ω respects \prec .

By definition, the ranking loss measures the number of mis-ordered couples in \mathcal{X}^2 :

$$R(\mathcal{X}, \omega) = \sum_{\substack{(x, x') \in \mathcal{X}^2 \\ x \prec x'}} \chi(x, x') \quad (3)$$

where $\chi(x, x') = 1$ if $f_\omega(x) > f_\omega(x')$ and 0 otherwise.

Ranking aims at learning ω for minimizing (3).

Exponential loss. In practice, this expression is not very useful since χ is not differentiable, ranking algorithms use to optimize another loss criterion called the exponential loss:

$$R_e(\mathcal{X}, \omega) = \sum_{\substack{(x, x') \in \mathcal{X}^2 \\ x \prec x'}} e^{f_\omega(x) - f_\omega(x')}. \quad (4)$$

It is straightforward that $R(\mathcal{X}, \omega) \leq R_e(\mathcal{X}, \omega)$. (4) is differentiable and convex, and then can be minimized using standard optimization techniques. Minimizing (4) will allow to minimize $R(\mathcal{X}, \omega)$.

We can compute a gradient descent. The components of the gradient of R_e are:

$$\frac{\partial R_e}{\partial \omega_k}(\mathcal{X}, \omega) = \sum_{\substack{(x, x') \in \mathcal{X}^2 \\ x \prec x'}} (x_k - x'_k) e^{f_\omega(x) - f_\omega(x')}. \quad (5)$$

With no more hypothesis, the computation of (5) is in $O(|\mathcal{X}|^2)$.

3 Application to CO tasks

3.1 Definitions

Let denote \mathcal{D} is the set of doxels for all the documents in the collection and \mathcal{Q} the set of CO queries. $\mathcal{X} = \mathcal{Q} \times \mathcal{D}$ is the set of elements we want to order.

We suppose that there exists a partial order \prec on $\mathcal{X} = \mathcal{Q} \times \mathcal{D}$, this partial order will reflect for some queries, the evidence we have about preferences between

doxels provided via manual assessments. Note that these relevance assessments are only needed for a few queries and doxels in the collection. We consider here the task which consists in producing a ranked list of doxels which answer the query $q \in \mathcal{Q}$. For that, we will train the ranking model to learn a total strict order on \mathcal{X} .

3.2 Vector Representation

Each element $x \in \mathcal{X}$ is represented by a vector (x_1, x_2, \dots, x_d) where x_i represents some feature which could be useful to order elements of \mathcal{X} . Let denote \mathcal{L} the set of doxel types, which are defined according to the DTD of the document collection: article, abstract, sections, paragraphs, lists...

We used the following combination:

$$f_w(x) = \omega_1^l + \omega_2^l Okapi(x) + \omega_3^l Okapi(parent(x)) + \omega_4^l Okapi(document(x))$$

where l is the node type of x and *Okapi* is the SIR adapted Okapi model [7] described in [8]. This adaptation consists in using doxels rather than documents for computing the term frequencies, and using as normalization factor for each doxel, the mean size of the doxels with the same node type.

This combination take into account the information provided by the context of the doxel and the structural information given by the node type of the doxel.

This combination leads to the following vector representation:

$$x = \left((x_1^{l_1}, x_2^{l_1}, x_3^{l_1}, x_4^{l_1}), (x_1^{l_2}, x_2^{l_2}, x_3^{l_2}, x_4^{l_2}), \dots, (x_1^{l_{|\mathcal{L}|}}, x_2^{l_{|\mathcal{L}|}}, x_3^{l_{|\mathcal{L}|}}, x_4^{l_{|\mathcal{L}|}}) \right)$$

where $|\mathcal{L}|$ is the number of different doxel types in the collection.

In the above expression all vector components of the form $(x_1^{l_i}, x_2^{l_i}, x_3^{l_i}, x_4^{l_i})$ are equal to $(0, 0, 0, 0)$ except for one where l_i is the doxel type of x which is equal to $(1, Okapi(x), Okapi(parent(x)), Okapi(document(x)))$.

3.3 Reduction of complexity

In this section, we use some properties of SIR in order to decrease the complexity of the computation of (4) and (5).

Queries. Comparing elements from different queries has no sense. We can define a partition $\mathcal{X} = \bigcup_{q \in \mathcal{Q}} \mathcal{X}_q$, where

$$\mathcal{X}_q = \{x = (d, q') \in \mathcal{X} / q' = q\}$$

and we can rewrite (4):

$$R_e(\mathcal{X}, \omega) = \sum_{q \in \mathcal{Q}} \left\{ \sum_{\substack{(x, x') \in \mathcal{X}_q \times \mathcal{X}_q \\ x \prec x'}} e^{f_\omega(x)} e^{-f_\omega(x')} \right\}. \quad (6)$$

Assessments. For each subset \mathcal{X}_q , the preferences among doxels are expressed according to a several discrete dimensions. We have:

- an information of exhaustivity, which measures how much a doxel answers the totality of an information need (0 not exhaustive, ..., 3 fully exhaustive)
- an information of specificity, which measures how much a doxel answers only the information need (0 not specific, ..., 3 means fully specific)

There is no preference between doxels sharing the same value of exhaustivity and specificity.

An assessment is a couple (exhaustivity, specificity). Let denote \mathcal{A} the set of assessments and $A(x)$ the assessment of element x . We can define a partition $\mathcal{X}_q = \bigcup_{a \in \mathcal{A}} \mathcal{X}_q^a$, where

$$\mathcal{X}_q^a = \{x \in \mathcal{X}_q / A(x) = a\}.$$

We can rewrite (6):

$$R_e(\mathcal{X}, \omega) = \sum_{q \in \mathcal{Q}} \sum_{a \in \mathcal{A}} \left\{ \left(\sum_{x \in \mathcal{X}_q^a} e^{f_\omega(x)} \right) \left(\sum_{\substack{b \in \mathcal{A} \\ \mathcal{X}_q^b \prec \mathcal{X}_q^a}} \sum_{x \in \mathcal{X}_q^b} e^{-f_\omega(x)} \right) \right\}. \quad (7)$$

where $\mathcal{X}_q^b \prec \mathcal{X}_q^a$ means that the assessments of the elements of \mathcal{X}_q^a are better than those of \mathcal{X}_q^b . An possible order between assessments is represented in figure 1.

The complexity for computing this expression is $O(K \cdot |\mathcal{Q}| \cdot |\mathcal{X}|)$ whereas it is $O(|\mathcal{X}|^2)$ for (4) where K is the number of sets in the partition of \mathcal{X} . The worst case occurs when $K = \mathcal{X}$.

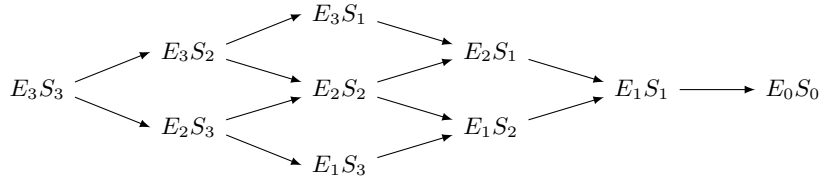


Fig. 1. Graph representing the order between elements for a given query, according to the two dimensional discrete scale of INEX. Doxels labeled E_3S_3 must be the highest ranked, and doxels labeled E_0S_0 the lowest ranked.

3.4 Gradient descent

Since (7) is convex, we can use a gradient descent technique to minimize it. The components of the gradient has the following form:

$$\begin{aligned} \frac{\partial R_e}{\partial \omega_k}(\mathcal{X}, \omega) = & \sum_{q \in \mathcal{Q}} \sum_{a \in \mathcal{A}} \left\{ \left(\sum_{x \in \mathcal{X}_q^a} x_k e^{f_\omega(x)} \right) \left(\sum_{\substack{b \in \mathcal{A} \\ \mathcal{X}_q^b \prec \mathcal{X}_q^a}} \sum_{x \in \mathcal{X}_q^b} e^{-f_\omega(x)} \right) \right. \\ & \left. + \left(\sum_{x \in \mathcal{X}_q^a} e^{f_\omega(x)} \right) \left(\sum_{\substack{b \in \mathcal{A} \\ \mathcal{X}_q^b \prec \mathcal{X}_q^a}} \sum_{x \in \mathcal{X}_q^b} -x_k e^{-f_\omega(x)} \right) \right\}. \end{aligned} \quad (8)$$

The complexity for computing the gradient is the same ($O(K \cdot |\mathcal{Q}| \cdot |\mathcal{X}|)$) as that of (7).

3.5 Incorporation of unlabeled samples

The natural way to incorporate an unlabeled sample y would be to compute all probabilities $P(y \prec x)$ for x in \mathcal{X} . But this method would be computationally costly because we could not use property 7 to reduce the complexity since y is not in a particular \mathcal{X}_k^i .

A better method is to affect y to only one \mathcal{X}_k^i , using a certain probability of belonging. We say that a sample belongs to the group with which it has the maximum probability of indifference:

$$P(y \in \mathcal{X}_k^i) = P(\{y\} \perp \mathcal{X}_k^i) = \prod_{x \in \mathcal{X}_k^i} P(y \perp x) = \prod_{x \in \mathcal{X}_k^i} P(y \prec x)P(x \prec y)$$

If we use the exponential ranking loss, we will choose the group \mathcal{X}_k^i which minimize the ranking loss:

$$\exp(s_\omega(y)) \sum_{x \in \mathcal{X}_k^i} \exp(-s_\omega(x)) + \exp(s_\omega(-y)) \sum_{x \in \mathcal{X}_k^i} \exp(s_\omega(x)).$$

Note that is not ordinal regression because there is some groups (\mathcal{X}_k 's) which are each other indifferent and if we could know a priori if an unlabeled sample belongs to one of this group. Even if our model produce a total order on \mathcal{X} , there is some comparisons without sens (for instance, we see in section 3.3 that comparing search results from different queries is none sense).

4 Experiments

4.1 Learning base

We used the Wikipedia collection with a small set of anotated queries (3) as a learning base. We will comment the results.

4.2 Filtering

In CO-Focussed task, overlapping doxels were not allowed. In order to suppress all overlapping elements from the lists computed by the ranking algorithm, we used a strategy which consists in removing all elements which are overlapping with an element ranked higher in the list.

As for Okapi model, we used the same strategy except that biggest doxels like articles or bdy’s were not allowed in the final ranking list to reach better performance.

4.3 Results

We comment the results obtained with the nxCG official metric with generalized quantization and overlap on and off which is more related to the ranking loss criterion and the different levels of assessment we have used in our model.

CO-Focussed. We can see that the ranking algorithms perform better than Okapi.

Table 1 and 2 show that the ranking models are always better than its baseline Okapi model, and that is quite good to retrieve the most informative doxels in the beginning of the list.

Table 1. Rank of Okapi and ranking models among all participant submissions using nxCG metric with generalised quantization and overlap on for CO-Focussed task

	@5	@10	@25	@50
Ranking	2	1	2	3
Ranking semi-supervised	14	12	12	8
Okapi	50	39	37	32

Table 2. Rank of Okapi and ranking models among all participant submissions using nxCG metric with generalised quantization and overlap off for CO-Focussed task

	@5	@10	@25	@50
Ranking	1	1	4	8
Ranking semi-supervised	1	7	9	11
Okapi	49	39	40	33

CO-Thorough. Table 3 shows that the ranking model is always better than its baseline Okapi model, and that is quite good to retrieve the most informative doxels in the beginning of the list. This can be explained by the expression of the ranking loss which penalize more a irrelevant doxel when it is located in the beginning of the list.

Table 3. Rank of Okapi and ranking models among all participant submissions using nxCG metric for CO-Thorough task

	rank score	
Ranking	18	0.0281
Ranking semi-supervised	18	0.0281
Okapi	27	0.0186

5 Conclusion

We have described our ranking model for CO tasks which relies on a combination of scores from the Okapi model and takes into account the document structure. This score combination is learned from a training set by a ranking algorithm.

For both tasks, the ranking algorithm has been able to increase by a large amount the performance of the baseline Okapi with a very small set of labeled examples. Ranking methods thus appear as a promising direction for improving SIR search engine performance. It remains to perform tests with additional features (for example the scores of additional IR systems).

References

1. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. In Jordan, M.I., Kearns, M.J., Solla, S.A., eds.: *Advances in Neural Information Processing Systems*. Volume 10., The MIT Press (1998)
2. Bartell, B.T., Cottrell, G.W., Belew, R.K.: Automatic combination of multiple ranked retrieval systems. In: *Research and Development in Information Retrieval*. (1994) 173–181
3. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. In: *Proceedings of ICML-98, 15th International Conference on Machine Learning*. (1998)
4. Amini, M.R., Usunier, N., Gallinari, P.: Automatic text summarization based on word-clusters and ranking algorithms. In: *ECIR*. (2005) 142–156
5. Craswell, N., Robertson, S., Zaragoza, H., Taylor, M.: Relevance weighting for query independent evidence. In: *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference*. (2005)
6. Cléménçon, S., Lugosi, G., Vayatis, N.: Ranking and scoring using empirical risk minimization. In Auer, P., Meir, R., eds.: *COLT*. Volume 3559 of *Lecture Notes in Computer Science*., Springer (2005) 1–15

7. Robertson, S.E., Walker, S., Hancock-Beaulieu, M., Gull, A., Lau, M.: Okapi at TREC. In: Text REtrieval Conference. (1992) 21–30
8. Vittaut, J.N., Piwowarski, B., Gallinari, P.: An algebra for structured queries in bayesian networks. In: Advances in XML Information Retrieval. Third Workshop of the INitiative for the Evaluation of XML Retrieval. (2004)

PF/Tijah at INEX 2006

Thijs Westerveld¹, Henning Rode², Roel van Os², Djoerd Hiemstra², Georgina Ramírez¹, Vojkan Mihajlović², and Arjen P. de Vries¹

¹ CWI, Amsterdam, The Netherlands

² University of Twente, Enschede, The Netherlands

Abstract. CWI and Utwente collaborated again for INEX. This year, we participated in the Ad Hoc and Multimedia tasks. For both tasks, we relied on the pf/tijah system we developed for flexible information retrieval from structured document collections[1]. Pf/tijah integrates NEXI based IR functionality with full XQuery support.

In the Ad Hoc track we focused on three aspects. First, we studied whether element retrieval could do better than article retrieval. We expected article or full document retrieval to be competitive in this collection with relatively short and specific articles. Second, we experimented with context weighting. Can we improve an element ranking based on the elements' context. We investigated if it helps to take the article context into account as well as whether using the article's title could leave to an improvement. Third, we looked at approaches to identify good entry-points in relevant articles for the AllInContext and BestInContext tasks. For the AllInContext task, we ranked articles by either the maximum or the average score of its top 1500 elements. All these elements are kept as good entry-points. For the BestInContext task, we ranked articles by the sum of the scores of all its elements found in the top1500; the first of these elements (in document order) was then submitted as the best entry-point. This means, for this task, the number of retrieved documents was limited by the number of different articles retrieved from in the top 1500 elements, and thus typically much lower than 1500.

For the MMfragments task of the multimedia track. we limited our system to return only fragments that contain at least one image that was part of the multimedia collection. Apart from that, we did not do any multimedia processing, and simply experimented with text based approaches. We extended the collection with the metadata from the MMimage collection; for each of the images in the MMfragments collection, we included the text from the corresponding image in the MMimages collection inside the <image> tag. For the MMimages task, we experimented with different textual query variants, including adding the metadata from example images to the text query.

The results have not been analysed in detail yet, and will be discussed in the final version of this paper.

References

1. D. Hiemstra, H. Rode, R. van Os, and J. Flokstra. Pftijah: text search in an XML databases system. In *Proceedings of the 2nd International Workshop on Open Source Information Retrieval (OSIR)*, 2006.

XSee: Structure Xposed

Roelof van Zwol¹ * and Wouter Weerkamp²

¹ Yahoo! Research, C/ Ocata 1, 08003 Barcelona, Spain
roelof@yahoo-inc.com

² Utrecht University, Department of Information and Computing Sciences,
Padualaan 14, 3508 TB Utrecht, The Netherlands
wouter@weerkamp.com

Abstract. XML retrieval is a discipline of information retrieval that focuses on the retrieval of specific document fragments that answer a user information need, while exploiting the structural information available in a document. The contribution of this article is twofold. It presents the effective and scalable retrieval model of the XSee search engine for XML documents, and shows that the model is stable over the different collections used for INEX. Furthermore, we discuss how based on the structural characteristics of the document collection the effectiveness of the retrieval model can be enhanced using different reranking methods.

1 Introduction

The Initiative for the evaluation of XML retrieval (INEX) is a yearly event that provides an international forum for the evaluation of XML retrieval strategies. These strategies aim to harness the enriched source of syntactic and semantic information that XML markup provides. Current work in XML IR focuses on exploiting the available structural information in documents to implement a more focused retrieval strategy and to return document components, the so-called XML elements - instead of complete documents - in response to a user query [1]. This article discusses the approach conducted by Utrecht University in their third year of participation.

Each year we choose a specific angle of interest for our research, besides our general objective to develop innovative retrieval strategies for XML retrieval. This year's contribution focuses specifically on the exploitation of the structural characteristics of the XML collections, to improve the performance on thorough and focused retrieval tasks.

The contribution in this article is therefore twofold. First we will present the underlying retrieval model that is used by our *XML Search* engine, nicknamed XSee. The retrieval model for XSee is derived from the GPX [2] and the B3-SDR model [3] for efficient use of structural clues.

* Halfway during this year's INEX initiative, Roelof van Zwol has left Utrecht University to join Yahoo! Research in Barcelona.

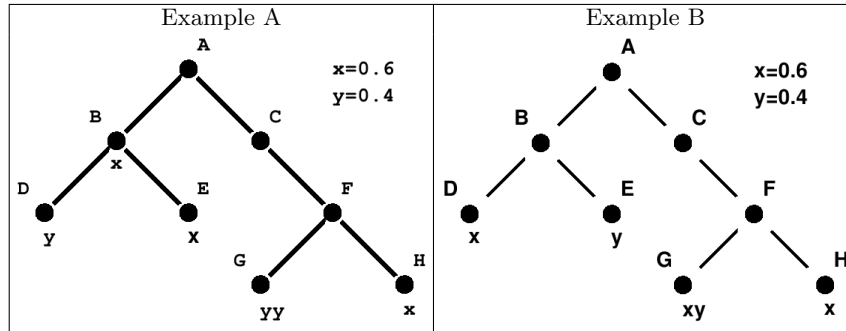


Fig. 1. XML tree of Example A and B.

1.1 Behavior of the XSee vs. GPX retrieval model

The main difference of the XSee model compared to the GPX model is that the relevance of a node in the XSee model is more directly influenced by the number of distinct query terms that are contained by that node. In the GPX model, the relevance of the leaf nodes is determined first, and based on those scores the relevance of the ascending nodes has to be determined iteratively by traversal of the tree to the root node. Although this give good results in retrieval performance, computing a ranking is rather expensive. Instead of computing the relevance scores from the leaves to the root, the XSee model determines at indexing time the contribution of each term to a node, such that at query time only the result for Equation 1 needs to be calculated.

To illustrate the effect of this modification, we use the two examples of Figure 1, and the derived ranking for a fictive query $[x\ y]$ of Table 1. The Figure shows an XML tree with 8 nodes, and the occurrence of the query terms $[x\ y]$ in the tree for examples A and B. To configure the GPX model [2] three parameters need to be defined: a single node decay, a multiple node decay, and a term factor, which are set to 0.5, 0.75, and 5 respectively. The XSee model needs two parameters to be set: a node decay ($nd=0.4$), and a term factor ($F_t = 5$) respectively³. For both examples we assume the following term weights: $x = 0.6$, and $y = 0.4$.

The data under Example A of Table 1 shows the results for this example XML tree. For each node, its position in the ranking and the computed relevance score is given on the XSee and GPX model. It shows that the weight of those nodes that combine new query terms (nodes B and F) have a higher position in the ranking for the XSee model than for the GPX model. Furthermore, the XSee model lowers the position of those nodes that have no direct descendants containing relevant information.

³ The minimum node size is set to 0, which eliminates the definition of the inverse node decay. Remember that for INEX 2006 there is no such thing as a 'too small' node.

This last aspect can also be observed in the ranking of Example B, where positions of nodes A,B, and C have been swapped. If a node contains all the requested information the two models behave similarly, as is the case for nodes F, and G on the XSee model.

2 Exploiting XML Structure

In this section we will first examine the structure of the XML collections at hand, after which we introduce a number of reranking methods that exploit certain structural aspects of the XML document.

2.1 Characteristics of IEEE and Wikipedia Collections

A starting point for using the document structure to improve on retrieval performance would be to have some statistics available on both the IEEE and the Wikipedia collection. In Table 2 the collection properties are given for both collections. In addition we present the characteristics of the full- and ideal recall base.

Based on the properties given for both collections, it can be easily seen that on average a document of the Wikipedia collection has fewer nodes (x5), with a smaller path depth (-1), a higher ratio of leaf nodes (x2), and an almost equal node size.

More differences can be observed when inspecting the full and ideal recall bases for both collections. Besides being based on different collections, one also has to take into account that the assessment procedure has changed, which will also have its impact on the characteristics of the recall bases. Nonetheless, our primary interest is investigating the structural aspects of the data, to derive effective methods for exploiting the document structure.

Comparing the full recall base against the IEEE collection reveals that if a document is relevant, only a small fragment is highlighted, while on the Wikipedia collection this portion is much larger. Interesting to see though is that the total number of nodes being highlighted remains the same.

Table 1. Relevance scores for Example A and B

Node	Example A		Example B	
	XSee	GPX	XSee	GPX
A	3 - 2.6	1 - 1.53	4 - 1.31	3 - 2.07
B	1 - 5.5	2 - 1.4	3 - 2	5 - 0.8
C	4 - 1.75	7 - 0.51	5 - 1.28	4 - 1.79
D	8 - 0.4	8 - 0.4	7 - 0.6	7 - 0.6
E	7 - 0.6	6 - 0.6	8 - 0.4	8 - 0.4
F	2 - 2.8	3 - 1.28	2 - 3.2	2 - 4.48
G	5 - 0.8	4 - 0.8	1 - 5	1 - 5
H	7 - 0.6	6 - 0.6	7 - 0.6	7 - 0.6

Table 2. characteristics of collection vs. full and ideal recall base

property	IEEE (2005)			Wikipedia (2006)		
	collection	full recall	ideal recall	collection	full recall	ideal recall
nodes / doc.	430.9	31.6	2.53	79.7	39.5	5.77
desc. / node	4.68	41.42	22.16	3.86	18.21	5.41
leaf nodes / nodes	0.319	0.428	0.641	0.68	0.34	0.60
path depth	5.68	3.68	4.05	4.86	5.07	3.74
node size	30.1	308.7	218.7	32.9	147.37	64.21

When inspecting the average path depth we see some more unexpected behaviour. For the full recall base on the Wikipedia collection the depth increases, while going from full recall base to ideal recall base it decreases again. The behaviour on the IEEE collection shows the opposite effect. We expect that the main cause is to be found in the fact that for INEX 2005 there was a notion of so called 'too small' nodes, which is not considered an issue in INEX 2006. The ratio of leaf nodes in a document going from full recall base to ideal recall base increases in both collections. The ratio does show a difference between both collections when comparing the collection and full recall base; on the IEEE collection the ratio first increases, while on the Wikipedia collection it decreases.

Given these huge differences in structure of the XML collections, and composition and structure of the recall bases, it is interesting to compare the performance of the retrieval model on the different collections, and the impact of the reranking methods.

2.2 Reranking Methods

Based on the characteristics of the IEEE collection of INEX 2005 and the corresponding recall bases, we have defined a number of reranking methods, which aim at using the structure of a document. In theory, these reranking methods can be applied for both the Thorough and Focused tasks as a post-processing step.

Rerank on node size Given a run R and a result element in $r \in R$ with relevance score rsv_r and $node_size_r$, the rerank method produces a run R' that computes a new relevance score $rsv_{r'}$ for each element $r' \in R'$ using the formula: $rsv_{r'} = \frac{rsv_r}{node_size_r}$. The effect of this method is that elements with a smaller node size, will be pushed higher in the ranking.

Rerank on leafs In this case the leaf property $leaf_node_r$ of an element r is used to compute a new ranking using the formula: $rsv_{r'} = rsv_r \cdot leaf_node_r$. With $leaf_node_r = 1$, if the node is a leaf node, e.g. one of its children is a text node, or $leaf_node_r = 0.5$ otherwise.

Rerank on descendants The descendant rerank uses the number of descendant nodes $descendants_r$ as a measure for reranking the relevance score of a node r using: $rsv_{r'} = \frac{rsv_r}{descendants_r + 1}$. The objective is to retrieve smaller nodes higher in the ranking.

Rerank on path depth Elements that are positioned deeper in the XML tree, or in other words have a larger path depth, are preferred over less deep positioned elements in a Focused run. The path depth $path_depth_r$ of an element r is used to compose a new ranking, where elements with longer paths will be ranking higher, using the formula: $rsv_{r'} = path_depth_r \cdot rsv_r$.

3 Thorough Task

In this section the results for the Thorough task are presented. We will start with discussing the basic configuration of the retrieval model and the official runs for the Thorough tasks. At the end of this section present improvements over the baseline that are obtained by applying the reranking methods discussed in the previous section.

3.1 Thorough runs - basic

The configuration of the official runs submitted for the Thorough task is based on their performance on the INEX 2005 data. It assumes that the configuration of retrieval model should be stable for different collections. In Table 3 the overall results for the official runs are presented. It shows that the configuration of best three runs on the IEEE collection, gives a similar performance on the Wikipedia collection. In Figure 2.a the performance of the official runs is plotted using effort precision vs gain recall. It reveals that there is room for improvement at the lower gain recall levels, e.g. the top of the ranking.

Table 3. Overall performance of the official runs on the thorough task

	$nd : .39F_t : 4$	$nd : .29F_t : 3$	$nd : .49F_t : 5$
MAep on IEEE	0.0767	0.0757	0.0708
MAep on Wikipedia	0.0302	0.0294	0.0290
Ranked	14	15	16

To gain more insight into the effect of different node decay vs. term factor parameters, we have measured the performance of the retrieval model on the thorough task on a range of node decays (0.09 - 0.49) and term factors (1 - 10), which results in the matrix of Table 4. It shows that optimal performance of the retrieval model is obtained, when using a node decay of 0.39, in combination with a term factor of 4. The highlighted cells around this configuration show

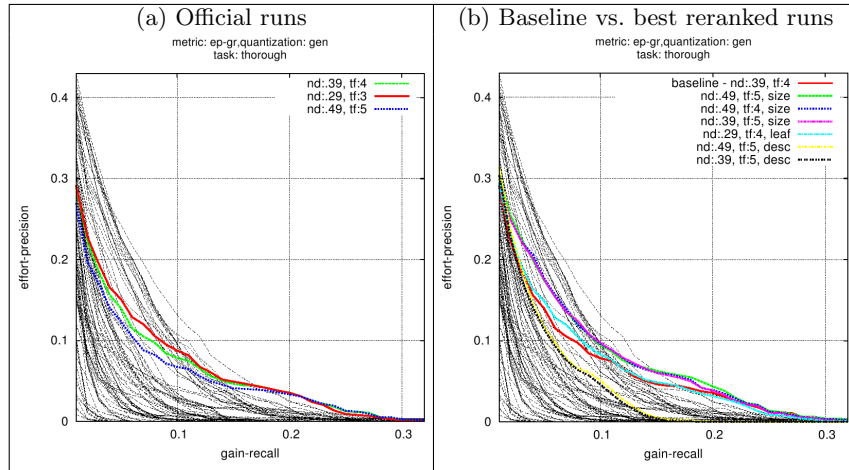


Fig. 2. Thorough runs: Effort precision vs. Gain recall

Table 4. Performance of thorough runs on MAep

		Node decay				
		0.09	0.19	0.29	0.39	0.49
Term factor	1	0.0146	0.0167	0.0184	0.0198	0.0208
	2	0.0210	0.0249	0.0273	0.0283	0.0286
	3	0.0237	0.0273	0.0294	0.0299	0.0292
	4	0.0249	0.0282	0.0299	0.0302	0.0294
	5	0.0255	0.0290	0.0301	0.0301	0.0290
	6	0.0257	0.0290	0.0301	0.0301	0.0289
	10	0.0265	0.0293	0.0299	0.0292	0.0281

how performance on the MAep is clustered around this optimal configuration of the basic retrieval model.

The results show that the performance of the basic retrieval model is stable, e.g. the same configuration leads to the best performance for the two collections, which is a desirable effect.

3.2 Thorough runs - using structure

Besides evaluating the new retrieval model, we also wanted to investigate how to effectively use the structural characteristics of the document. Based on the four reranking methods of Section 2, we have post-processed the set of thorough runs to see how we can benefit from the available of structure. Table 5 compares the performance of our baseline run $[nd : .39, F_t : 4]$ against the best performing reranked versions. In the table, we present the results of the best performing runs on both the MAep and nxCG@5 measures. When optimizing for MAep, we see that a reranking based on the size of the node is most effective, and results in a 6% improvement over the baseline when using the configuration

[$nd : .49, F_t : 5, size$]. While a increase in performance of 8.3% on $nxCG@5$ is obtained for the configuration [$nd : .49, F_t : 5, desc$], when reranking is based on the number of descendants.

Table 5. Effect of rerank based on structure

Run	MAep	$nxCG@5$	$nxCG@10$	$nxCG@25$	$nxCG@50$
nd:.39, tf:4	0.302	0.355	0.320	0.268	0.233
$nd : .49, F_t : 5, size$	0.0322	0.3338	0.3106	0.2831	0.2560
$nd : .49, F_t : 4, size$	0.0313	0.3206	0.3033	0.2798	0.2550
$nd : .39, F_t : 5, size$	0.0305	0.3324	0.3125	0.2829	0.2538
$nd : .29, F_t : 4, leaf$	0.0288	0.3767	0.3407	0.2824	0.2440
$nd : .49, F_t : 5, desc$	0.0230	0.3843	0.3695	0.2992	0.2426
$nd : .39, F_t : 5, desc$	0.0211	0.3797	0.3615	0.2979	0.2375
<i>Color coding:</i>	Best run	Top 3	Top 10	Top20	Other

Figure 2 shows the performance of the reranked runs against the baseline run. It shows that reranking runs based on size give better results on the effort precision, although it shows no specific improvement on the lower gain recall levels. Nonetheless, we can conclude that reranking based on size has a positive effect on the MAep, while reranking based on descendants is a useful tool to improve on the $nxCG5$ measures.

4 Focused Task

Goal of the Focused task is to return relevant elements regarding a query without elements overlapping each other. In Section 2 various methodes of exploiting XML structure of documents are explained and we used these methods for the Focused task.

Overlap removal After applying one of the reranking methods to the Thorough input run, overlap must be removed. We use a very straightforward method for this in which we go over the results per topic ordered by decreasing relevance score. Only elements that do not overlap with a previous selected element are selected for the Focused run.

Rerank vs. De-rerank. Evaluation of the reranking strategies on the IEEE collection shows that performances on $MAep$ can be improved, but the baseline runs outperform the reranked runs on the $nxCG$ metric, especially on low cut-off levels. A logical explanation for this symptom is that by reranking the input run, an element that was less relevant (element a) could be ranked above a more relevant element (element b) due to its characteristics. When both elements are preserved after the removal of overlap a is suddenly more relevant

than b . So although reranking makes it possible to select elements with the right characteristics, it does not improve ranking. To eliminate this error we propose a de-reranking strategy; de-reranking cancels the reranking after removal of overlap by returning the remaining elements to their original order.

4.1 Focused runs - submissions

Based on evaluation results of various configuration on the INEX 2005 collection and assessments, three configurations are selected to be submitted as official runs for INEX 2006. These runs differ from each other on node decay, term factor and (de-)reranking strategy. The performance results of the official runs are summarised in Table 6.

Table 6. Performance results for the official Focused runs

characteristic	nd:.19 tf:5 size rerank	nd:.04 tf:3 leaf rerank	nd:.04 tf2 desc. dererank
nxCG[5]	0.2907	0.3090	0.2641
nxCG[10]	0.2687	0.2840	0.2368
nxCG[25]	0.2342	0.2227	0.1856
nxCG[50]	0.2001	0.1910	0.1531
MAep	0.0541	0.0509	0.0304

On the nxCG metric with low cut-off levels we see that the run with leaf reranking [nd:.04 tf:3] performs best, but compared to other INEX participants results are modest (position 20–30). At higher cut-off levels and on MAep the run with size reranking [nd:.19 tf:5] performs best, and is ranked 11–16 on nxCG. The descendants de-reranked run [nd:.04 tf:2] cannot compete with other runs, even though it was the best performing run (on MAep) on the IEEE collection.

4.2 Focused runs - Wikipedia evaluation

The differences between the IEEE and Wikipedia collections identified in Section 2 indicate that other configurations might perform better on the Wikipedia collection than the best performing configurations on the IEEE collection. To examine whether this is true, we evaluated a large number of baseline Focused runs for the Wikipedia collection. Table 7 shows the top configurations on nxCG[5] and MAep without any (de-)reranking strategy applied to it.

From these results we can conclude that for a Focused run without (de-)reranking strategies a small node decay [nd:.04] is preferred; this smaller node decay makes sure smaller elements are awarded a relatively higher relevance score. The term factor should be between 3 and 5 to get the best results. The results of the official runs compared to the results of the baseline runs also show that using structure in reranking the runs has a positive influence on MAep.

Table 7. Performance results for baseline Focused runs

Run	nxCG[5]	nxCG[10]	nxCG[25]	nxCG[50]	MAep
nd:.04 tf:3	0.3099	0.2699	0.2062	0.1772	0.0494
nd:.04 tf:4	0.3324	0.2760	0.2104	0.1766	0.0486
nd:.04 tf:6	0.3311	0.2703	0.2100	0.1734	0.0476
nd:.04 tf:10	0.3396	0.2706	0.2047	0.1656	0.0462
nd:.04 tf:4	0.3324	0.2760	0.2104	0.1766	0.0486
nd:.04 tf:5	0.3320	0.2730	0.2029	0.1709	0.0472
<i>Color coding:</i>	Best run	Top 3	Top 10	Top 20	Other

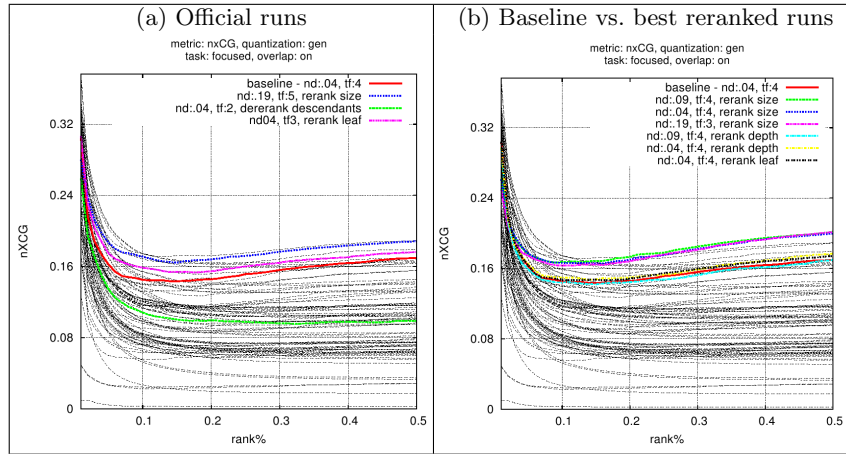


Fig. 3. Focused runs: Effort precision vs. Gain recall

For each of the node decays [nd:.04 nd:.09 nd:.19] we selected the best configuration [tf:4 tf:4 tf:3] to apply the (de-)reranking strategies. The results of the best adjusted Focused runs on nxCG[5] and MAep compared to the best baseline run [nd:.04 tf:4] are shown in Table 8.

The size reranking strategy shows excellent results on MAep, with a maximum performance increase of 15.8% compared to the [nd:.04 tf:4] baseline run. The results also show that the best Focused baseline configuration [nd:.04] is not necessarily the best configuration for the (de-)reranking input.

5 Conclusions

In this article we have presented a novel model for XML retrieval that is used by the XSee system. It is derived from the GPX system, and aims to provide a scalable approach to XML retrieval, while maintaining good retrieval properties. The model has shown to be stable under changing collection characteristics for the Thorough task.

From the special focus on exploiting structural characteristics of XML document collections, we have learned that reranking based on node size is a use-

Table 8. Performance results for (de-)reranked Focused runs

Run	nxCG[5]	nxCG[10]	nxCG[25]	nxCG[50]	MAep
nd:.04 tf:4	0.3324	0.2760	0.2104	0.1766	0.0486
nd:.09 tf:4 rerank size	0.2808	0.2408	0.2105	0.1864	0.0563
nd:.04 tf:4 rerank size	0.2762	0.2375	0.2111	0.1875	0.0556
nd:.19 tf:3 rerank size	0.2522	0.2269	0.2064	0.1811	0.0547
nd:.09 tf:4 rerank depth	0.3288	0.2695	0.2063	0.1676	0.0486
nd:.04 tf:4 rerank depth	0.3260	0.2751	0.2093	0.1756	0.0497
nd:.04 tf:4 dererank size	0.3236	0.2724	0.2071	0.1747	0.0479
<i>Color coding:</i>	Best run	Top 3	Top 6	Top 12	Other

ful feature to enhance the retrieval performance in terms of MAep in a post-processing process. However, the reranking methods are less successful in improving the top of the ranking given the results for the nxCG@5,10,25,50 measures. Improvement can however be found on the nxCG family of measures when inspecting the higher recall levels.

References

1. Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai, editors. *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005, Revised Selected Papers*, volume 3977 of *Lecture Notes in Computer Science*. Springer, 2006.
2. Shlomo Geva. Gpx - gardens point xml information retrieval at inex 2004. In Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltán Szlávik, editors, *INEX*, volume 3493 of *Lecture Notes in Computer Science*, pages 211–223. Springer, 2004.
3. Roelof van Zwol. B^3 -sdr and effective use of structural hints. In Fuhr et al. [1], pages 146–160.
4. Wouter Weerkamp. Optimising structured document retrieval using focused and relevant in context strategies. Master’s thesis, Utrecht University, the Netherlands, 2006.

Using Rich Document Representation in XML Information Retrieval

Farhad Oroumchian¹, Fahimeh Raja², Mostafa Keikha², Masued Rahgozar²

¹The College of IT, University of Wollongong in Dubai

FarhadO@uowdubai.ac.ae

² Database Research Group, Control and Intelligent Processing center of Excellence, faculty of ECE, School of Engineering, University of Tehran, Tehran, Iran

{f.raja,m.keikha}@ece.ut.ac.ir

rahgozar@ut.ac.ir

Abstract. In this paper we present an XML information retrieval approach for retrieving meaningful units of XML documents based on Rich Document Representation (RDR). RDR is a form of document representation that utilizes single words, phrases, logical terms and logical statements for representing documents. Vector Space model is used to compute index terms weight and similarity between each element and query. This system has participated in INEX 2006 and tested with 125 Content Only queries of the given collection. The results have been less than expected but an analysis of failure has revealed that it has been caused by a mismatch between the IDs we assigned to document elements in processing single terms, phrases and logical terms and statements.

Keywords: Rich Document Representation, XML Information Retrieval.

1 Introduction

The widespread use of Extensible Markup Language (XML) has brought up a number of challenges for information retrieval (IR) systems. In contrast with traditional IR, XML information retrieval exploits the logical structure of documents and is concerned not only with finding relevant documents but with finding the most appropriate unit in the document that satisfies users' information need [1].

In this paper we present an NLP approach for representing XML text in XML information retrieval. In this method we use Rich Document Representation (RDR) to represent documents by their single words, phrases, logical terms and logical statements.

The most popular document representation in IR is called single term where stemmed single words are used as a representation of document [2]. A more sophisticated representation is based on single terms and phrases. These phrases could be formed statistically or linguistically. The usefulness of using phrases and their contribution largely depends on the type of the system and weighting scheme used [3]. Adding phrases to a single term representation in vector space system with a good

weighting such as Lnu.ltu will only add 3-5% [4] to precision. However, in a system with weaker weighing the contribution of the phrases could be as much as 10% [4].

However there is more information in the text that could be used as a document representation. Since text is written purposefully for a specific reason, it is full of clues to help clarify its meaning. There are many studies into identifying and representing these relationships. The number of reported relationships varies from 10 to 120 based on the context and system [5][6]. However in this research we are introducing a few relationships that could be used for representing documents. The main advantage of these relationships is that they could be converted into logical forms similar in syntax to multi-valued logic of Michalski [7].

This representation is the main document representation of a system called PLIR (Plausible Information Retrieval). PLIR assumes that retrieval is an inference as pointed out by Prof. Van Rijsbergen [8]. However, unlike Van Rijsbergen work which has only a single inference, PLIR uses many inferences of the theory of Human Plausible Reasoning and offers calculations for the certainty of the inferences. PLIR outperforms a typical vector space model [9]. Previously, it was proven that RDR is better representation for clustering than single words in the second stage of a two stage retrieval system [10].

The rest of the paper is organized as follows: Section 2 will introduce RDR as our method of representing document text. In Section 3 we will describe our experiments and weighting scheme used in this study. Section 4 is about our results in INEX 2006. This section gives a brief description of the cause of our poor results in this competition. Finally Section 5 concludes the article.

2 What is RDR?

Rich Document Representation (RDR) is a method of representing documents by logical forms with the syntax of multi-valued logic. These logical forms could be any of the following:

1. Concepts (single stemmed words and phrases)
2. Logical terms: logical terms are in the form of $A(B)$ where A is the descriptor and B is the argument. Logical forms are similar to predicates in Predicate logic. A logical term can have a descriptor and one or more arguments.
3. Logical statements: logical statements are in the form of $A(B) = \{C\}$ Where A is the descriptor and B is the argument and C is the referent. For example Flower (England) = {Daffodil} which basically means Daffodils are flowers of England.

Multi-valued logic allows logical statements to have multiple referents or refer to an infinite set. However in the context of IR, there is no infinite document or sentence therefore it is not possible for logical statements to reference an infinite set. Also, in order to make matching concepts easier, logical statements have been restricted to have only a single referent. Statements that could translate to Logical statements with multiple referents are written as multiple logical statements with the same descriptor and argument and a single referent.

PLIR uses RDR as the main document representation. PLIR treats all the statements representing all the documents as a single knowledge base that describes a possible world which includes some documents. In such a space, statements of different documents could provide complement each other and provide more information. In that space, PLIR uses reasoning in order to guess the relevancy of documents to queries. In a normal vector space model, there is no reasoning so the only use of RDR would be providing a deeper representation for the text.

RDR representation is extracted automatically from the text. The process of producing these logical forms is as follows:

1. Tag the text: The text has to be tagged by Part of Speech tags.
2. Rule based or Clue based extraction: in this process the output of the POS tagger is scanned for clues. These clues signify the existence of the relations in the text. For example a proposition such as ‘of’ or ‘in’, signifies a relationship between two noun phrases that it connects.

Table 1 shows a few sentence fragments and their equivalent in logical forms.

Table 1. A few sentence fragments and their equivalent in logical forms

Sentence fragment	Representation in multi-valued logic and PLIR	Type
Resonances for glutamine	Resonance(glutamine)	Logical Term
Syngeneic tumour of BALB	Syngeneic_tumour(BALB)	Logical Term
Linux as an operating system for PCs	Operting_system(PC) ={Linux}	Logical Statement
Kish, which is an island in Persian Gulf	Island (Persian_Gulf) ={Kish}	Logical Statement
	ISA (island, Kish)	

Several different tools have been developed and used for processing text and extracting relationships so far, some of them are written in Perl, some others in Java. The most general tool so far is a general purpose NLP package written in Java. We are in the process of completing it in order to cover all aspects of relation extraction and generating representation. After that we can make it available for public use.

2 Experiments

In our implemented system, first we index all the elements of a document by its single terms, phrases and logical terms and statements. In this system, 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, in the query “algorithm for index compression” a logical term will be detected and represented as “algorithm(index_compression)”. Therefore, first we scan the query and extract single terms, phrases and logical terms and then we find all the references in the collection for the followings:

1. All the single words such as “algorithm” in the query.
2. All the phrases such as “index_compression” in the query.

3. All the logical terms such as “algorithm(index_compression)” that are in query.

This is a case of direct retrieval where the document is indexed by the term. This action is similar to regular keyword matching by search engines except that search engine do not index logical terms.

Any kind of weighting can be used for weighting these logical terms and statements. The best way of weighting them is treating them as phrases and weight them accordingly. However, PLIR uses a different scheme which is called Dominance. In this work we use "tf.idf" in our weighting model. We apply the following formula for weighting each index term of queries:

$$w(t, q) = \alpha * termFreq(t, q) * idf(t) * nf(q) \quad (1)$$

Where:

$$\alpha = \frac{2}{3}, \text{ for terms with " _" before them in the query} \quad (2)$$

$$= \frac{4}{3}, \text{ for terms with "+" before them in the query}$$

$$= 1, \text{ otherwise}$$

$$idf(t) = \ln\left(\frac{N}{n(t)}\right) \quad (3)$$

$$nf(q) = \frac{1}{lenq} \quad (4)$$

termFreq(t,q): frequency of occurrence of term t within the query q

idf(t): inverse document frequency of term t

N: number of documents in the collection

n(t): number of documents in the collection that contain term t

nf(q): normalization factor of query

lenq: query length which is equal to number of terms in the query

It should be mentioned that the parameter "nf" is computed separately for single terms, phrases, and logical terms and statements; i.e. "lenq" is calculated three times as follows:

- number of single terms in the query
- number of phrases in the query
- number of logical terms and statements in the query

For weighting index terms of document elements we use:

$$w(t, e) = tf(t, e) * idf(t) * nf(e) * childEffect(e) \quad (5)$$

Where:

$$tf(t, e) = \frac{(1 + \log(\text{termFreq}(t, e)))}{(1 + \log(\text{avg}(\text{termFreq}(e)))} \quad (6)$$

$$idf(t) = \ln\left(\frac{N}{n(t)}\right) \quad (7)$$

$$nf(e) = \frac{1}{\sqrt{\text{len}(e)}} \quad (8)$$

$$\text{childEffect}(t, e) = \frac{\text{\#of sublements of } e \text{ with term } t}{\text{\#of sublements of } e} \quad (9)$$

$\text{termFreq}(t, e)$: frequency of occurrence of term t within the element e

$\text{avg}(\text{termFreq}(e))$: average term frequency in element e

$\text{idf}(t)$: inverse document frequency of term t

N : number of documents in the collection

$n(t)$: number of documents in the collection containing term t

$\text{nf}(e)$: normalization factor of element e

$\text{len}(e)$: element length which is equal to number of terms in the element

$\text{childEffect}(t, e)$: effect of occurrence of term t within subelements of element e in the weight of element e

As the case for queries, the parameter " nf " for document elements is calculated separately for single terms, phrases and logical terms and statements.

After weighting index terms of queries and document elements, we made three separate vectors from the weights of single terms, phrases, logical terms and statements for each query and element in the given collection. For example, the corresponding vectors for the query " $a_b(c)$ " are:

$$V_{\text{Single_Terms}} = (w(a, q), w(b, q), w(c, q))$$

$$V_{\text{Phrases}} = (w(a_b, q))$$

$$V_{\text{Logical_Terms}} = (w(a_b(c), q))$$

Once vectors have been computed for the query and for each element, using a weighting scheme like those described above, the next step is to compute a numeric "similarity" between the query and each element. The elements can then be ranked according to how similar they are to the query.

The usual similarity measure employed in document vector space is the "inner product" between the query vector and a given document vector [11]. We use this measure to compute the similarity between the query vectors (for single terms, phrases, logical terms and statements) and an element vectors. Finally, we add these

three relevance values to get the total relevance value for each element and rank the elements based on this relevance value.

3 Results

Our implemented system has participated in INEX 2006 [12]. Our results are much lower than expected but in an analysis of failure it was discovered that we have some mistakes in preprocessing of the collection.

In the preprocessing phase of our experiments we assumed that the content of XML documents is placed only in paragraphs. Therefore, if we have text in elements other than paragraphs, we consider a child for it, called "*virtual paragraph (VP)*", and assign an ID to it. This preprocessing leads to mismatch between IDs assigned to elements for single terms, phrases, logical terms and statements. For example, consider the following element:

```
<section>
  a_b
</section>
```

For the text "*a_b*" in this element, we will add a child (VP) for both single terms and phrases, while we will not have this element for logical terms and statements, because it does not contain any logical term and statement. Therefore we have mismatch between element IDs in our system.

We resolved the problem in our system and currently we are running the (corrected) system again on the INEX 2006 collection and we hope to get better results.

4 Conclusion

In this paper we present an NLP/Logical approach that uses a rich set of features to represent text. These features are single terms, phrases, logical terms and logical statements. Logical terms and statements are extracted from text by using some clues. The simplest clue is the proposition. This representation is called RDR (Rich

Document Representation). We have experimented with INEX2004 collection and had satisfactory results demonstrating RDR's better representation of elements.

However, when we conducted our experiments using INEX 2006 test collection, the performance of our runs has been lower than our expectation due to indexing errors. Currently we are analyzing the results of our experiments and checking all the steps underlying the development of our system. We have identified the preprocessing mistakes that may have lead to poor results of our experiments. We are re-running the experiments and hoping to get satisfactory results this time.

In the light of the insight we gained, we are sure it is possible to improve this approach and to gain more respectful results. In a different direction, we are looking into using relevance feedback and learning methods to make it possible for the system to adapt itself to the user judgments.

References

1. Oroumchian, F., Karimzadegan, M., Habibi, J.: XML Information Retrieval by Means of Plausible Inferences. RASC 2004, 5th International Conference on Recent Advances in Soft Computing, RASC, Nottingham, United Kingdom (2004) 542-547
2. Salton, G., Allan, J., Buckley, C.: Approaches to passage retrieval in full text information systems. Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1993) 49-58
3. Fox, E.A.: Lexical relations: enhancing effectiveness of information retrieval systems. SIGIR Newsletter, Vol 15, No. 3 (1981) 5-36
4. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1996) 21-29
5. Evens, M., Vandendrope, J., Wang, Y.-C.: Lexical Relations in Information Retrieval, Human and Machines. Proceedings of The 4th Delaware Symposium on Language Studies, Albex Norwood N.J. (1985)
6. Cohen, P.R., Kjeldsen, R.: Information retrieval by constrained spreading activation in semantic networks. Vol. 23. Pergamon Press, Inc. Tarrytown, NY, USA (1987) 255-268
7. Collins, A., Michalski, R.: The Logic Of Plausible reasoning A Core Theory. Cognitive Science, Vol. 13 (1989) 1-49.
8. VAN RIJSBERGEN, C. J.: Logics For Information Retrieval, AL4T 88, Rome, March (1988)
9. Oroumchian, F., Oddy, R.N.: An application of plausible reasoning to information retrieval. Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Switzerland, August (1996) 18-22
10. Oroumchian, F., Jalali, A.: Rich document representation for document clustering. RIAO 2004 Conference Proceedings: Coupling approaches, coupling media and coupling languages for information retrieval, Le Centre de Hautes Etudes Internationales d'Informatique Documentaire - C.I.D., France (2004) 1-9

11. Greengrass, E.: Information Retrieval: A Survey. DOD Technical Report TR-R52-008-001 (2000)
12. <http://inex.is.informatik.uni-duisburg.de/2006/>accessed 25th of November 2006

NLPX at INEX 2006

Alan Woodley, Shlomo Geva

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

Abstract. XML information retrieval (XML-IR) systems aim to better fulfil users' information need than traditional IR systems by returning results lower than the document level. Historically, these structured queries have been formatted using formal languages such as NEXI. Unfortunately, formal query languages are very complex and too difficult to be used by experienced - let alone casual - users and are too closely bound to the underlying physical structure of the collection. INEX's NLP task investigates the potential of using natural language to specifying structured queries. QUT has participated in the NLP task with our system NLPX since its inception. Here, we discuss the changes we've made to NLPX since last year, including our efforts to port NLPX to Wikipedia. Second, we present the results from the INEX 2006 where NLPX was the best performing participant in the Thorough and Focused tasks.

Shallow parsing of INEX Queries

Haïfa Zargayouna, Victor Rosas, Sylvie Salotti

LIPN, Université Paris 13 - CNRS UMR 7030,
99 av. J.B. Clément, 93440 Villetaneuse, France.
firstname.lastname@lipn.univ-paris13.fr

Abstract. This article presents the contribution of the Laboratoire d'Informatique de Paris Nord (France) to the *Natural Language Queries to NEXI (NLQ2NEXI)* task (part of the *Natural Language Processing (NLP)* track) of the *Initiative for Evaluation of XML Retrieval (INEX 2006)*. It discusses the use of shallow parsing methods to analyse natural language queries.

1 Introduction

XML Information Retrieval (XML-IR) systems combines features from traditional IR and from database retrieval. Their aim is to find *relevant parts* of information according to explicit documents' structure and content semantics. XML-IR interests both IR and database communities which propose formal languages to help users express their needs according to content and structure. Unfortunately, these languages are complex and difficult to use even by experienced users. The need to know well the documents' structure is also an obstacle to their use. Research on Natural Language Interfaces for the retrieval of XML documents has become increasingly acute as more and more casual users access a wide range of document collections. Our aim is to translate Natural Language Queries (NLQ) into a formal query language (i.e NEXI).

Automatically understanding natural language is still an open research problem. The translation of natural language queries into a database query language has been addressed by [1], [2], [3]. Natural language interfaces for an XML database is proposed by [4], [5]. The goal of these interfaces is the same as for IR: mapping the intent into a specific database schema. But the extracted informations are different such as grouping and nesting, variable binding, etc.

Natural language interfaces for an XML retrieval are proposed by [6] and [7]. We aim to evaluate the contribution of existant robust NLP (Natural Language Processing) tools to extract the main query parts: content, query, boolean operators. This analysis produces an annotated NLQ which is then formalized in any formal language.

The remainder of this paper is organized as follows. First we present the motivations for this research. Next, we introduce the INEX campaign' context. We detail our propositions' overview in section 4. Section 5 to 8 detail the proposed components. We conclude with a discussion of future work.

2 INEX context

In the 2006 campaign, the previous topic types are all merged into one type: Content Only + Structure (CO+S). The CO+S topics consist of topics where structure hints can be specified to help the system but they are not mandatory.

We participate to the NLQ2NEXI which is a task that requires the translation of a natural language query, provided in the descriptive part of a Co+S topic, into a formal NEXI query. Two parts of the CO+S topics are of interest (see an example in figure 1):

- <castitle>: in which Content And Structure (CAS) queries are given
- <description>: from which NLP queries are derived.

The proposed systems have to analyse automatically the *description* part in order to generate automatically *castitle* part.

```
<inex_topic topic_id="301" ct_no="37">
...
<castitle>
//article[about(.,Algebraic models)OR about(.,vector space model or generalized vector space model or latent semantic indexing)]
//section[about(./title,Topic-Based vector space model) or about(./title, Latent Semantic Indexing)or about(./title,extended Boolean model) or
about(./title, enhanced topic based) or about(./title, Salton SMART)]
</castitle>
<description>
Seek articles about about Algebraic models such as Vector Space Model, the generalized vector space model or Latent Semantic Indexing.
I also need sections with titles Topic-Based vector space model, Latent Semantic Indexing, extended Boolean model, enhanced topic based and
finally Salton SMART
</description>
...
</inex_topic>
```

Fig. 1. An extract of the topic 301

The castitle part contains the query formalised in NEXI [8]. A NEXI query has the following form:

$$//s_1[c_1]//s_{k-1}[c_{k-1}]/s_k[c_k]$$

It applies to a sequence of XML elements and returns elements of type s_k about c_k which are descendants of elements of type s_{k-1} about c_{k-1}, \dots , which are elements of type s_1 about c_1 .

Predicates c_1, \dots, c_{k-1}, c_k are built from textual or numerical predicates connected by disjunctive or conjunctive connectors. A textual predicate has the following form:

about(relative_location_path, string_litteral)

and a numerical predicate has the following form:

relative_location_path connector string_litteral

We studied the submitted topics in 2006. The queries are very diverse but we can make these distinctions between different types of query:

- Question Answering type : This type of query requires more precise responses to users’ queries than Information Retrieval. There is no structural hint to help the system generating a formal (structured) query. Example, the query 380 (*How should I prepare acorns for eating?*). This type of query is hard to analyse to find a relevant structure in the context of the actual INEX corpora. As the elements’ structures are essentially logical and typographical. Nevertheless, we can envision in a ”semantically” structured corpora to infer that the answer can be retrieved in *Recipe* element for example.
- Queries containing no structure: This type of query are asked by a user who does not know (or does not want to use) the actual structure of the XML documents in a query. Example query 320 (*I’m looking for ways of transport from Gare de Lyon station to Gare du Nord in Paris*). The query is therefore a Content Only (CO) Query.
- Queries containing structure: The user expresses constraints on structure elements that she looks for or want to be returned.

Table 1 presents statistics¹ about the percentage per type of query in the topics proposed by participants.

Q/A queries	CO queries	COS queries
14 → 11%	80 → 64%	31 → 25%

Table 1. Query type statistics

Only approximately 25% of the topics precise structural constraints, this is -in our opinion- due to the fact that there is no clearly identified structure. The efforts made to extract all possible paths show that the structure is quite complex, the elements’ names are sometimes not understandable. We will discuss this point in section 5. We test our approach on all query types, even if intuitively it has to be more efficient for COS queries.

3 Motivations

The problem with formal XML Information Retrieval query languages is that they are difficult to use and users require an intimate knowledge of documents’ structure. Formulate the user need is then limited by these two aspects. We submitted two runs in 2006 campaign: manual and automatic. We discover that our

¹ This is an approximate repartition analysis, we can consider that ”what” questions can refer to the definition structure or that queries containing ”I want articles” do not provide structure hints.

manual formulation of NEXI queries varies from submitted queries by participants (we call them reference queries). Only 7,2% of the NEXI reference queries are equivalent to NEXI manual queries. In our opinion the main differences are caused by the implicit knowledge of retrieval system capacities. For example people do not differentiate or/and operators (for content) as the majority of retrieval systems relax these constraints. The difference in structure elements is also important (see table 2), this can be due to the complexity of the collection schema (see section 5).

Difference degree	1	2	3	4
Percentage	33,6%	38,4%	64,8%	11,2%

Table 2. Differences between human query formulation. Difference degree 1 means difference in boolean operators' use, 2: difference in content terms, 3: difference in structure used, 4: difference in boolean, content and structure

We propose a shallow parsing method that does not go in disambiguation processes or anaphora recognition. The offered services by the NLP interface are related to the language expressiveness. There is no need, for example, to extract boolean operators if the formal language do not allow to express such constraints. We propose a method that extract the main parts of COS queries (content, structure and operators). Our aim is to test the validity of a lexical approach combined with a "light" semantic matching (for structure extraction).

4 Propositions' overview

Our approach is subdivided in two phases (see figure 2):

1. NLQ Analysis: This phase consists in extracting the content, structure and boolean operators of the query. The extracted segments are annotated in XML format and can be translated to any XML query language.
2. Query Generation: This phase consists in taking into account NEXI expressiveness and syntax requirements to generate a well-formed NEXI query. This module can be adapted to generate formal queries in other languages such as XOR [9], XQuery [10], etc.

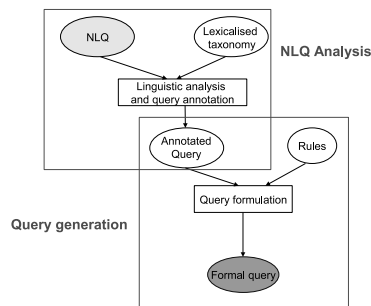


Fig. 2. Two main modules

The analysis phase is based on a *part-of-speech* (POS) tagging. It is composed by the following components:

- Extracting the structure: This component focuses on chunks of the NLQ that could allow us to identify which are the structural hints. We associate these NLQ chunks to the relevant structuring elements using an external resource: taxonomy of Wikipedia structuring elements².
- Extracting the content: This component identifies and extracts the chunks that will form the content terms of the formal query. Those chunks are the information requested by the user which might be retrieved in the documents.
- Extracting the operators: This part of the analysis identifies and extracts chunks of the NLQ that can be associated with boolean operators. The boolean operators can be associated to content and structure.

To carry out the cited steps of the NLQ analysis we use NooJ. NooJ is a linguistic development environment that includes large-coverage dictionaries and grammars, and parses corpora. It includes tools to create and maintain large-coverage lexical resources, as well as morphological and syntactic grammars. Dictionaries and grammars are applied to texts in order to locate morphological, lexical and syntactic patterns and tag simple and compound words [11].

² For instance, build manually.

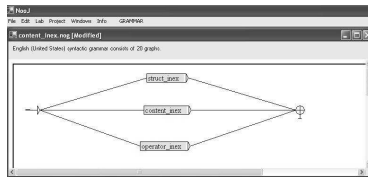


Fig. 3. Extraction of Content, structure and boolean operators with NooJ

Context free Grammar rules are defined in order to extract the main segments (structure, content and boolean operators). More specific rules are applied to each segment. We detail each part in the following sections.

5 Recognizing structure expression

INEX 2006 uses a document collection made from English documents from Wikipedia. The collection contains more than 5000 different elements. In this context, it is difficult to envision that user could have a precise idea of the structure. The lack of an annotated DTD to know tags' semantics leads to the large number of CO query. The COS queries use the most common elements (article, section, paragraph, figure, etc.)

In order to overcome this problem, we define a taxonomy³ that abstracts the global schema of the collection. The concepts are related to possible paths in the collection. For example the concept section is related to *article/body/section/*, *article/body/indentation1/sub/sub/sub/sub/sub/sub/sub/sub/section/*, *article/body/notinclud/section/*, etc. Possible paths are quite complex, without semantics we are unable to choose a path. One possible solution is to associate the concept to the disjunction of all possible paths. We choose to compute an abstract path which summarizes all the possible paths (i.e. contains common elements). The concept section is then related to the path *article//section*. Concepts are also lexicalised (see figure 4) which is helpful to find different occurrences (structural hints) of the concept in the NLQ (for example the concept chapter can be lexicalised by chapter, chapters, chapitre, etc.). The use of lexicalised taxonomy is particularly interesting in heterogeneous collections or multilingual ones.

³ A taxonomy is the classification, or categorization, of things. It is composed of concept related by hierarchical relations (is-a, part-of).

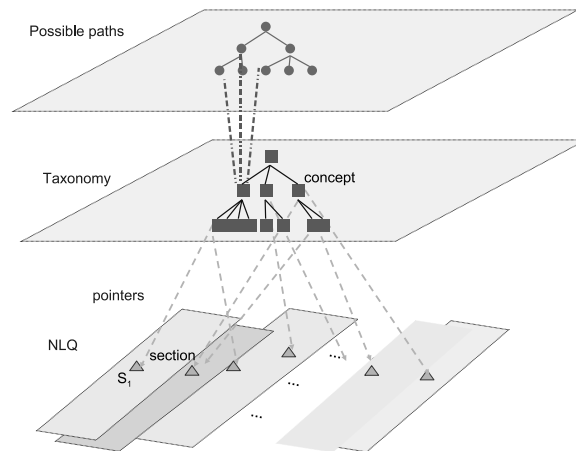


Fig. 4. The use of taxonomy : an intermediary abstraction between structural hints and elements

The identified structure segment is annotated by an XML element: `<Structure value="path"> structural chunks </Structure>` or `<ResultStruct value="path"> structural chunks </ResultStruct>` if the result structure is specified.

For example the query *306 Seek articles about genre theory, classification and other structuralist approaches. In particular I would like sections regarding Plato and Aristotle's ideas of form.* is annotated by:

```
<Structure value="//article"> articles </Structure> ...
<ResultStruct value="//article//section"> In particular sections </ResultStruct>
```

We ignore chunks before structural hints because generally they contains introductory phrases (e.g: Find, Seek, Locate, Retrieve, I want, I would like, etc.). In case where no structural hints are found (the case of CO queries), the query is annotated with the root element. Many NL queries contain as structural hints the terms *information* or *passage*, this is a generic structural hint which means that the retrieval system has to identify the relevant element to return. This minimal unit of information depends on documents structure, we make the hypothesis that for INEX collection, this unit can be `//article/(section)p`.

- or: expressed by "or", "such as", ",", " if follows "such as" ...
- and: expressed by "and", "also like", ",", "also need" ...
- not: expressed by "not", "nor" ...

We annotate the chunks by `<Operator value="operator-type">`.

The binary operators (or, and) are applied to the (right and left) nearest segment (structure or content). The operator not is applied to the nearest right segment.

For example, the query 292:

*I want figures containing Renaissance paintings of Flemish **or** Italian artists, but **not** French **or** German ones.*

is annotated by:

```
<Content1 value="Flemish">Flemish</Content1> <Operator value="or">or</Operator>
<Content2 value="Italian artists"> <Operator value="not"> not </Operator>
<Content1 value="French"> French </Content1> <Operator value="or">or</Operator>
<Content2 value="German ones">
```

8 Constructing the NEXI query

The analysis phase produces the NL query annotated in XML. The generation phase consists in translating the XML representation into a formal query with respect to expressiveness and syntactic constraints of the chosen language.

We use the annotations' value to generate the equivalent form:

- The Structure annotation value is reproduced. We process some factorization when needed. For example if there is two extracted paths `//A//B` and `//A//B//C`, we generate `//A//B[about(., ...) ... about(./C, ...)]`. Factorization is computed by finding shared ancestors between elements in the specified paths.
- The content annotation value is translated in the about clause, "" are added to complex terms.
- Boolean operators are applied to content and structure with the adequate syntax . For example `|` for the structure, if no need to factorize, and `OR` for the content. We applied "associativity and commutativity" to operators. For example the query *C and A or B* is translated to `about(., C A) OR about(., C B)`. The negation is applied only to content with the minus operator. For example "not C" is translated to `about(., -C)`.

9 Conclusion and Perspectives

For casual users formulating NEXI queries to express their information needs is a difficult task. Indeed we can imagine how difficult it could be for a doctor to write

a NEXI query. But we can also imagine how useful it could be for this doctor, looking for diabetes diagnosis information for example, to be able to extract only this specific information from medical XML documents. But formulating NEXI queries is not only a difficult task for casual users. Indeed, there is always syntactically correct queries that does not meet the descriptive part⁴.

We proposed a shallow parsing method to extract main components of NL queries. Considerable efforts needs to be done to digest the vast number of elements introduced in this year's INEX DTD, to enable to identify correctly the structural hints and to make sense of their implications for future adaptations of our system.

Unfortunately, we did not have time to interpret our results. We need to analyse the overall behavior of our method and its efficiency for the different type of queries (QA, CO and COS).

References

1. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: Natural language interfaces to databases - an introduction. *Natural Language Engineering* **1**(1) (1995) 29–81
2. Hulgeri, A., Bhalotia, G., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword search in databases. In: *IEEE Data Engineering Bulletin*. (2001)
3. Popescu, A., Etzioni, O., Kautz, H.: Towards a theory of natural language interfaces to databases. In: *Proceedings of the conference on Intelligent User Interfaces*. (2003)
4. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: Xsearch: A semantic search engine for xml. In: *VLDB*. (2003)
5. Li, Y., Yang, H., Jagadish, H.: Constructing a generic natural language interface for an xml database. In: *EDBT, LNCS 3896*. (2006)
6. Tannier, X.: From natural language to nexi, an interface for inex 2005 queries. In: *Advances in XML Information Retrieval: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval*. (2005)
7. Woodley, A., Geva, S.: Nlpx at inex 2005. In: *Advances in XML Information Retrieval: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval*. (2005)
8. O'Keefe, R., Trotman, A.: The simplest query language that could possibly work. (2003)
9. Geva, S., Hassler, M., Tannier, X.: Xor - xml oriented retrieval language. In: *Proceedings of SIGIR 2006, XML Element Retrieval Methodology Workshop*. (2006) 5–12
10. Boag, S., Chamberlin, D., Fernández, M., Florescu, D., Robie, J., Simon, J.: Xquery 1.0: An xml query language. *W3C Working Draft* (2005)
11. Silberztein, M.: Nooj: a linguistic annotation system for corpus processing. In: *Proceedings of HLT/EMNLP 2005 Interactive Demonstration*. (2005) 10–11

⁴ For example, in the NEXI query `//article[about(., "immanuel kant" AND "moral philosophy")]//section[about(., "categorical imperative")]`, the AND operator in the about clause will not be interpreted as a boolean operator but as a string.

The Heterogeneous Collection Track at INEX 2006

Ingo Frommholz¹ and Ray Larson²

¹ University of Duisburg-Essen
Duisburg, Germany

`ingo.frommholz@uni-due.de`

² University of California
Berkeley, California 94720-4600
`ray@sims.berkeley.edu`

Abstract. While the primary INEX test collection is based on a single DTD, it is realistic to assume that most XML collections consist of documents from different sources. This leads to a heterogeneity w.r.t. syntax, semantics and document genre. In order to cope with the challenges posed by such a diverse environment, the heterogeneous track was offered at INEX 2006. Within this track, we set up a collection consisting of several different and diverse collections. We defined retrieval tasks and identified a set of topics. These are the foundations for future run submissions, relevance assessments and proper evaluation of the proposed methods dealing with a heterogeneous collection.

1 Introduction

The primary INEX test collection has been based on a single DTD. In practical environments, such a restriction will hold in rare cases only. Instead, most XML collections will consist of documents from different sources, and thus with different DTDs or Schemas. In addition, distributed systems (federations or peer-to-peer systems), where each node manages a different type of collection, will need to be searched and the results combined. If there is a semantic diversity between the collections, not every collection will be suitable to satisfy the user's information need. On the other hand, querying each collection separately is expensive w.r.t. communication costs and result post-processing, therefore it has been suggested in the distributed IR literature that preselection of appropriate collections should be performed. Given these conditions and requirements, heterogeneous collections pose a number of challenges for XML retrieval, which is the primary motivation for including a heterogeneous track in INEX 2006.

2 Research Questions

Dealing with a set of heterogeneous collections that are syntactically and semantically diverse poses a number of challenges. Among these are:

- For content-oriented queries, most current approaches use the DTD or Schema for defining elements that would form reasonable answers. In heterogeneous collections, DTD-independent methods need to be developed.
- For content and structure queries, there is the added problem of mapping structural conditions from one DTD or Schema onto other (possibly unknown) DTDs and Schemas. 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. One goal of an INEX track on heterogeneous collections is to set up such a test collection, and investigate the new challenges posed by such structural diversity.
- Both content-only and content-and-structure approaches should be able to preselect suitable collections. This way, retrieval costs can be minimised by neglecting collections which would probably not yield valuable answers but are expensive to query w.r.t. communication efforts.

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

- For content-oriented queries, what methods are possible for determining which elements contain reasonable answers? Are pure statistical methods appropriate, or are ontology-based approaches also helpful?
- What methods can be used to map structural criteria onto other DTDs?
- Should mappings focus on element names only, or also deal with element content or semantics?
- How can suitable collections be preselected in order to improve retrieval efficiency and without corrupting retrieval effectiveness?
- What are appropriate evaluation criteria for heterogeneous collections?

In order to cope with above questions, we need collections which are both heterogeneous syntactically (based on different DTDs) and semantically (dealing with different topics, in this case from computer science research to IT business to non-IT related issues). As in the previous years, the main focus of effort for the track was on the construction of an appropriate testbed, consisting of different single collections, and on appropriate tools for evaluation of heterogeneous retrieval. The testbed provides a basis for the exploration of the research questions outlined above.

3 Testbed Creation

In order to create a testbed for heterogeneous retrieval, we had to find suitable collections first. Subsequently, corresponding topics had to be found and relevance assessments to be performed.

3.1 Collection Creation

We reused several subcollections offered in the last years' and the current INEX runs. Most of the collections from previous years of the Heterogeneous track were

restructured so that each document was a separate file embedded within a new directory structure, in order to be able to use the normal INEX evaluation tools. Additionally, we downloaded and prepared new collections like ZDNet News (IT related articles and discussion) and IDEAlliance. A specific DTD was defined for every subcollection, if not already given, ensuring syntactic heterogeneity. Table 1 shows some statistics about the subcollections.

Collection	Size	SubColl.	Documents	Elements	Mean Elements per Document
Berkeley	52M		12800	1182062	92.3
bibdb Duisburg	14M		3465	36652	10.6
CompuScience	993M		250986	6803978	27.1
DBLP	2.0G		501102	4509918	9.0
hcibib	107M		26390	282112	10.7
IEEE (2.2)	764M		16820	11394362	677.4
IDEAlliance	58M	eml	156	66591	426.9
		xml1	301	45559	151.4
		xml2	264	58367	221.1
		xmle	193	32901	170.5
		xtech	71	14183	199.8
Lonely Planet	16M		462	203270	440.0
qmulcsdbpub	8.8M		2024	23435	11.6
Wikipedia	4.9G		659385	1193488685	1810.0
ZDNet	339M	Articles	4704	242753	51.6
		Comments	91590	1433429	15.7
Totals	9.25G		1570713	1219818257	776.6

Table 1. Components of the heterogeneous collection. *Element counts estimated for large collections.*

The subcollections serve different domains, ranging from computer science (e.g. bibdb Duisburg, IEEE, DBLP) through technology news (ZDNet) to travel advice (Lonely Planet) and general purpose information (Wikipedia). We find several document genres like metadata records, fulltexts of scientific papers, articles and web sites as well as textual annotations which form discussion threads. The bottom line is that we have subcollections which differ with respect to their syntax (DTD), semantic (domains served) and document genre.

3.2 Topic Creation

The topic creation phase resulted in 61 topics. Among these are selected topics of the adhoc track as well as 36 new topics created especially for the heterogeneous track. In order to develop the latter topics we used the Cheshire II¹ system.

¹ <http://cheshire.lib.berkeley.edu/>

Converting the subcollections into a unified internal representation turned out to be a very time-consuming task as new collections had to be incorporated. Besides keyword and titles, also content and structure titles were given for most topics in order to make them suitable for the CAS task. Whenever given, the scope of a topic provides a hint about the collection used to identify the topic (which in fact does not necessarily mean that a topic is not relevant for other subcollections as well). Appendix A shows the topic DTD used.

4 Tasks and Run Submissions

The following tasks were proposed for this year's heterogeneous track:

Adhoc CO Task Here, content-oriented queries are implied. The systems return a ranked list of documents from all collections.

CAS Task 1 The system should return only elements specified in `<castitle>`.

CAS Task 2 The system should basically return the elements specified in `<castitle>`, but also similar elements. As an example, `<doctitle>` in ZDNet and `<title>` in other collections are most probably equivalent. The `<description>` in ZDNet, which is the description grabbed from RSS feeds, is similar, but not equivalent, to the `<about>` tag elsewhere. A possible scenario would be a system which likes to present the user only the title and a representative summary of the content so that she could decide if a document is relevant or not without higher cognitive overload (the need for reading the whole article). Furthermore, short comments could be presented here as well.

Resource Selection The goal here is to select the most relevant resources (i.e., collections) for a given topic. The system should return a ranked list of collections for this task. The scenario is that a system should identify relevant collections beforehand and query them, instead of querying all resources (which might be expensive when it comes to communication or access costs).

For run submissions we defined a DTD which can be viewed in Appendix B. This DTD covers rankings of elements as well as rankings of subcollections. Note that this DTD allows those submitting runs to specify the collections actually used in resolving the topics. Thus it permits users to submit runs for only a subset of the collections, and in principle such runs could be scored without counting the ignored collections.

5 Pooling and Assessment

Having set up the heterogeneous collection with tasks and topics, next steps include the actual submission of runs. The once submitted runs are the basis for a *pooling* procedure to extract the set of relevant elements for each query and task. This step can also provide us with new insights whether the pooling procedure can be applied to a heterogeneous scenario or if there is the need for suitable adaptations.

We plan to use the XRai system for relevance assessments based on the pooled elements. Part of the motivation in the restructuring of the collections so that each record or document was a separate file was to be able to use XRai.

6 Conclusion and Outlook

In this year's heterogeneous track, we managed to set up a collection whose subcollections are heterogeneous w.r.t. syntax, semantics and document genre. We also set up a number of test topics for evaluation. we therefore laid the foundations for a new heterogeneous track which should then concentrate on submitting runs, create a pooled test set and provide relevance assessments which are in turn used to evaluate the submitted runs. Unfortunately, only a single group (Berkeley) submitted runs for the track, and therefore it was decided not to perform the relevance assessments, since the outcomes would not contain enough diversity to be interesting. We hope that if the Heterogeneous track continues next year, that more groups will want to participate in both submitting runs and helping to perform relevance assessment.

Acknowledgement

Special thanks go to Miro Lehtonen for providing us with the IDEAlliance collection, and to Saadia Malik and Benjamin Piwowarski for technical support.

A Topic DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY lt      "&#60;">
<!ENTITY gt      ">">
<!ENTITY amp     "&#38;">

<!ELEMENT inex_het_topic
  (title,castitle?,description,narrative,ontopic_keywords,scope?)>
<!ATTLIST inex_het_topic
  topic_id    CDATA    #REQUIRED
>

<!ELEMENT title (#PCDATA)>
<!ELEMENT castitle (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT narrative (#PCDATA)>
<!ELEMENT ontopic_keywords (#PCDATA)>
<!ELEMENT scope (collection+) >
<!ELEMENT collection (#PCDATA) >
```

B Run Submission DTD

```
<!ENTITY % collection-ids "berkeley | bibdbpub | compuscience | dblp | hcibib |
                             qmulcsdbpub | ieee | zdnetart | zdnetcom | wikipedia |
                             lp | idea_eml | idea_xml1 | idea_xml2 | idea_xmle |
                             idea_xtech">
<!ELEMENT inex-het-submission (topic-fields, description, collections, topic+)>
<!ATTLIST inex-het-submission
  participant-id CDATA #REQUIRED
  run-id CDATA #REQUIRED
  task (CO | CAS1 | CAS2 | RS) #REQUIRED
  query (automatic | manual) #REQUIRED
>
<!ELEMENT topic-fields EMPTY>
<!ATTLIST topic-fields
  title (yes|no) #REQUIRED
  castitle (yes|no) #REQUIRED
  description (yes|no) #REQUIRED
  narrative (yes|no) #REQUIRED
  ontopic_keywords (yes|no) #REQUIRED
>
<!ELEMENT description (#PCDATA)>
<!ELEMENT topic (result*|collections)>
<!ATTLIST topic topic-id CDATA #REQUIRED >
<!ELEMENT collections (collection*)>
<!ELEMENT collection (rank?, rsv? )>
<!ATTLIST collection collectionid (%collection-ids;) #REQUIRED>
<!ELEMENT result (file, path, rank?, rsv?)>
<!ATTLIST result collectionid (%collection-ids;) #IMPLIED>
<!ELEMENT file (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT rank (#PCDATA)>
<!ELEMENT rsv (#PCDATA)>
```

Probabilistic Retrieval approaches for Thorough and Heterogeneous XML Retrieval

Ray R. Larson

School of Information
University of California, Berkeley
Berkeley, California, USA, 94720-4600
ray@ischool.berkeley.edu

Abstract. For this year’s INEX UC Berkeley focused on the Heterogeneous track, and submitted only two runs for the Adhoc tasks, a “thorough” run and a “BestInContext” run. For these runs we used the TREC2 probabilistic model with blind feedback. The Thorough run was ranked 21st out of the 106 runs submitted for the Thorough task. The majority of our effort for INEX was on the Heterogeneous (Het) track where we submitted three collection ranking runs and two content-only runs. As the only group to actually submit runs for the Het track, we are guaranteed both the highest, and lowest, effectiveness scores for each task. However, because it was deemed pointless to conduct the actual relevance assessments on the submissions of a single system, we do not know the exact values of these results.

1 Introduction

In this paper we will first discuss the algorithms and fusion operators used in our official INEX 2006 Adhoc and Heterogenous (Het) track runs. Then we will look at how these algorithms and operators were used in the various submissions for these tracks, and finally we will examine the results, at least for the Adhoc thorough task and discuss possible problems in implementation, and directions for future research.

2 The Retrieval Algorithms and Fusion Operators

This section describes the probabilistic retrieval algorithms used for both the Adhoc and Het track in INEX this year. These are the same algorithms that we have used in previous years for INEX, and also include the addition of a blind relevance feedback method used in combination with the TREC2 algorithm. In addition we will discuss the methods used to combine the results of searches of different XML components in the collections. The algorithms and combination methods are implemented as part of the Cheshire II XML/SGML search engine [15, 14, 12] which also supports a number of other algorithms for distributed search and operators for merging result lists from ranked or Boolean sub-queries.

2.1 TREC2 Logistic Regression Algorithm

Once again the principle algorithm used for our INEX runs is based on the *Logistic Regression* (LR) algorithm originally developed at Berkeley by Cooper, et al. [8]. The version that we used for Adhoc tasks was the Cheshire II implementation of the “TREC2” [6, 5] that provided good Thorough retrieval performance in the INEX 2005 evaluation [15]. As originally formulated, the LR model of probabilistic IR attempts to estimate the probability of relevance for each document based on a set of statistics about a document collection and a set of queries in combination with a set of weighting coefficients for those statistics. The statistics to be used and the values of the coefficients are obtained from regression analysis of a sample of a collection (or similar test collection) for some set of queries where relevance and non-relevance has been determined. More formally, given a particular query and a particular document in a collection $P(R | Q, D)$ 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 derived from the query and database, such that:

$$\begin{aligned} \log O(R|C, Q) &= \log \frac{p(R|C, Q)}{1 - p(R|C, Q)} = \log \frac{p(R|C, Q)}{p(\bar{R}|C, Q)} \\ &= c_0 + c_1 * \frac{1}{\sqrt{|Q_c| + 1}} \sum_{i=1}^{|Q_c|} \frac{qt f_i}{ql + 35} \\ &+ c_2 * \frac{1}{\sqrt{|Q_c| + 1}} \sum_{i=1}^{|Q_c|} \log \frac{tf_i}{cl + 80} \\ &- c_3 * \frac{1}{\sqrt{|Q_c| + 1}} \sum_{i=1}^{|Q_c|} \log \frac{ct f_i}{N_t} \\ &+ c_4 * |Q_c| \end{aligned}$$

where C denotes a document component and Q a query, R is a relevance variable, and

$p(R|C, Q)$ is the probability that document component C is relevant to query Q ,

$p(\bar{R}|C, Q)$ the probability that document component C is not relevant to query Q , (which is $1.0 - p(R|C, Q)$)

$|Q_c|$ is the number of matching terms between a document component and a query,

$qt f_i$ is the within-query frequency of the i th matching term,

tf_i is the within-document frequency of the i th matching term,

$ct f_i$ is the occurrence frequency in a collection of the i th matching term,

ql is query length (i.e., number of terms in a query like $|Q|$ for non-feedback situations),

cl is component length (i.e., number of terms in a component), and N_t is collection length (i.e., number of terms in a test collection). c_k are the k coefficients obtained through the regression analysis.

Assuming that stopwords are removed during index creation, then ql , cl , and N_t are the query length, document length, and collection length, respectively. If the query terms are re-weighted (in feedback, for example), then qtf_i is no longer the original term frequency, but the new weight, and ql is the sum of the new weight values for the query terms. Note that, unlike the document and collection lengths, query length is the “optimized” relative frequency without first taking the log over the matching terms.

The coefficients were determined by fitting the logistic regression model specified in $\log O(R|C, Q)$ to TREC training data using a statistical software package. The coefficients, c_k , used for our official runs are the same as those described by Chen[3]. These were: $c_0 = -3.51$, $c_1 = 37.4$, $c_2 = 0.330$, $c_3 = 0.1937$ and $c_4 = 0.0929$. Further details on the TREC2 version of the Logistic Regression algorithm may be found in Cooper et al. [6].

2.2 Blind Relevance feedback

It is well known that blind (also called pseudo) relevance feedback can substantially improve retrieval effectiveness in tasks such as TREC and CLEF. (See for example the papers of the groups who participated in the Ad Hoc tasks in TREC-7 (Voorhees and Harman 1998)[18] and TREC-8 (Voorhees and Harman 1999)[19].)

Blind relevance feedback is typically performed in two stages. First, an initial search using the original queries is performed, after which a number of terms are selected from the top-ranked documents (which are presumed to be relevant). The selected terms are weighted and then merged with the initial query to formulate a new query. Finally the reweighted and expanded query is run against the same collection to produce a final ranked list of documents. It was a simple extension to adapt these document-level algorithms to document components for INEX.

The TREC2 algorithm has been combined with a blind feedback method developed by Aitao Chen for cross-language retrieval in CLEF. Chen[4] presents a technique for incorporating blind relevance feedback into the logistic regression-based document ranking framework. Several factors are important in using blind relevance feedback. These are: determining the number of top ranked documents that will be presumed relevant and from which new terms will be extracted, how to rank the selected terms and determining the number of terms that should be selected, how to assign weights to the selected terms. Many techniques have been used for deciding the number of terms to be selected, the number of top-ranked documents from which to extract terms, and ranking the terms. Harman [11] provides a survey of relevance feedback techniques that have been used.

Lacking comparable data from previous years, we adopted some rather arbitrary parameters for these options for INEX 2006. We used top 10 ranked

components from the initial search of each component type, and enhanced and reweighted the query terms using term relevance weights derived from well-known Robertson and Sparck Jones[17] relevance weights, as described by Chen and Gey[5]. The top 10 terms that occurred in the (presumed) relevant top 10 documents, that were not already in the query were added for the feedback search.

2.3 TREC3 Logistic Regression Algorithm

In addition to the TREC2 algorithm described above, we also used the TREC3 algorithm in some of our Het track runs. This algorithm has been used repeatedly in our INEX work, and described many times, but we include it below for ease of comparison. The full equation describing the ‘‘TREC3’’ LR algorithm used in these experiments is:

$$\begin{aligned}
 \log O(R | Q, C) = & \\
 & b_0 + \left(b_1 \cdot \left(\frac{1}{|Q_c|} \sum_{j=1}^{|Q_c|} \log qtf_j \right) \right) \\
 & + \left(b_2 \cdot \sqrt{|Q|} \right) \\
 & + \left(b_3 \cdot \left(\frac{1}{|Q_c|} \sum_{j=1}^{|Q_c|} \log tf_j \right) \right) \\
 & + \left(b_4 \cdot \sqrt{cl} \right) \\
 & + \left(b_5 \cdot \left(\frac{1}{|Q_c|} \sum_{j=1}^{|Q_c|} \log \frac{N - n_{t_j}}{n_{t_j}} \right) \right) \\
 & + (b_6 \cdot \log |Q_d|)
 \end{aligned} \tag{1}$$

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 ,
 n_{t_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 through the regression analysis.

This equation, used in estimating the probability of relevance for some of the Het runs in this research, is essentially the same as that used in [7]. The b_i coefficients in the original version of this algorithm were estimated using relevance

judgements and statistics from the TREC/TIPSTER test collection. In INEX 2005 we did not use the original or “Base” version, but instead used a version where the coefficients for each of the major document components were estimated separately and combined through component fusion. This year, lacking relevance data from Wikipedia for training, we used the base version again. The coefficients for the Base version were $b_0 = -3.70$, $b_1 = 1.269$, $b_2 = -0.310$, $b_3 = 0.679$, $b_4 = -0.0674$, $b_5 = 0.223$ and $b_6 = 2.01$.

2.4 CORI Collection ranking algorithm

The resource selection task in the Heterogeneous track is basically the same as the collection selection task in distributed IR. For this task we drew on our previously experiments with distributed search and collection ranking [12, 13], where we used the above “TREC3” algorithm for collection selection and compared it with other reported distributed search results.

The collection selection task attempts to discover which distributed databases are likely to be the best places for the user to begin a search. This problem, distributed information retrieval, has been an area of active research interest for many years. Distributed IR presents three central research problems:

1. How to select appropriate databases or collections for search from a large number of distributed databases;
2. How to perform parallel or sequential distributed search over the selected databases, possibly using different query structures or search formulations, in a networked environment where not all resources are always available; and
3. How to merge results from the different search engines and collections, with differing record contents and structures (sometimes referred to as the collection fusion problem).

Each of these research problems presents a number of challenges that must be addressed to provide effective and efficient solutions to the overall problem of distributed information retrieval. Some of the best known work in this area has been Gravano, et al’s work on GLOSS [10, 9] and Callan, et al’s [2, 20, 1] application of inference networks to distributed IR. One of the best performing collection selection algorithms developed by Callan was the “CORI” algorithm. This algorithm was adapted for the Cheshire II system, and used for some of our Resource Selection runs for the Het track this year. The CORI algorithm defines a belief value for each query term using a form of tfidf ranking for each term and collection:

$$T = \frac{df}{df + 50 + 150 \cdot cw/\overline{cw}}$$

$$I = \frac{\log(\frac{|DB|+0.5}{cf})}{\log(|DB| + 1.0)}$$

$$p(r_k|db_i) = 0.4 + 0.6 \cdot T \cdot I$$

Where:

df is the number of documents containing terms r_k ,
 cf is the number of databases or collections containing r_k ,
 $|DB|$ is the number of databases or collections being ranked,
 cw is the number of terms in database or collection db_i ,
 \overline{cw} is the average cw of the collections being ranked, and
 $p(r_k|db_i)$ is the belief value in collection db_i due to observing term r_k

These belief values are summed over all of the query terms to provide the collection ranking value.

2.5 Result Combination Operators

As we have also reported previously, the Cheshire II system used in this evaluation provides a number of operators to combine the intermediate results of a search from different components or indexes. With these operators we have available an entire spectrum of combination methods ranging from strict Boolean operations to fuzzy Boolean and normalized score combinations for probabilistic and Boolean results. These operators are the means available for performing fusion operations between the results for different retrieval algorithms and the search results from different different components of a document. For Heterogeneous search we used a variant of the combination operators, where MINMAX normalization across the probability of relevance for each entry in results from each sub-collection was calculated and the final result ranking was based on these normalized scores.

In addition, for the Adhoc Thorough runs we used a merge/reweighting operator based on the “Pivot” method described by Mass and Mandelbrod[16] to combine the results for each type of document component considered. In our case the new probability of relevance for a component is a weighted combination of the initial estimate probability of relevance for the component and the probability of relevance for the entire article for the same query terms. Formally this is:

$$P(R | Q, C_{new}) = (X * P(R | Q, C_{comp})) + ((1 - X) * P(R | Q, C_{art})) \quad (2)$$

Where X is a pivot value between 0 and 1, and $P(R | Q, C_{new})$, $P(R | Q, C_{comp})$ and $P(R | Q, C_{art})$ are the new weight, the original component weight, and article weight for a given query. Although we found that a pivot value of 0.54 was most effective for INEX04 data and measures, we adopted the “neutral” pivot value of 0.5 for all of our 2006 adhoc runs, given the uncertainties of how this approach would fare with the new database.

3 Database and Indexing Issues

Use of the Wikipedia, without a pre-defined DTD or XML Schema led to a very difficult indexing process, primarily due to the requirement that the data in

Cheshire II databases must correspond to a DTD or Schema. To avoid having to reject and remove thousands of records that did not correspond to a basic DTD provided by ?? we ran indexing until some new undefined tag or attribute was encountered, and then updated the dtd and restarted indexing from the beginning. This meant that literally hundreds of indexing runs needed to be performed in order to come up with a database that corresponded to the manually elaborated DTD.

The final result encoded into the DTD that we needed a large number of nonsensical tags and attributes (for example the tag “wikipedialink” has over 2000 defined attributes, obviously due to a parsing error in the original conversion that converted each word of entire sentences into attributes of the tag). In addition, hundreds of tags were used only once in the collection including tags such as “contin.c_evalcontinuousatt_28” and “akademie”, “pokemon”, and “zrubek”.

3.1 Indexing the INEX 2006 Database

All indexing in the Cheshire II system is controlled by an XML/SGML Configuration file which describes the database to be created. This configuration file is subsequently used in search processing to control the mapping of search command index names (or Z39.50 numeric attributes representing particular types of bibliographic data) to the physical index files used and also to associated component indexes with particular components and documents. This configuration file also includes the index-specific definitions for the Logistic Regression coefficients (when not defined, these default to the “Base” coefficients mentioned above).

Name	Description	Contents	Vector?
docno	doc ID number	//name@id	No
title	Article Title	//name	No
topic	Entire Article	//article	no
topicshort	Selected Content	//fm/tig/atl //abs //kwd //st	Yes
xtnames	Template names	//template@name	no
figure	Figures	//figure	No
table	Tables	//table	No
caption	Image Captions	//caption	Yes
alltitles	All Titles	//title	Yes
links	Link Anchors	//collectionlink //weblink //wikipedialink	No

Table 1. Wikipedia Article-Level Indexes for INEX 2006

Table 1 lists the document-level (/article) indexes created for the INEX database and the document elements from which the contents of those indexes were extracted.

Name	Description	Contents
COMPONENT_SECTION	Sections	//section
COMPONENT_PARAS	Paragraphs	//p //blockquote //indentation1 //indentation2 //indentation3
COMPONENT_FIG	Figures	//figure

Table 2. Wikipedia Components for INEX 2006

As noted above the Cheshire system permits parts of the document subtree to be treated as separate documents with their own separate indexes. Tables 2 & 3 describe the XML components created for INEX and the component-level indexes that were created for them.

Table 2 shows the components and the path used to define them. The first, COMPONENT_SECTION, component consists of each identified section in all of the documents, permitting each individual section of a article to be retrieved separately. Similarly, each of the COMPONENT_PARAS and COMPONENT_FIG components, respectively, treat each paragraph (with all of the alternative paragraph elements shown in Table 2), and figure (<figure> ... </figure>) as individual documents that can be retrieved separately from the entire document.

Component or Index Name	Description	Contents	Vector?
COMPONENT_SECTION			
sec_title	Section Title	//section/title	Yes
sec_words	Section Words	*†	Yes
COMPONENT_PARAS			
para_words	Paragraph Words	*†	Yes
COMPONENT_FIG			
fig_caption	Figure Caption	//figure/caption	No

Table 3. Wikipedia Component Indexes for INEX 2006†Includes all subelements of section or paragraph elements.

Table 3 describes the XML component indexes created for the components described in Table 2. These indexes make individual sections (COMPONENT_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. Individual paragraphs (COMPONENT_PARAS) are searchable by any of the terms in the paragraph, also with proximity searching. Individual figures (COMPONENT_FIG) are indexed by their captions.

Few of these indexes and components were used during Berkeley’s simple runs of the 2006 INEX Adhoc topics. The two official submitted Adhoc runs and scripts used in INEX are described in the next section.

3.2 Heterogeneous Indexing

The Heterogeneous track data includes 15 additional collections and subcollections beyond Wikipedia. The collections are described in Table 4. The Wikipedia

collection is the largest, and for the Heterogeneous track we used the same indexes as for the the Adhoc track.

Collection	Size	SubColl.	Documents	Elements	Mean Elements per Document
Berkeley	52M		12800	1182062	92.3
bibdb Duisburg	14M		3465	36652	10.6
CompuScience	993M		250986	6803978	27.1
DBLP	2.0G		501102	4509918	9.0
hcibib	107M		26390	282112	10.7
IEEE (2.2)	764M		16820	11394362	677.4
IDEAlliance	58M	eml	156	66591	426.9
		xml1	301	45559	151.4
		xml2	264	58367	221.1
		xmle	193	32901	170.5
		xtech	71	14183	199.8
Lonely Planet	16M		462	203270	440.0
qmulcsdbpub	8.8M		2024	23435	11.6
Wikipedia	4.9G		659385	1193488685	1810.0
ZDNet	339M	Articles	4704	242753	51.6
		Comments	91590	1433429	15.7
Totals	9.25G		1570713	1219818257	776.6

Table 4. Components of the heterogeneous collection. *Element counts estimated for large collections.*

For the rest of the collections we created indexes that accessed similar elements in each individual collection. These typically included indexes for the classes of elements listed in Table 5. Naturally each of these classes of elements had differing structures and names in the different collections. For example, “title” elements in the Berkeley collection mapped to the XML structures “//US-MARC/VarFlds/VarDFlds/Fld24*/a—b—n—p—f—l” (where “*” is a wildcard matching anything up to the end of the tag name and “—”, while in the IEEE collection they mapped to “//fm/tig/at1” and in the CompuScience collection simply to “//title”.

Name	Description	Vector?
docno	doc ID number	No
pauthor	Author(s)	No
title	Article Title	No
topic	Entire Article	no
date	Date or Year	no
journal	Journal title	No
kwd	Keywords or Subjects	No
Abstract	Abstracts	Yes
alltitles	All Titles	Yes

Table 5. Document Indexes for all Heterogeneous collections for INEX 2006

Indexes were created for each of the mapped sets of elements shown in Table 5, and during search the common class names were used to specify the desired set of elements across the different collections.

4 INEX 2006 Official Adhoc and Heterogeneous Runs

4.1 Adhoc Runs

Berkeley submitted a total of 2 retrieval runs for the INEX 2006 adhoc tasks, one for the Thorough task and one for the BestInContext task.

The primary task that we focussed on was the CO.Thorough task. For these tasks some automatic expansion of items in the XPath to the root of the document was used. The submitted run used the TREC2 algorithm on the full articles and figure components, and TREC2 with Blind Feedback for the section components and paragraph components. Weights for each of the retrieved component elements were adjusted using the pivot method described above against the score for the article obtained using the same query terms. The average EP/GR score for the Berkeley BASE_T2FB run was 0.0252, which was ranked 21 out of the 106 submissions reported on the official results pages.

BestInContext Runs We also submitted a single run for the BestInContext task. This run (BASE_T2FB_BC) was prepared by keeping only the single highest ranked element for each of the top-ranked 1500 documents from the Thorough resultset (using the full Thorough resultset, including all results). This run obtained ranked in the middle range of the officially reported results (depending on the particular measure)

4.2 Heterogeneous Runs

Three runs were submitted for the Resource Selection task, and 2 for the Content-Only task. The Resource selection runs used the TREC2, TREC3, and CORI algorithms, respectively, with no blind feedback. The two Content-Only runs used the TREC2 and TREC3 algorithms, also with no blind feedback.

Since Berkeley was the only group to submit Het track runs, it was decided not to go to the effort of evaluation with such a limited pool, so we do not have any figures on the actual or relative performance of these different techniques for the Heterogeneous track.

5 Conclusions and Future Directions

Our participation in INEX this year was severely constrained by other competing IR work and conference committee work for a number of other conferences. In addition, due to the irregularities of the Wikipedia data we ended up spending much more time than expected in creating a valid DTD that could be used in

indexing. In addition a lot of time was spent setting up collections and indexing the Heterogeneous collections. This left little time to actually test and evaluate different run strategies and to adapt previous work to the new collections.

We hope to be able to try some other methods, and new techniques for INEX 2007.

References

1. J. Callan. Distributed information retrieval. In W. B. Croft, editor, *Advances in Information Retrieval: Recent research from the Center for Intelligent Information Retrieval*, chapter 5, pages 127–150. Kluwer, Boston, 2000.
2. J. P. Callan, Z. Lu, and W. B. Croft. Searching Distributed Collections with Inference Networks . In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington, 1995. ACM Press.
3. A. Chen. Multilingual information retrieval using english and chinese queries. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum, CLEF-2001, Darmstadt, Germany, September 2001*, pages 44–58. Springer Computer Science Series LNCS 2406, 2002.
4. A. Chen. *Cross-Language Retrieval Experiments at CLEF 2002*, pages 28–48. Springer (LNCS #2785), 2003.
5. A. Chen and F. C. Gey. Multilingual information retrieval using machine translation, relevance feedback and decompounding. *Information Retrieval*, 7:149–182, 2004.
6. W. S. Cooper, A. Chen, and F. C. Gey. Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression. In *Text REtrieval Conference (TREC-2)*, pages 57–66, 1994.
7. 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.
8. 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.
9. L. Gravano and H. García-Molina. Generalizing GLOSS to vector-space databases and broker hierarchies. In *International Conference on Very Large Databases, VLDB*, pages 78–89, 1995.
10. L. Gravano, H. García-Molina, and A. Tomasic. GLOSS: text-source discovery over the Internet. *ACM Transactions on Database Systems*, 24(2):229–264, 1999.
11. D. Harman. Relevance feedback and other query modification techniques. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 241–263. Prentice Hall, 1992.
12. 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.

13. R. R. Larson. Distributed IR for digital libraries. In *Research and Advanced Technology for Digital Libraries (ECDL 2003)*, pages 487–498. Springer (LNCS #2769), 2003.
14. R. R. Larson. A fusion approach to XML structured document retrieval. *Information Retrieval*, 8:601–629, 2005.
15. R. R. Larson. Probabilistic retrieval, component fusion and blind feedback for XML retrieval. In *INEX 2005*, pages 225–239. Springer (Lecture Notes in Computer Science, LNCS 3977), 2006.
16. Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement for xml retrieval. In *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX2004*, pages 73–84. Springer (LNCS #3493), 2005.
17. S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, pages 129–146, May–June 1976.
18. E. Voorhees and D. Harman, editors. *The Seventh Text Retrieval Conference (TREC-7)*. NIST, 1998.
19. E. Voorhees and D. Harman, editors. *The Eighth Text Retrieval Conference (TREC-8)*. NIST, 1999.
20. J. Xu and J. Callan. Effective retrieval with distributed collections. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–120, 1998.

Social Media Retrieval using Image Features and Structured Text

D.N.F. Awang Iskandar, Jovan Pehcevski, James A. Thom, and S. M. M. Tahaghoghi

School of Computer Science and Information Technology, RMIT University
Melbourne, Australia
{dayang, jovanp, jat, saied}@cs.rmit.edu.au

Abstract. The eXtensible Markup Language (XML) is a markup language that is used to annotate and provide the structural information of a document. Since an XML document could contain various media, the retrieval process is challenging. This paper reports on the RMIT University group's participation in the Ad hoc and MM tracks of INEX 2006. Using the Wikipedia collection, we show that Okapi BM25 probabilistic model has proven to be the best information retrieval approach for the Thorough task of the Ad hoc Track. For the Multimedia track, combining image and text retrieval results reveals a better retrieval performances.

Key words: Content-based image retrieval, text-based information retrieval, social media, linear combination of evidence

1 Introduction

A structured document could contain text, images, audios and videos. The eXtensible Markup Language (XML) is a markup language developed by the World Wide Web Consortium for documents containing structured information. This markup language describes the content and also indicates the role that this content plays. For example, the content in a section heading has a different meaning than the content in a footnote, which in turn means something different than the content in a figure caption or the content in a database table.

Retrieving the desired information from an XML document involves the retrieval of the XML elements. This is not a trivial task as it may involve retrieving text and other multimedia elements. Existing research on multimedia information retrieval from XML document collection has shown to be challenging [3, 9, 10]. Questions that are yet to be answered are:

- How to relate and combine the relevance of different multimedia fragments?
- How to combine the retrieval status values (RSVs) of the retrieved XML elements containing the information need with the relevance multimedia elements?
- What are the effective techniques to combine these RSVs?
- Which RSVs should be given more weight in determining the retrieval result?

The INitiative for the Evaluation of XML Retrieval (INEX) provides a platform for participants to evaluate the effectiveness of their XML retrieval techniques using uniform scoring procedures, and a forum to compare results. Of the nine tracks at INEX

2006, this paper presents the RMIT university group's participation in the Ad hoc and Multimedia (MM) tracks. This paper begins with a discussion on the full-text information retrieval that is use for both tracks, followed by the performance results of the Ad hoc track, then the approach taken for the MM track along with the performance results. However, this paper emphasizes more on the MM track participation.

The objective of the INEX 2006 MM track is to exploit the XML structure that provides a logical level at which multimedia objects are connected and to improve the retrieval performance of an XML-driven multimedia information retrieval system.¹ Besides RMIT University, three other groups participated in the multimedia track — Queensland University of Technology (QUTAU), University of Twente (UTWENTE) and Institut de Recherche en Informatique de Toulouse (IRIT).

The aim of the RMIT University group in participating in the INEX 2006 MM track was to explore and analyse methods for combining evidence from content-based image retrieval (CBIR) with full-text information retrieval (IR). In this paper, we describe a fusion system that combines evidence and ranks the query results based on text and image similarity. The fusion system consists of two subsystems: the GNU Image Finding Tool (GIFT), and the full-text IR system (Zettair). A technique for linear combination of evidence is used to merge the relevance scores from the two subsystems.

The retrieval strategy has been evaluated using Wikipedia, a social media collection that is an online encyclopaedia. Social media describes the online tools and platforms that people use to share opinions, insights, experiences, and perspectives with each other. Social media can take many different forms, including text, images, audio, and video. Popular social mediums include blogs, message boards, podcasts, wikis, and vlogs.²

The remainder of this paper is organised as follows. Section 2 describes the text retrieval approach used for the Ad hoc and MM track. In Section 3, we present the multimedia topics and their corresponding relevance judgements. We describe our approach to retrieve the XML articles and the associated images based on these multimedia topics in Section 4. In Section 5, we present results obtained from our experiments on the INEX 2006 MM track. We conclude in Section 6 with a discussion of our findings and suggestions for future work.

2 Full-Text Information Retrieval

In this section, we introduce Zettair as our choice of a full-text IR system. We describe the three similarity measures implemented in Zettair, and show performance results on the Thorough task of the INEX 2006 Ad hoc track.

2.1 The Zettair Search Engine

Zettair is a compact and fast text search engine developed by the Search Engine Group at RMIT University.³ Zettair supports on-the-fly indexing and retrieval of large textual document collections. To process the queries for the INEX 2006 Ad hoc and MM

¹ INEX 2006 Multimedia Track Guidelines

² <http://en.wikipedia.org/wiki/Wiki>

³ <http://www.seg.rmit.edu.au/zettair/> (Zettair was formerly known as Lucy).

track, we first obtained the document content by extracting the plain document text (and by completely removing all the XML tags). We then indexed these documents using fast and efficient inverted index structure as implemented in many modern search engines [11]. To retrieve the likely relevant XML documents to an INEX topic, three similarity measures are implemented and their retrieval performances are compared relative to each other.

2.2 Similarity Measures

The *similarity* of a document to a query, denoted as $S_{q,d}$, indicates how closely the content of the document matches the query.

To calculate the query-document similarity, statistical information about the distribution of the query terms — within both the document and the collection as a whole — is often necessary. These term statistics are subsequently utilised by the similarity measure. Following the notation and definitions of Zobel and Moffat [13], we define the basic term statistics as:

- q , a query;
- t , a query term;
- d , a document;
- $N_{\mathcal{D}}$, the number of all the documents in the collection;
- For each term t :
 - $f_{d,t}$, the frequency of t in the document d ;
 - $N_{\mathcal{D}_t}$, the number of documents containing the term t (irrespective of the term frequency in each document); and
 - $f_{q,t}$, the frequency of t in query q .
- For each document d :
 - $f_d = |d|$, the document length approximation.
- For the query q :
 - $f_q = |q|$, the query length.

We also denote the following sets:

- \mathcal{D} , the set of all the documents in the collection;
- \mathcal{D}_t , the set of documents containing term t ;
- \mathcal{T}_d , the set of distinct terms in the document d ;
- \mathcal{T}_q , the set of distinct terms in the query, and $\mathcal{T}_{q,d} = \mathcal{T}_q \cap \mathcal{T}_d$.

We now describe the three similarity measures implemented in Zettair, which respectively follow the three major models to information retrieval: the vector-space model, the probabilistic model, and the language model.

Vector-Space Model In the vector-space model, both the document and the query are representations of n -dimensional vectors, where n is the number of distinct terms observed in the document collection. The best-known technique for computing similarity under the vector-space model is the cosine measure, where the similarity between a document and the query is computed as the cosine of the angle between their vectors.

Zettair uses pivoted cosine document length normalisation [6] to compute the query-document similarity under the vector-space model:

$$S_{q,d} = \frac{1}{W_D \times W_q} \times \sum_{t \in \mathcal{T}_{q,d}} (1 + \log_e f_{d,t}) \times \log_e \left(1 + \frac{N_{\mathcal{D}}}{N_{\mathcal{D}_t}} \right) \quad (1)$$

In the above equation, $W_D = \left((1.0 - s) + s \times \frac{W_d}{W_{AL}} \right)$ represents the pivoted document length normalisation, and W_q is the query length representation. The parameter s represents the *slope*, whereas W_d and W_{AL} represent the document length (usually taken as f_d) and the average document length (over all documents in \mathcal{D}), respectively. We use the standard value of 0.2 for the slope, which is shown to work well in traditional IR experiments [6].

Probabilistic Model Probabilistic models of information retrieval are based on the principle that documents should be ranked by decreasing probability of their relevance to the expressed information need. Zettair uses the Okapi BM25 probabilistic model developed by Sparck Jones et al. [7], which has proved highly successful in a wide range of experiments:

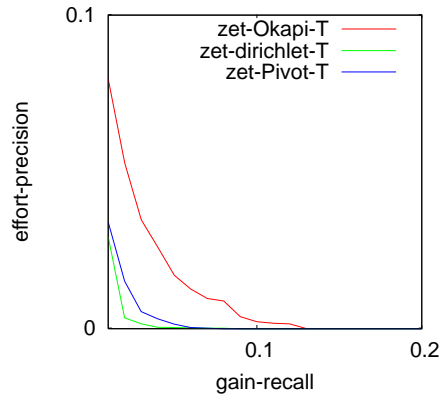
$$S_{q,d} = \sum_{t \in \mathcal{T}_{q,d}} w_t \times \frac{(k_1 + 1) f_{d,t}}{K + f_{d,t}} \times \frac{(k_3 + 1) f_{q,t}}{k_3 + f_{q,t}} \quad (2)$$

where $w_t = \log_e \left(\frac{N_{\mathcal{D}} - N_{\mathcal{D}_t} + 0.5}{N_{\mathcal{D}_t} + 0.5} \right)$ is a representation of inverse document frequency, $K = k_1 \times \left[(1 - b) + \frac{b \cdot W_d}{W_{AL}} \right]$, and k_1 , b and k_3 are constants, in the range 1.2 to 1.5 (we use 1.2), 0.6 to 0.75 (we use 0.75), and 1,000,000 (effectively infinite), respectively. W_d and W_{AL} represent the document length and the average document length.

Language Model Language models are probability distributions that aim to capture the statistical regularities of natural language use. In information retrieval, language modelling involves estimating the likelihood that both the document and the query could have been generated by the same language model. Zettair uses a query likelihood approach with Dirichlet smoothing [12]:

$$S_{q,d} = f_q \times \log \lambda_d + \sum_{t \in \mathcal{T}_{q,d}} \log \left(\frac{N_{\mathcal{D}} \times f_{d,t}}{\mu \times N_{\mathcal{D}_t}} + 1 \right) \quad (3)$$

where μ is a smoothing parameter (we use the value of 2,000), while λ_d is calculated as: $\lambda_d = \mu / (\mu + f_d)$.



Run Type	Similarity Measure	MAep
zet-Okapi-T	Okapi	0.0058
zet-Dirichlet-T	Dirichlet	0.0052
zet-Pivot-T	Pivoted Cosine	0.0047

Fig. 1. Retrieval performance of the three similarity measures implemented in Zettair on the Thorough task of the INEX 2006 Ad hoc track.

2.3 Performance Results

In this subsection, we compare the performances of the three similarity measures implemented in Zettair on the Thorough task of the INEX 2006 Ad hoc track.⁴

The official measures of retrieval effectiveness for the INEX 2006 Thorough task are ep/gr and $MAep$ of the XCG metrics family [4]. The ep/gr graphs provide a detailed view of the run's performance at various gain-recall points. The $MAep$ measure provides a single-valued score for the overall run performance. This evaluation measure was also used for the MMFragments task evaluation of the INEX 2006 MM track. The measures make use of the *Specificity* relevance dimension, which is measured automatically on a continuous scale with values in the interval $[0, 1]$. A relevance value of 1 represents a fully specific component (that contains only relevant information), whereas a relevance value of 0 represents a non-relevant component (that contains no relevant information). Values of *Specificity* were derived on the basis of the ratio of relevant to both relevant and non-relevant text, as highlighted by the assessor.

Figure 1 shows the performance results obtained for the three similarity measures using both the $MAep$ scores and the ep/gr graphs. We observe that Okapi produced the best $MAep$ score among the three similarity measures, substantially outperforming the other two similarity measures. This performance difference is especially reflected on the ep/gr graphs. Of the other two measures, Dirichlet seems to perform better than

⁴ Similar relative performance differences between the three similarity measures were also observed on the Focussed task of the INEX 2006 Ad hoc track.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_mm_topic topic_id="8" ct_no="18"
task="MMImages">
<title>Images of bees with flowers.</title>
<castitle>//article[about(., src:60248)
and about(., bee)
and about(.,concept:animal)]
</castitle>
<description> Find images depicting a bee or
bees with flowers. </description>
<narrative>
Bees play an important role in pollinating flowering plants,
and are the major type of pollinators in ecosystems that contain
flowering plants. The flower's nectar is the primary source for energy,
and the pollen is primarily for protein and other nutrients. We are
looking for pictures depicting a bee or bees with flowers. We are not
interested in pictures of a basketball coach "Clair Bee" and album cover
for the Bee Gee's Stayin' Alive.
</narrative>
</inex_mm_topic>

```




Fig. 2. Example of a MMImages query with image ID 60248, a bee and a flower (original in colour)

the Pivoted Cosine measure.⁵ Interestingly, the ep/gp graphs generated on the *article-level* Fetch and Browse retrieval task of the INEX 2005 Ad hoc track show similar relative performance differences between the three similarity measures, even though different XML document collection (IEEE instead of Wikipedia) was used as part of the evaluation testbed [5]. However, it is also worth noting that the Pivoted Cosine similarity measure outperformed the other two measures on the *element-level* Fetch and Browse retrieval task of the INEX 2005 Ad hoc track.

We now describe the research activities we carried out for the INEX 2006 MM track. We start with a description of the INEX 2006 MM tasks, along with their associated topics and their corresponding relevance judgements.

3 Multimedia Tasks, Topics and Relevance Judgements

The INEX 2006 MM track topics were organised differently compared to the INEX 2005 MM topics. The INEX 2005 MM track topics are only based on the Multimedia Fragments (MMFragments) task. A new task was introduced in the INEX 2006 MM track, namely Multimedia Images (MMImages).

Since there are two tasks, the Wikipedia collection has been divided into two sub-collections: Wikipedia Ad Hoc XML collection (Wikipedia) — contains XML docu-

⁵ Due to some problems in running the local instance of EvalJ, at this stage we did not check for significance of the reported results.

ments and images and Wikipedia image collection (Wikipedia_IMG)— contains 170 000 royalty free images. The MMFragments task utilises the Wikipedia collection and the Wikipedia_IMG is used for the MMImages task.

3.1 Multimedia Images Task

In the MMImages task, the participants have to find relevant images in the articles based on the topic query. Hence, this task is basically using image retrieval techniques. Even though the target element is an image, the XML structure in the documents could be exploited to get to the relevant images. An Example of an MMImages topic is depicted in Figure 2.

Each XML document in the Wikipedia_IMG collection contain an image. Therefore, the MMImages task is a document retrieval task, the only results allowed there are full documents (that are articles) from the image XML collection. This mean the path of each of the results for this task are in the form of /article[1] — no document fragment are retrieved.

3.2 Multimedia Fragments Task

The objective of the MMFragments task is to find relevant XML fragments given an multimedia information need. Figure 3 illustrates a MMFragments topic. The target elements are ranked in relevance order and element overlapping is permitted.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_mm_topic topic_id="16" ct_no="12" task="MMfragments">
<title>Kiwi shoe polish</title>
<castitle>
//article[about(../history,kiwi shoe polish)]//image[about(., kiwi)]
</castitle>
<description>
Find images related to the Kiwi shoe polish product.
</description>
<narrative>Kiwi is the brand name of a shoe polish, first made in Australia
in 1906 and as of 2005 sold in almost 180 countries. Owned by the Sara Lee
Corporation since 1984, it is the dominant shoe polish in some countries,
including the United Kingdom and the United States, where it has about
two-thirds of the market. Find images related to the Kiwi shoe polish
product. We are not interested in the kiwi fruit.</narrative>
</inex_mm_topic>
```

Fig. 3. Example of a MMFragments query.

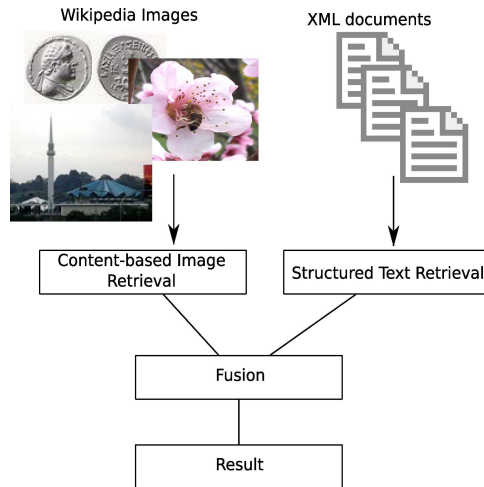


Fig. 4. Conceptual representation of the system (original in colour)

4 Our Approach

In this section, we describe the approach taken for the INEX 2006 MM track. We used two systems and fused the results from these systems to obtain the results for the multimedia queries. The overall structure of the system is depicted in Figure 4. Since the XML document structure serves as a semantic backbone for retrieval of the multimedia fragments, we use Zettair to retrieve the relevant articles. The GNU Image Finding Tool (GIFT),⁶ a content-based image retrieval system, is used to retrieve the results based on the visual features of the images.

For INEX 2006 MM track, we adopted the same approach as in INEX 2005 MM track [3]. The only different is that we used Zettair instead of the Hybrid XML Retrieval approach. With Zettair, we decided to use pivoted cosine similarity measure because it performed best among the three measures in INEX 2005 Ad hoc test collection (using the IEEE document collection) [5].

5 Content-Based Image Retrieval

The GNU Image Finding Tool was used to retrieve the similar images. The image features from the Wikipedia_IMG collection was extracted and indexed using an inverted file data structure.

Two main image features — colour and texture were extracted during the indexing process. GIFT uses the HSV (Hue-Saturation-Value) colour space for local and global colour features [8]. For extracting the image texture, a bank of circularly symmetric Gabor filters is used. GIFT evaluates and calculates the query image and the target

⁶ <http://www.gnu.org/software/gift/>

image feature similarity based on the data from the inverted file. The results of a query are presented to the user in the form of a ranked list.

For the multimedia topics, we presented the images listed in the source (`src`) element of the multimedia CAS query as the query image to GIFT. We used the default Classical IDF algorithm and set the search pruning option to 100%. This allows us to perform a complete feature evaluation for the query image, even though the query processing time is longer. For each query, we retrieved and ranked all the images in the `Wikipedia_IMG` collection. Referring to the multimedia topic presented earlier, the query image of Figure 2 is provided to GIFT.

6 Fusing and Ranking The Image and Text Retrieval

To fuse the two RSV lists into a single ranked result list R for the multimedia queries, we use a simple linear combination of evidence [1]:

$$R = \alpha \cdot S_T + (1 - \alpha) \cdot S_I$$

Here, α is a weighting parameter (determines the weight of GIFT versus Zettair retrieval), S_I represents the image RSV obtained from GIFT, and S_T is the RSV of the same image obtained from the Zettair.

To investigate the effect of giving certain biases to a system, we vary the α values between 0 to 1. When the value of α is set to 1, only the RSVs from Zettair are used. On the other hand, only the GIFT's RSVs are used when the value of α is set to 0. For the INEX 2006 MM track official runs, we submitted six runs with the α value set to 0.0, 0.5 and 1.0. The fusion RSVs of the image and structured text retrieval are then ranked in a descending order of similarity.

7 Experiments and Results

The experiment for the runs was conducted by varying the alpha values and the mixture of the multimedia topic elements. For each task, the experiments were divided into two types based on the INEX 2006 multimedia topic element that was automatically translated as an input query to Zettair:

1. *Title* run utilises the content of the title element; and
2. *Mix* run utilises the content of the title, castitle and description from each topic.

7.1 Evaluation Metrics

Two evaluation metrics were used to evaluate the submitted runs. The TREC evaluation metric was adopted to evaluate the MMImages task and the evaluation is based on the standard recall and precision retrieval performance measures. The following measures were also used:

- Precision at cut-off ($P@n$): Precision after n document fragments have been retrieved.

Table 1. Retrieval performances for the MMImages task: mean average precision (MAP) and bpref.

Run Type	α value	MAP	bpref
RMIT-MMI-Title-00	0.0	0.2281	0.2152
RMIT-MMI-Title-05	0.5	0.3174	0.2975
RMIT-MMI-Title-10	1.0	0.3153	0.2957
RMIT-MMI-Mix-00	0.0	0.2232	0.2071
RMIT-MMI-Mix-05	0.5	0.2808	0.2532
RMIT-MMI-Mix-10	1.0	0.2599	0.2328

- Mean Average Precision (MAP): The mean of the average precisions calculated for each topic. Average precision is the average of the precisions calculated at each natural recall level.
- bpref: It computes a preference relation of whether judged relevant documents are retrieved ahead of judged irrelevant documents. Thus, it is based on the relative ranks of judged documents only.
- Average interpolated precision at 11 standard recall levels (0%-100%).

7.2 Multimedia Image Task

Since the MMImages task results are articles, we are still able to fuse the image and text retrieval RSVs even though the title element was translated as the query. Based on Table 1, we observed that using the title element as the query produces better MAP and bpref performance compared to using a mixture of the multimedia topic elements for the MMImages task.

In the title runs, having the alpha value of 0.5 yielded the best MAP and bpref performance. This shows that combining image and text RSVs improves the retrieval performances.

Figure 5 illustrates the interpolated recall precision for all the runs. At 10% recall, the RMIT-MMI-Mix-05 run performed slightly better than RMIT-MMI-Title-05 and RMIT-MMI-Title-10. However, the RMIT-MMI-Title-05 and RMIT-MMI-Title-10 runs produces an overall interpolated recall precision. The precision values drop as the α value is decreased.

There was no visible difference in precision for run RMIT-MMI-Title-05, RMIT-MMI-Title-10 and RMIT-MMI-Mix-05 with P@1. With P@2 and P@3, combining evidence from image and text at the same weight ($\alpha = 0.5$) leads to similar performance as when α value is 1.0.

Comparing the performances between INEX 2005 and INEX 2006 MM track, we observed a similar trend in the precision values. Using the text retrieval system alone produces better precision value compared to using only the content-based image retrieval system.

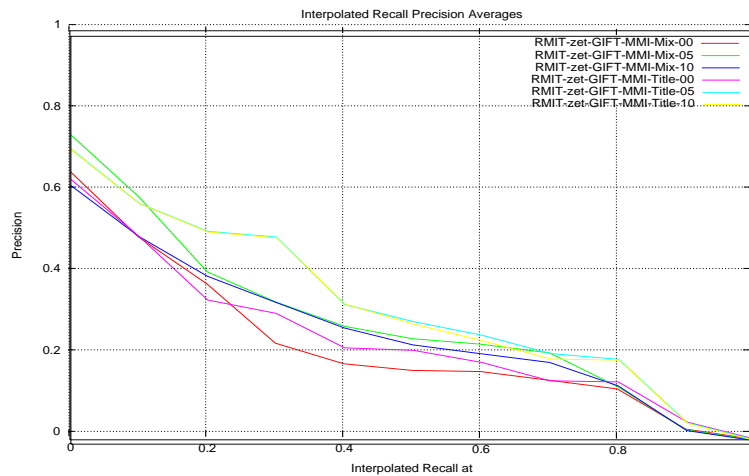


Fig. 5. Interpolated recall precision plot for the MMImages task

7.3 Multimedia Fragment Task

For the MMFragments task, we conducted two official runs. The fusion function was not applied since a title run will not include any image retrieval as no image ID exist in the title elements and a mix run with the α value of 1.0 does not require RSVs from the image retrieval system.

Based on Figure 7, we observed an opposite behaviour in the mean average effort precision. Run RMIT-MMF-Mix-10 performed better than run RMIT-MMF-Title-10. This shows that the present of other elements content improves the retrieval performances.

We also analysed the additional runs retrieval performance. There was a minor improvement when incorporating the RSVs from image retrieval, however the improvement was not significant.

8 Conclusions

In this paper we have reported on our participation in the Ad hoc and MM tracks of INEX 2006. We utilised a full-text information retrieval system for both tracks to retrieve the XML documents and a content-based image retrieval system was used for the MM track.

Three runs was submitted to the Ad hoc Thorough task. The Okapi BM25 probabilistic model has proven to be the best information retrieval approach for the Wikipedia collection.

As part of the XML-multimedia retrieval task, we submitted six runs for the MMImages task and two runs for the MMFragments task. The runs for the MMImages task reflect the various relative weights of 0 to 1. For the MMFragments task, the official

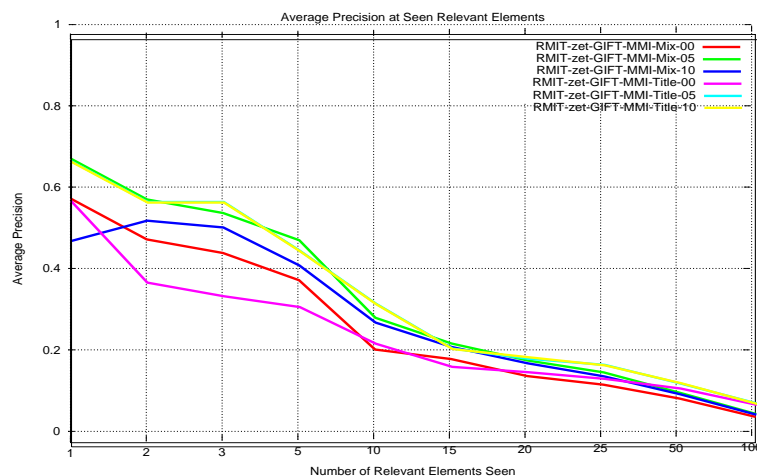


Fig. 6. Precision at cut-off n plot for the MMImages task.

runs were only based on the text retrieval system. We executed the additional runs that include image retrieval; however there was a minor performance improvement.

We have used the linear combination of evidence to merge the RSVs from two retrieval subsystems for retrieving multimedia information from structured documents. We conclude that the content-oriented XML retrieval system benefits by using some evidence from a CBIR system; indeed, as measured by MAP, increasing the weight of the text retrieval system component in the fusion system yields better performance than when any of the two subsystems are used alone. When only a CBIR system is used to retrieve multimedia document fragments, precision is poor.

Acknowledgements

This research was undertaken using facilities supported by the Australian Research Council, an RMIT VRH grant and a scholarship provided by the Malaysian Ministry of Higher Education.

References

1. Y. A. Aslandogan and C. T. Yu. Evaluating Strategies and Systems for Content-Based Indexing of Person Images on the Web. In *MULTIMEDIA 2000: Proceedings of the Eighth ACM International Conference on Multimedia*, pages 313–321, New York, NY, USA, 2000. ACM Press.
2. N. Fuhr, M. Lalmas, S. Malik and G. Kazai (editors). *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005, Revised Selected Papers*, Volume 3977 of *Lecture Notes in Computer Science*. Springer, 2006.

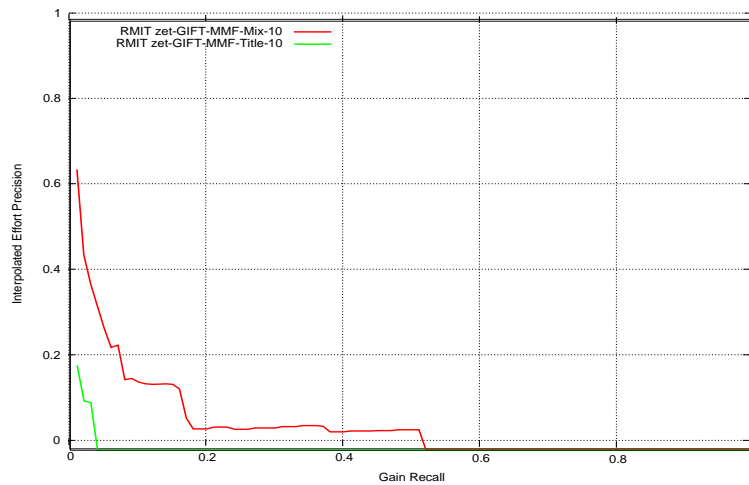


Fig. 7. mean average effort precision (MAep) plot for the MM Fragment task.

3. D. N. F. Awang Iskandar, J. Pehcevski, J. A. Thom and S. M. M. Tahaghoghi. Combining image and structured text retrieval. In Fuhr et al. [2], pages 525–539.
4. G. Kazai and M. Lalmas. Inex 2005 evaluation measures. In Fuhr et al. [2], pages 16–29.
5. J. Pehcevski, J. A. Thom and S.M. M. Tahaghoghi. RMIT University at INEX 2005: Ad Hoc Track. In Fuhr et al. [2], pages 306–320.
6. A. Singhal, C. Buckley and M. Mitra. Pivoted document length normalization. In *Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 21–29, Zurich, Switzerland, 1996.
7. K. Sparck Jones, S. Walker and S. E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments. Parts 1 and 2. *Information Processing and Management*, Volume 36, Number 6, pages 779–840, 2000.
8. D. M. Squire, W. Müller, H. Müller and T. Pun. Content-based Query of Image Databases: Inspirations from Text Retrieval. *Pattern Recognition Letters*, Volume 21, Number 13–14, pages 1193–1198, 2000. (special edition for SCIA'99).
9. D. Tjondronegoro, J. Zhang, J. Gu, A. Nguyen and S. Geva. Integrating text retrieval and image retrieval in xml document searching. In Fuhr et al. [2], pages 511–524.
10. R. van Zwol. Multimedia strategies for \mathcal{I}^3 -sdr, based on principal component analysis. In Fuhr et al. [2], pages 540–553.
11. I. H. Witten, A. Moffat and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images, Second Edition*. Morgan Kaufmann Publishers, 1999.
12. C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, Volume 22, Number 2, pages 179–214, 2004.
13. J. Zobel and A. Moffat. Exploring the similarity space. *ACM SIGIR Forum*, Volume 32, Number 1, pages 18–34, 1998.

XFIRM at INEX 2006 - Preliminary work. Ad-hoc, Relevance Feedback and MultiMedia tracks

Lobna Hlaoua, Mouna Torjmen, Karen Pinel-Sauvagnat, and Mohand
Boughanem

SIG-RFI, IRIT, Toulouse, France

Abstract. This paper describes experiments carried out with the XFIRM system in the INEX 2006 framework. The XFIRM system uses a relevance propagation method to answer CO and CO+S queries. Runs were submitted to the ad-hoc, relevance feedback and multimedia tracks.

At the time where we're writing the paper for the pre-proceedings, all official results are still not known and further experimentations are currently running. Results of all runs will be published in the final proceedings.

1 Ad-hoc track

The aim of our participation to the ad-hoc track is to evaluate on the wikipedia collection [2] the algorithms we proposed last year. CO and CO+S queries are processed as explained in [5].

1.1 Runs

Thorough strategy. For the Thorough task, all nodes having a non-zero relevance value are returned by the XFIRM system.

Focussed strategy. In order to reduce/remove nodes overlap, for each relevant path, we keep the most relevant node in the path. The results set is then parsed again, to eliminate any possible overlap among ideal components.

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

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

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

Best in Context strategy. For the Best in Context task, nodes relevance is first computed in the same way than for the thorough strategy. We then only keep the most relevant node per article.

2 Relevance feedback track

In our previous works, we proposed two main approaches for relevance feedback:

- a content-oriented approach, which expands the query by adding keywords terms,
- a structure-oriented approach, which expands the query by adding structural conditions.

For the 2006 RF track, we propose to apply the probabilistic theory in the Content-Oriented approach and to improve our Structure-Oriented approach already presented in [5].

2.1 Content-Oriented Relevance Feedback

Probabilistic method

Simple term extraction (without query terms reweighting) using the Rocchio's algorithm [4] has already been explored and did not show any improvement [5]. The main question is still how to extract and to weight the best terms that will be added to the query. Our approach in this paper is inspired from another very known way to do RF, the probabilistic model [3].

Let us consider E^r as a set of relevant elements; we define the probability of term expressiveness by $P(t_i/E)$. We estimate this probability according to a simplified version of the Robertson's formula [3]:

$$P(t_i/E^r) = r_e/R_e \quad (1)$$

where r_e is the number of elements in E^r containing term t_i and $R_e(= ||E^r||)$ is the number of relevant elements.

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

If original query terms appear in the top k , we do not add them again.

Re-weighting keywords using the probabilistic method

In previous work, we considered term relevance as a binary measure (relevant/not relevant). No preference between terms could be expressed and the user need was not really refined. Therefore, we propose here to add weights to query terms, that represent their degree of importance. Weight values vary in]0,1]. We use the scores calculated according to the probabilistic method described above. The higher weight is assigned to original query terms (weight=1).

For example, let $Q = t_1, \dots, t_n$ be the initial query. If we choose to add 3 relevant terms to the initial query, the new query will be:

$Q'=(t_1, 1), \dots, (t_n, 1), (t_o, w_o), (t_p, w_p), (t_r, w_r)$ where t_o, t_p and t_r are the added terms and w_o, w_p and w_r their associated weights.

2.2 Combined approach: content-and-structure RF

In the combined approach, both structural constraints and keywords terms are added to the initial query, i.e. we combine the content-oriented and the structure-oriented approaches. The principle of the structure-oriented approach is recalled in the following paragraph.

Structure-oriented approach

Our hypothesis in the structure-oriented approach is that for a given query, the user may only be interested to some types of elements (like for example *section*, *article*, *images*,...). Our approach consists in refining the initial query by adding some structures, extracted from the set of judged elements that could contain the information needed by the user. The idea behind structure-oriented RF is therefore to find for each query the appropriate generic structures, called here the generative structures, shared by the greatest amount of relevant elements. The generative structures are extracted as follows. Let:

- e_i be an element $\in E^r$; e_i is characterized by a path p_i and a score w_i initialized to 1 at the beginning of the algorithm. p_i is only composed of tag names, like the following example: */article/body/sec*.

- CS be a set of Common Structures, obtained as algorithm output.

For each $(e_i, e_j) \in E^r \times E^r, i \neq j$, we apply the SCA algorithm, which allows the identification of the smallest common ancestor of e_i and e_j . The path of this smallest common ancestor is then added to the set of common structures CS. The SCA algorithm is processed for each pair of E^r elements, and is described below.

```

SCA( $e_i, e_j$ ) return boolean
Begin
if  $p_i.first = p_j.first$  then
  if  $p_i.last = p_j.last$  then
    if  $\exists e_p(p_p, w_p) \in CS / p_p = p_i$  then
       $w_p \leftarrow w_p + w_j$ 
      return true
    else  $CS \leftarrow e_j(p_j, w_j + w_i)$ 
      return true
  else
    if  $head(p_j) \neq null$  then
       $p'_j \leftarrow head(p_j)$ 
       $w'_j \leftarrow w_j / 2$ 
      SCA ( $e_i(p_i, w_i), e'_j(p_j, w_j)$ )
    else SCA( $e_j, e_i$ )
return false
End

```

$p.last$ and $p.first$ are respectively the last and the first tag of the path p and $head(p)$ is a function allowing to reduce the path p , by removing the last tag of the path. For example, $head(/article/body/section) = /article/body$.

In our approach, we are only interested in element tags and we choose to only compare the *p.last* tags of paths (even if elements having the same *p.last* can have different paths). When no result is added in the CS set by the $SCA(e_i, e_j)$ algorithm, we try to run $SCA(e_j, e_i)$.

In order to express the new (COS) query, we then extract the k top ranked structures according to their score w_i in the CS set. The selected structures are then used in their simple form (i.e. the last tag).

Let $Q1 = t_1, t_2, \dots, t_n$ be a query composed of n keywords (CO query) and $S1, S2$ be two generative structures of the set of Common Structures CS. The new query derived from $Q1$ using our structure-oriented relevance feedback method will be: $Q1' = S1[t_1, t_2, \dots, t_n] \text{ OR } S2[t_1, t_2, \dots, t_n]$.

Combined approach

As explained before, the combined approach consists in adding both content and structure constraints to the initial query. The new query (that will be a CO+S query), is thus composed of the most appropriate generic structures and of the k best terms according to formula 1. Terms are added to the original query terms with their associated weights. To generalize, the extended query is rewritten according to the grammar in table 1.

Table 1. Grammar of queries re-writing according to the combined approach.

<p>Let R be the reformulation of an unstructured query: R::= <structure constraint><c1><Keywords><c2><Structured expressions>* Keywords::= <InitialKeywords>*<ExtractedKeywords>* Initialkeywords::=<term><,1> Extractedkeywords::=<extractedTerm><,score> term::=term of initial query extractedTerm::=extracted term from relevant elements score::=term score calculated according to the probabilistic method Structured expressions::=<Operator> <structure constraint> <c1> <Keywords><c2> Operator::="OR" structure constraint::=name of the extracted tag in the CS set c1::="[" c2::="]"</p>
--

Example

Let $Q1 = t_1, t_2, \dots, t_n$ be a query composed of n keywords (CO query) and $S1$ and $S2$ be two generative structures extracted from the set of Common Structures. The new query derived from $Q1$ using our combined Relevance Feedback method will be (we choose to add for example 2 generative structures and 2 relevant terms t_o , and t_p): $Q1' = S1[(t_1, 1), \dots, (t_n, 1), (t_o, 1), (t_o, w_o), (t_p, w_p)] \text{ OR } S2 [(t_1, 1), \dots, (t_n, 1), (t_n, 1), (t_o, w_o), (t_p, w_p)]$

2.3 Runs

CO.thorough task

For official runs and according to previous experiments, we use :

- the Content-Oriented RF approach by adding 3 or 10 expressive terms to the initial query,
- the combined approach with 3 expressive terms and 3 generative structures selected according to the SCA algorithm.

The top 20 elements of each query is used to select relevant terms / relevant structures.

Non-standard methods to select elements used to choose relevant terms and relevant structures are also tested. We propose to evaluate the impact of the number of judged elements used to extend queries, by using elements in the top 10 and top 40. We also propose to consider a fixed number of relevant elements to extend queries (4 strictly relevant elements are used in this paper).

COS thorough task

For this task we tested the content-oriented RF using 3 and 5 terms and the combined approach using 3 expressive terms and 1 generative structure.

3 Multimedia track

Two types of topics are explored in the INEX Multimedia Track: MMFragments and MMImages.

A MMFragments topic is a request which objective is to find relevant XML fragments given a multimedia information need. Here, the topic asks for multimedia fragments, *i.e* fragments composed of text and /or images.

A MMImages topic is a request which objective is to find relevant images given an information need. Here, the type of the target element is defined as an 'image'. This is basically image retrieval, rather than XML element retrieval.

Some topics use image as query: the user indicates by this way that results should have images similar to the given example.

In Image Retrieval, there are two main approaches [6] : (1) Context Based Image Retrieval and (2) Content Based Image Retrieval:

1. The context of an image is all information about the image coming from others sources than the image itself. At the moment, only the textual information is used as context. The main problem of this approach is that documents can use different words to describe the same image or can use the same words to describe different concepts.
2. Content Based Image Retrieval (CBIR) systems use low-level image features to return images similar to an image used as example. The main problem of this approach is that visual similarity does not correspond to semantic

similarity(for example a CBIR system can return a picture of blue sky when the example image is a blue car).

The work presented in this paper belongs to the first approach.

3.1 Runs

MMFragments topics processing: For MMFragments topics, we use the CO and the CO+S query processing model of the XFIRM system [5].

MMImages topics processing: Our method uses the text surrounding images and structure of document to judge the relevance of images. A first step is to search relevant nodes according to the XFIRM Content Only method. Then, we only use relevant documents and we reduce our retrieval domain to both relevant nodes and images nodes belonging to relevant documents. For each image, we use the closest nodes to judge its relevance. The used nodes are: the descendant nodes, the ancestor nodes and the brother nodes(Figure 1). We also used the name of the image in our model.

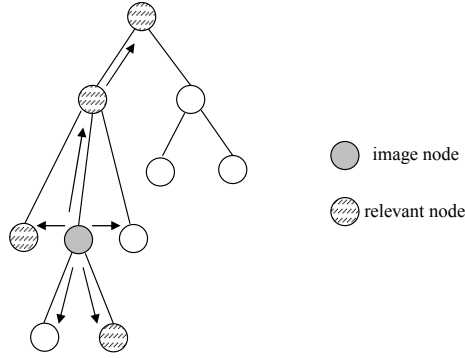


Fig. 1. Use of ancestors, brothers and descendants nodes to evaluate images relevance

An image weight corresponding to each of the preceding sources of evidence is computed:

- W_d^{im} is the image weight computed using descendant nodes,
- W_b^{im} is the image weight computed using brother nodes,
- W_a^{im} is the image weight computed using ancestor nodes,
- W_{name}^{im} is the image weight computed using image name.

The total image weight is then expressed as follows:

$$W_{im} = p_1 \cdot W_d^{im} + p_2 \cdot W_a^{im} + p_3 \cdot W_b^{im} + p_4 \cdot W_{name}^{im} \quad (2)$$

where p_1, p_2, p_3 and p_4 are parameters used to emphasize some weights types. In these experiments, we use $p_1 = p_2 = p_3 = p_4 = 1$. With this method, all the images of the relevant documents are evaluated and will have a weight > 0 . Indeed, they will inherit at least of the root node weight. We detail the evaluation of each weight in the following paragraph. To evaluate the weight of an image using its descendant nodes, we use the weight of each relevant descendant node obtained by the XFIRM-CO method (W_{rdi}), the number of relevant descendant nodes according to the XFIRM model ($|d|$) and the number of non-relevant descendant nodes ($|\bar{d}|$).

$$W_d^{im} = f(W_{rdi}, |d|, |\bar{d}|) \quad (3)$$

If the number of relevant descendant nodes is greater than the number of non-relevant descendant nodes then they will have more importance in the weight evaluation. Basing on this intuition, we use the following formula in our experiments.

$$W_d^{im} = \left(\frac{|d| + 1}{|\bar{d}| + 1} \right) * \sum_{i=1}^{|d|} W_{rdi} \quad (4)$$

To evaluate the weight of an image using its brother nodes, we use the weight of each relevant brother nodes obtained by the XFIRM-CO method (W_{rbi}), the distance between the image node and each brother node ($dist(im, b_i)$): the larger the distance of the brother node from the image node is, the less it contributes to the image relevance. Finally, we use the number of relevant brother nodes $|b|$ and the number of non-relevant brother nodes $|\bar{b}|$

$$W_b^{im} = f(W_{rbi}, dist(im, b_i), |b|, |\bar{b}|) \quad (5)$$

Figure 2 shows how distances between an image node and brother nodes are calculated:

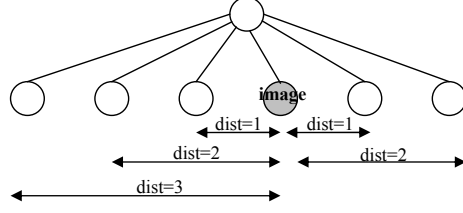


Fig. 2. distance between image node and brother nodes

The formula used in experiments presented here is :

$$W_b^{im} = \left(\frac{|b| + 1}{|b|}\right) * \left(\sum_{i=1}^{|b|} \frac{W_{rbi}}{dist(im, b_i)}\right) \quad (6)$$

To evaluate the weight of an image using its ancestors nodes, we add relevant ancestor nodes weights obtained with the XFIRM-CO method (W_{rai}). The XFIRM-CO method uses the distance between the relevant node and its ancestors to evaluate the ancestors weights of an element: the larger the distance of a node from its ancestor is, the less it contributes to the relevance of its ancestor. Our method also uses the distance $dist(im, a_i)$ between the image node and its ancestors: the larger the distance from an ancestor node is, the less it contributes to the relevance of the image node. We used the following formula:

$$W_a^{im} = \sum_{i=1}^{|a|} \frac{\log(W_{rai} + 1)}{dist(im, a_i) + 1} \quad (7)$$

where $|a|$ is the number of relevant ancestors nodes according to the XFIRM model.

Each image in the collection documents has a name, and this name can be used to determine if image is relevant or not. Our method evaluate the name image weight with (1) the frequency of each query term in the image name $freq(t_i^{im-name})$, (2) the number of image name terms: the size of image name is important to determine relevant images $|t_{im-name}|$ (finding a keyword 1 time in an image name having size=1 is better than finding a keyword 2 times in an image name having size=5), (3) the numbers of different keywords in the query $|t_q|$: this aims to give importance to the number of keywords found compared to the total number of query terms (having 3 relevant keywords from 3 keywords is better than having 3 relevant keywords from 5 keywords).

$$W_{name}^{im} = f(freq(t_i^{im-name}), |t_{im-name}|, |t_q|) \quad (8)$$

More precisely, we used the following formula:

$$W_{name}^{im} = \left(\frac{1}{|t_q|}\right) * \left(\sum_{i=1}^{|t_q|} \frac{freq(t_i^{im-name})}{|t_{im-name}|}\right) \quad (9)$$

3.2 Results

MMFragments results are based on 9 topics, whereas MMImages results are based on 13 topics.

MMFragments Results Results of the MMFragments task are based on the Mean Average Precision (MAP) metric.

Method	MAP
CO : $\alpha = 0.6, \rho = 1$	0.008599
CO : $\alpha = 0.6, \rho = 0.9$	0.015488
CO : $\alpha = 0.5, \rho = 0.9$	0.01596
CO : $\alpha = 0.7, \rho = 0.9$	0.01479
COS : $\alpha = 0.9, dict1$	0.010142
COS : $\alpha = 0.9, dict2$	0.01079
CO : $\alpha = 0.1, \rho = 0.9$	0.01765
CO : $\alpha = 0.6, \rho = 0.9$	0.01762

Table 2. MMFragments official results

Grayed boxes are results of our official runs. Runs using the COS query processing model of the XFIRM system use two different dictionary indexes: *dict1* contains simple tag equivalencies (for example, *image* and *figure* are considered as equivalent tags), whereas *dict2* contains very extended equivalencies between tags.

The best MAP is obtained with the MM method, where the target element is always an image node.

MMImages Results Two metrics are used to evaluate MMImages topics: MAP (Mean Average Precision) and BPREF (Binary PREFERence). Results are presented in table 3:

Method	MAP	BPREF
CO : $\alpha = 0.6, \rho = 1$	0.1084	0.1379
COS : $\alpha = 0.6$	0.2372	0.218
COS : $\alpha = 0.1$	0.1989	0.2005
COS : $\alpha = 0.2$	0.2154	0.2195
MM : $\alpha = 0.6, \rho = 1$	0.2119	0.2088
MM : $\alpha = 0.6, \rho = 0.9$	0.2152	0.2098
MM : $\alpha = 0.1, \rho = 1$	0.1906	0.2058
MM : $\alpha = 0.1, \rho = 0.9$	0.1902	0.2043
MM : $\alpha = 0.2, \rho = 1$	0.1892	0.2011
MM : $\alpha = 0.2, \rho = 0.9$	0.1895	0.2013

Table 3. MMImages Results

We used 3 methods: XFIRM CO method, XFIRM COS method and MMImages method. Grayed boxes are results of our official runs. Best MAP is 0.2372 using XFIRM COS method with parameter $\alpha=0.6$. Best BPREF is 0.2195 using XFIRM COS method with parameter $\alpha=0.2$.

Results with the XFIRM methods are better than results with the MMImages method. This can be explained by the structure of the images collection. The MMImages method is based on the place of textual information in the structure (ascendants, descendants, brothers), whereas in the MMimages collection, all textual content is in ancestors nodes.

In future work, we plan to:

- differentiate methods used to assign weights to image fragments and text fragments.
- process queries by example by using text surrounding images used as examples
- add additional sources of information to process queries of the MMImages task: Images classification scores, Images features vectors and a CBIR system,...

References

1. M. Boughanem, T. Dkaki, J. Mothe, and C. Soule-Dupuy. Mercure at TREC-7. In *Proceedings of TREC-7*, 1998.
2. L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 2006.
3. S. E.Robertson and J. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, pages 129–146, 1976.
4. J. Rocchio. *Relevance feedback in information retrieval*. Prentice Hall Inc., Englewood Cliffs, NJ, 1971.
5. K. Sauvagnat, L. Hlaoua, and M. Boughanem. Xfirm at inex 2005: ad-hoc and relevance feedback track. In *INEX 2005 Workshop proceedings*, pages 88–103, 2005.
6. W. Thijs. Image retrieval: Content versus context. In *RIAO*, pages 276–284, 2000.

Information Fusion in XML Document Searches by Combining Text and Image Retrieval Techniques

D. Tjondronegoro, C. Lau, J. Zhang, S. Geva
Queensland University of Technology
2 George Street, GPO Box 2434, Brisbane, QLD 4001 Australia
a3.lau@student.qut.edu.au
{[dian.jinglan.zhang](mailto:dian.jinglan.zhang@qut.edu.au), [s.geva](mailto:s.geva@qut.edu.au)}@qut.edu.au

Current multimedia technology has supported the increasing use of structured XML document collections that contain a mixture of text and multimedia data such as images, speech, and videos. Images play an important role in webpage or article presentation, however, this information has not been explored sufficiently in traditional Information Retrieval systems. The combination of text and image retrieval techniques in information retrieval has just begun recently and thus many research questions are still open. The main objective of this research is to develop and evaluate algorithms for structured document retrieval systems using a comprehensive database of XML documents containing text- and image-based information.

The document collection used is provided by the INEX 2006 organizing committee. The corpus contains 166,559 images in formats such as PNG, JPG and GIF. This complex selection of images depicting both natural and man-made objects (such as landscape, people, animals, buildings, and logos) comes in different sizes as well as different colour depths. This project aims to create a Content Based Image Retrieval (CBIR) system which can deal with a large set of heterogeneous images and will work together with an existing text-based search engine in order to elevate the quality of the final retrieval results.

In this research, a text and an image-based search engines are used concurrently and the results are obtained by fusing two or more independent result sets. Therefore, the primary contribution is to answer the challenging question on how to fuse multiple results to form an optimal query results for users. This involves the fusion of the XML document retrieval results of multiple search algorithms on the same collection. Queries consist of both text (keywords) and/or example images. Two document retrieval algorithms are used: a text-based retrieval using a *TF-IDF* variant, and an image-based retrieval using image feature similarity. The framework of the prototype system is shown Figure 1.

Three (3) image related tables are used in the database: Images, Features, and AdhocXML. The *Images* table stores the information about the images, including the image ID (e.g. 22625), its original filename (e.g. *DVD-RW_Spindle.jpg*), the collection set and the subfolder name. The *Features* table stores the image feature extracted using the feature extractor. The *AdhocXML* table stores the Adhoc XML filename and the absolute XPath of where the related image appears in the document. This allows the system to back-track the original XML document which uses the image and fuses the image and text search results.

The selected image features for this experiment include colour histograms (RGB/HSV/YCbCr), textures, detectable lines (Hough transformation), and the UvA features provided by INEX2006 organizing committee (developed by a research group in University of Van Amsterdam). The UvA feature uses Natural Image Statistics, Colour invariant edge detection and regional descriptors to represent an image. These features of every image are stored in the database for CBIR.

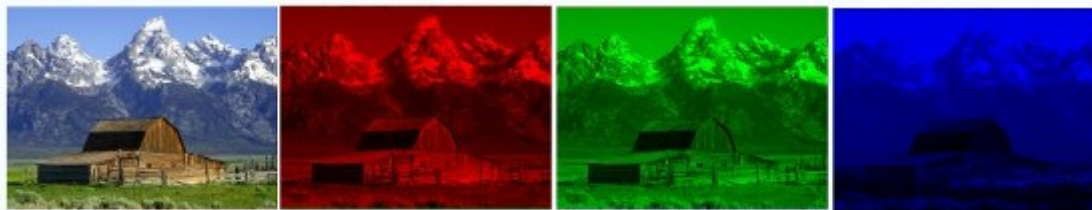
All image features are represented by vectors. Different distance measures such as the correlation coefficient, Euclidean distance, and Manhattan distance have been implemented and

can be chosen from the Graphical User Interface. The Euclidean distance between two vectors is used for the final submission. All individual distances are normalized to the range of [0, 1].

The image feature used in this system were **color histograms, texture and detectable lines.**

RGB Color Histogram

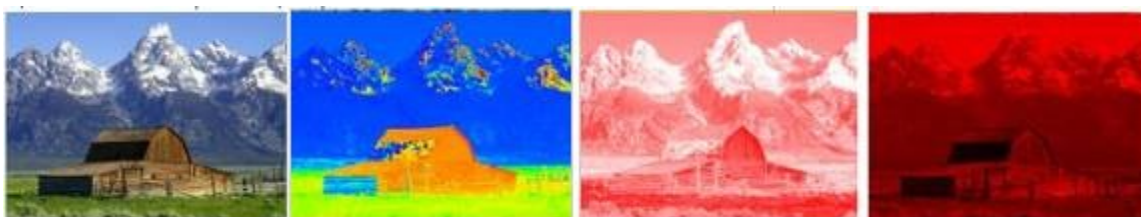
An RGB Color image is an $M \times N \times 3$ array of color pixel, where each color pixel is responding to the Red, Green and Blue component of an image at a specific location. It is derived from a computer scanning through each of these RGB value and counts how many are at each level from 0 to 255. This provides a far more compact overview of the data in an image than knowing the exact value of every pixel. The color histogram of an image is invariant with translation and rotation about the viewing axis, and varies only slowly with the angle of view. This makes the color histogram particularly suited to recognising an object of unknown position and rotation within a scene. Importantly, translation of an RGB image into the illumination invariant rg-chromaticity space allows the histogram to operate well in varying light levels.



RGB image representation

HSV Color Histogram

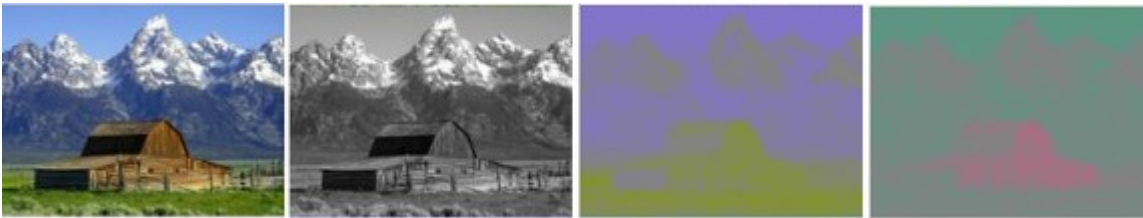
HSV, also known as HSB or HSL, stands for Hue, Saturation and Value(Brightness or Luminosity). Hue refers to the colour type (red, blue or yellow), whereas Saturation refers to how vibrant the colour is, usually measure from 0-100% and Value is the brightness of the colour. It is one of the most commonly used by people (especially artist) to select colour. As compared to RGB which is an additive model, HSV encapsulates information about a colour in a way more understandable and recognisable to humans because we humans perceive colour via its colour name, vibrancy and darkness/brightness. Artist believes that HSV is referring approximately to tint, shade and tone. It is also widely utilized in computer graphics applications, to apply colours to a particular graphical element.



HSV image representation

YcbCr Color Histogram

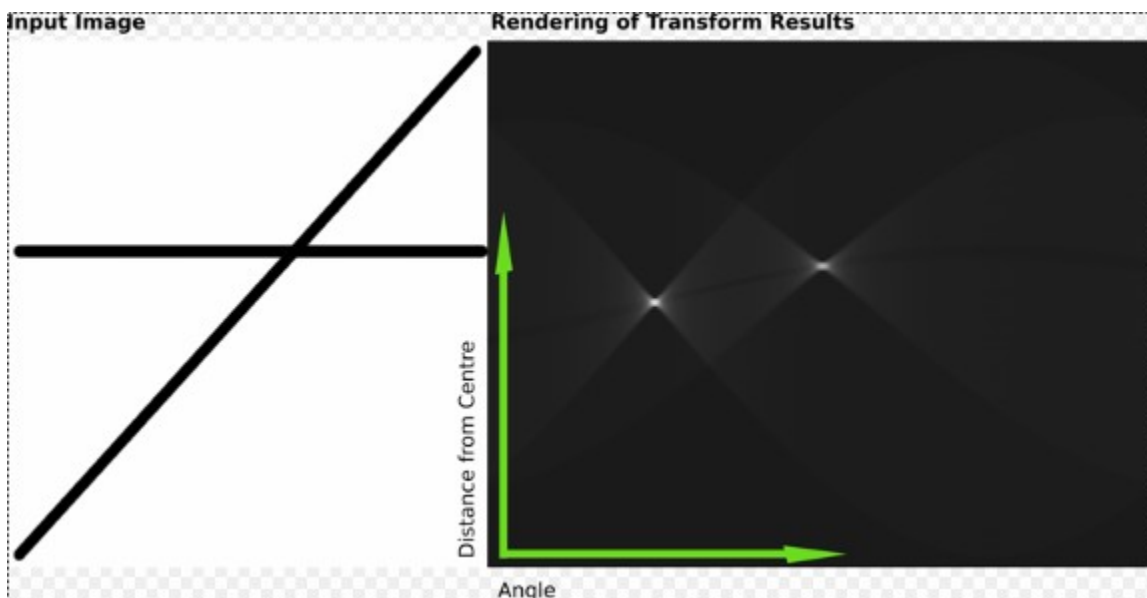
YCbCr is a family of color spaces used in video systems. Y is the luma component and Cb and Cr the blue and red chroma components. It is widely used in digital video. It represent the luminance level using a single component, Y, and the other color information is stored individually at Cb and Cr. It is a method to encode RGB information. The actual color displayed is dependant to the actual RGB colorants used to display the signal.



YCbCr image representation

Hough transform

Hough transform method is widely used in digital image processing to detect arbitrary shapes and identify lines and such as buildings. Therefore, in this project, we expect to utilize Hough Transform method to have excellence performance classification on some topics such as buildings, coins and some arbitrary shapes since they have unique lines strength characteristics. The efficiency of the Hough Transform is relies on the image qualities. Edges and corners must be sharp to improve the efficiency of Hough Transform. Performing Hough Transform on noisy images will result a lower accuracy. Therefore there will normally be a denoising phase beforehand. As seen in the figure, the results of is stored in matrix form to represent the number of lines passed through any point. The brighter point indicates the intersection of two lines.





Sample image retrieval of topic “coins” using Hough Transform

Texture

To effectively describe a region is to quantify its texture content. In this project, we will be using the following statistical moment of the image to perform texture measurement

Moment	Expression	Description
Mean	$m = \sum_{i=0}^{L-1} z_i p(z_i)$	To measure the average intensity
Standard deviation	$\sigma = \sqrt{\mu_2(z)} = \sqrt{\sigma^2}$	To measure the average contrast
Smoothness	$R = 1 - 1/(1 + \sigma^2)$	To measure the relative smoothness of the intensity in a region.
Third moment	$\mu_3 = \sum_{i=0}^{L-1} (z_i - m)^3 p(z_i)$	To measure the skewness of a histogram
Uniformity	$U = \sum_{i=0}^{L-1} p^2(z_i)$	To measure the uniformity
Entropy	$e = - \sum_{j=0}^{L-1} p(z_j) \log_2 p(z_j)$	To measure the randomness

UvA Features

The feature is named UvA because the algorithm from the research group in University of Van Amsterdam. This feature uses Natural Image Statistics, Color invariant edge detection and regional descriptors to represent an image. The paper present a 120D features using different texture parameters.

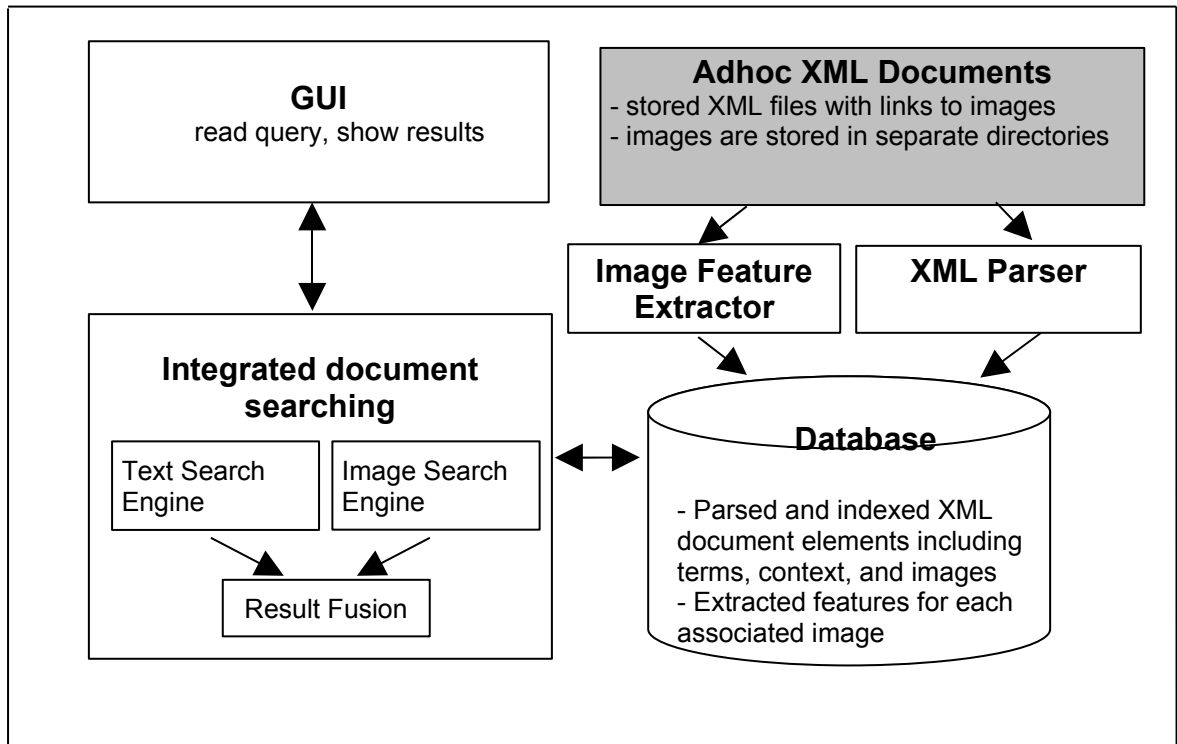


Figure 1. System Architecture

The fusion tasks consist of two parts: the fusion of the retrieval results from different image features and the fusion of the results from the text-based searches and the image-based searches. For combining the search results of different image features, a weighted sum is used for calculating the combined distance from two or more individual distances calculated using different features. This is based on the hypothesis that the combination of more evidence will increase the reliability of the CBIR system.

For the fusion of the results of the two text-based and image-based search engines, we use the result of the text search engine as the base, and then the image search result to boost the former result. More specifically, if a document appears in both the results of the text and image search engine, then the final rank of the document is pushed to the top. This is again based on the hypothesis that the combination of more evidence will increase the reliability of the retrieval. This hypothesis has not been rigorously tested in this experiment. At any rate, sometimes the individual search results (either image-based or text-based) are better than the combined result while other times the combined result is better.

Benchmarking Multimedia Search in Structured Collections

Thijs Westerveld¹ and Roelof van Zwol²

¹ CWI, Amsterdam, The Netherlands

² Yahoo! Research, Barcelona, Spain

Abstract. The multimedia track focuses on using the structure of the document to extract, relate, and combine the relevance of different multimedia fragments. This paper presents a brief overview of the track, its collection tasks and goals. We also report the results and the approaches of the participating groups.

1 Introduction

Structured document retrieval allows for the retrieval of document fragments, i.e. XML elements, containing relevant information. The main INEX Ad Hoc task focuses on text-based XML element retrieval. Although text is dominantly present in most XML document collections, other types of media can also be found in those collections. Existing research on multimedia information retrieval has already shown that it is far from trivial to determine the combined relevance of a document that contains several multimedia objects. The objective of the INEX multimedia track is to exploit the XML structure that provides a logical level at which multimedia objects are connected, to improve the retrieval performance of an XML-driven multimedia information retrieval system.

The multimedia track will continue to provide an evaluation platform for the retrieval of multimedia document fragments. In addition, we want to create a discussion forum where the participating groups can exchange their ideas on different aspects of the multimedia XML retrieval task. Ideas raised here, may provide input for this the task descriptions for this year and/or the coming years.

The remainder of this paper is organised as follows. First we introduce the main parts of the test collection: documents, topics, tasks and assessments (Sections 2–5). Section 6 summarises the main results and the approaches taken by the different participants. The paper ends with conclusions and an outlook on next year’s track in Section 7.

2 Wikipedia collection and other resources

2.1 Wikipedia

In 2006, the main INEX tracks use a corpus based on the English part of Wikipedia, the free content encyclopedia (<http://en.wikipedia.org>). Wiki-text pages have been converted to XML [1] making a structured collection of

659,388 XML pages. This collection is also the basis for the multimedia track. Statistics about the collection and its multimedia nature are listed in Table 1. Each of the images on Wikipedia comes with a small metadata document, usu-

Table 1. Wikipedia collection statistics

Total number of XML documents	659,388
Total number of images	344,642
Number of unique images	246,730
Average number of images per document	0.52
Average depth of XML structure	6.72
Average number of XML nodes per document	161.35

ally containing a brief caption or description of the image, the Wikipedia user who uploaded the image, and the copyright information. For the images in the Wikipedia collection, these metadata documents are downloaded and converted to XML. Figure 1 shows an example of such a metadata document. Some documents had to be removed because of copyright issues or parsing problems, leaving us with a collection of 171,900 images with metadata.



Fig. 1. Example Wikipedia image metadata document as in the *MMimages* collection.

Both the main Wikipedia collection and the Wikipedia images + metadata collection are used in the INEX 2006 multimedia track. The first in the Ad Hoc XML retrieval task of finding relevant XML fragments given a multimedia

information need, the second in a pure image retrieval task: Find images (with metadata) given an information need.

2.2 Additional sources of information

In 2006 the two Wikipedia based collections are the main search collections. The returned elements should come from these collections. A number of additional sources of information is provided to help participants get to the relevant information in these collections.

Image classification scores: For each image the classification scores for 101 different concepts are provided by UvA [4]. The concepts are shown in Figure 2.2. The UvA classifier is trained on manually annotated TRECVID video data and the concepts are picked for the broadcast news domain. It is unclear how well these concept classifiers will perform on the broad collection of Wikipedia images, but we believe it may be a useful source of information.

Image features: A set of 22D feature vectors, one for each image, is available that has been used to derive the image classification scores [7]. Participants can use these feature vectors to build a custom CBIR-system, without having to pre-process the image collection.

CBIR system: An on-line service to get a ranked list of similar images given a query image (from the collection) is provided by RMIT [3].

3 Topics

The topics used in the INEX MM track are descriptions of (structured) multimedia information needs. Structural and multimedia hints are expressed in the NEXI query language[6].

For the INEX 2005 multimedia track, NEXI has been extended to incorporate visual hints. If a user wants to indicate that results should have images similar to a given example image, this can be indicated in an about clause with the keyword *src:*. For example to find images of cityscapes similar to the image with identifier 789744, one could type

```
//image[about(.,cityscape) and about(.,src:789744)]
```

To keep things manageable, only example images from within the MM collection are allowed.

In 2006, we use the same visual similarity syntax. In addition, a second type of visual hints is introduced; it is directly related to the concept classifications that are provided as an additional source of information. If a user thinks the results should be of a given concept, this can be indicated with an about clause with the keyword *concept:*. For example, to search for cityscapes one could decide to use the concept building:

```
//image[about(.,cityscape) and about(.,concept:building)]
```

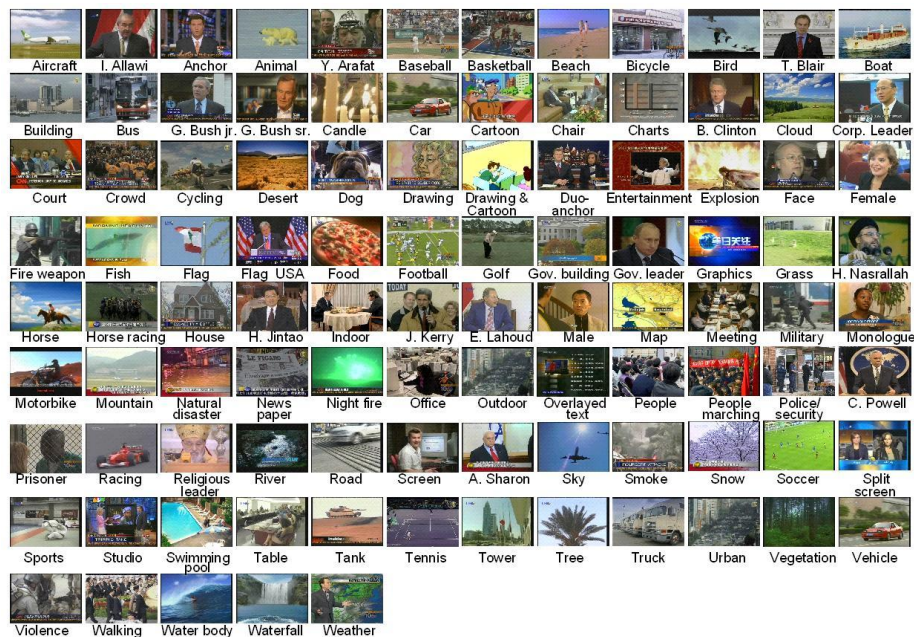


Fig. 2. The 101 concepts for which classification scores are available. This image is taken from [4]

Terms following the keyword *concept*: are obviously restricted to the 101 concepts for which classification results are provided (cf. Figure 2.2).

It is important to realise that all structural, textual and visual filters in the query should be interpreted loosely. It is up to the retrieval systems how to use, combine or ignore this information. The relevance of a fragment does not directly depend on these elements, but is decided by manual assessments.

3.1 Topic format and development

The INEX MM track topics are Content Only + Structure (CO+S) topics, like in the INEX Ad Hoc track. While in the field of multimedia the term content often refers to visual content, in INEX it means textual or semantic content of a document part. The term content-only is used within INEX for topics or queries that use no structural hints. CO+S topics are topics that do include structural hints. As explained above, in the multimedia track topics can also contain visual constraints.

The 2006 CO+S topics consist of the following parts.

`<title>` The topic `<title>` simulates a user who does not know (or does not want to use) the actual structure of the XML documents in a query and who does not have (or want to use) example images or other visual constraints.

The query expressed in the topic `<title>` is therefore a Content Only (CO) query. This profile is likely to fit most users searching XML digital libraries.

It is the standard web search type of keyword search.

- `<castitle>` A NEXI expression with structural and (optionally) visual hints.
- `<description>` A brief, matter of fact, description of the information need. Like a natural language description one might give to a librarian
- `<narrative>` A clear and precise description of the information need. The narrative unambiguously determines whether or not a given element fulfils the given need. It is the only true and accurate interpretation of a user's needs. Precise recording of the narrative is important for scientific repeatability - there must exist, somewhere, a definitive description of what is and is not relevant to the user. To aid this, the `<narrative>` should explain not only what information is being sought, but also the context and motivation of the information need, i.e., why the information is being sought and what work-task it might help to solve.
- `<ontopic_keywords>` Terms that are expected in relevant elements. These terms are recorded since they may be highlighted in the assessment interface for easier and more efficient assessing.
- `<offtopic_keywords>` Terms that are expected in non-relevant elements. Again, for highlighting during assessments.

In the 2005 pilot of the INEX multimedia track, topics came without a `<title>` field. This way, participants were encouraged to use the structural hints in the `<castitle>`. This year, the use of structural information is still encouraged, but we want to be able to compare structural approaches to baselines that use only `<title>` queries. Also 2005 topics lacked the `ontopic` and `offtopic` keywords fields.

3.2 Topic development

The topics in the multimedia track are developed by the participants. Each participating group has to create 6 multimedia topics. Topic creators first create a one to two sentence description of the information they are seeking. Then, in an exploration phase, they obtain an estimate of the amount of relevant information in the collection. For this, they can use any retrieval system, including their own system or the TopX system [5] provided through the INEX organisation. The topic creator then assesses the top 25 results and abandons the search if fewer than two or more than 20 relevant fragments are found. If between 2 and 20 fragments are found to be relevant, the topic creator should have a good idea of what query terms should be used, and the `<title>` is formulated. Using this title a new search is performed and the top 100 elements are assessed. Having judged these 100 documents, topic creators should have a clear idea of what makes a fragment relevant or not. Based on that, they could then first write the narrative and then the other parts of the topic. After each created topic, participants are asked to fill a questionnaire that gathers information about the users familiarity with the topic, the expected number of relevant fragments in the collection, the expected size of relevant fragments and the realism of the topic.

The submitted topics are analysed by the INEX organisers who check for duplicates and inconsistencies before distributing the full set of topics among the participants.

Only five groups submitted topics. Table 2 shows the distribution over tasks as well as some statistics on the topics.

Table 2. Statistics of INEX 2006 MM topics

	MMfrag	MMimg	Total
Number of topics	9	13	21
Avg. num. terms in <title>	2.7	2.4	2.6
Number of topics with src:	1	6	7
Number of topics with concept:	0	2	2

4 Tasks

The multimedia track knows two tasks, the first is –like in 2005– to retrieve relevant document fragments based on an information need with a structured multimedia character. A structured document retrieval approach should be able to combine the information in different media types into a single meaningful ranking that is presented to the user. This task is continued in INEX 2006 as the *MMfragments* task. A second task is started in 2006, *MMimages*, a pure image (with metadata) retrieval task. Both tasks are detailed below.

MM Fragments: Find relevant XML fragments given an multimedia information need. This task is in essence comparable to the retrieval of XML elements, as defined in the Ad Hoc track. The main differences with the INEX Ad Hoc track are that all topics in this track ask for multimedia fragments (i.e., fragments containing more than text only) and that the topics may contain visual constraints (see Section 3). The main difference with the 2005 task is that structural and visual constraints are not required and interpreted loosely if present. The core collection for this task is the Wikipedia main XML collection.

MM Images: Find relevant images given an information need. Here the type of the target element is defined (an image), so basically this is image retrieval, rather than element retrieval. Still, the structure of (supporting) documents could be exploited to get to the relevant images. The core collection for this task is the Wikipedia image collection.

All track resources (see Section 2) can be used for both tasks, but the track encourages participating groups to also submit a baseline run that uses no sources of information except for the target collection. This way, we hope to learn how

the various sources of information contribute to the retrieval results. Furthermore, we require each group to submit a run that is based on only the `<title>` field of the topic description. All other submissions may use any combination of the `<title>`, `<castitle>` and `<description>` fields. The fields used need to be reported.

5 Assessments

Since XML retrieval requires assessments at a sub-document level, a simple binary judgement at the document level is not sufficient. Still, for ease of assessment, retrieved fragments are grouped by document. Once all participants have submitted their runs, the top N fragments for each topic are pooled and grouped by document. To keep assessment loads within reasonable bounds, we used pools with a depth of 500 fragments. The documents are alphabetised so that the assessors do not know how many runs retrieved fragments from a certain document or at what rank(s) the fragments were found. Assessors then look at the documents in the pool and highlight the relevant parts of each document. The assessment system stores the relevance or non-relevance of the underlying XML elements.

In 2005, `<castitle>`s were enforced to be interpreted strictly. Thus, if someone asked for section elements, paragraphs could never be relevant. In 2006 this requirement is dropped; the relevance of a fragment is determined by the assessor and should be solely based on the `<narrative>`.

6 Results and Approaches

Only four participants submitted runs for the Multimedia track: CWI together with the University of Twente (CWI/UT), IRIT (IRIT), RMIT University (RMIT), Queensland University of Technology in Australia (QUTAU). Tables 3 and 4 give an overview of the topics fields and resources used by the runs. Most runs used either `<title>` or `<castitle>` in their runs, but RMIT experimented with runs using `<title>`, `<castitle>` and `<description>`. The wikipedia collections are mainly used for the tasks for which they are the target collection only, but CWI/UT experimented with using both wikipedia and wikipedia.IMG on the *MMfragments* task. The visual resources provided are used mostly in the *MMimages* task, although RMIT used their GIFT tool also on some *MMfragments* runs. QUTAU is the only group that used the concepts and features provided by UvA; they used them for *MMimages* runs.

Below the results are presented for the submitted runs, followed by a short description of the approaches of the participants.

6.1 Results

MMfragments Table 5 shows mean average precision scores for all submitted *MMfragments* runs, Figure 3 shows the ep/gr graphs. Judging from the fields

Table 3. Topic fields used by the submitted runs

field	<i>#MMfragments</i> runs using it	<i>#MMimages</i> runs using it
title	10	14
castitle	7	7
description	2	3
narrative	0	0

Table 4. Resources used by the submitted runs

resource	<i>#MMfragments</i> runs using it	<i>#MMimages</i> runs using it
wikipedia	15	0
wikipedia_IMG	3	16
UvAfeatures	0	3
UvAconcepts	0	1
RMIT_GIFT	2	10

and resources used as well as from the run descriptions, it appears that the top performing runs do not use any multimedia processing. These runs are based on `<title>` only queries and did not use any resources other than the wikipedia collection itself.

Table 5. Mean Average effort precision (MAep) for submitted *MMfragments* runs

group	run	MAep
QUTAU	MMfragmentstitlePSname	0.1592
QUTAU	MMfragmentstitlePS	0.1564
QUTAU	MMfragmentstitle	0.1544
QUTAU	MMfragmentstitlename	0.1536
QUTAU	MMfragmentsCASTitle	0.1168
QUTAU	MMfragmentscastitlePS	0.1147
RMIT	zet-GIFT-MMF-Mix-10	0.0656
IRIT	xfirm.MMfragments.co.06.09	0.0155
IRIT	xfirm.MMfragments.cos.09.dict2	0.0108
IRIT	xfirm.MMfragments.cos.09.dict	0.0101
RMIT	zet-GIFT-MMF-Title-10	0.0093
IRIT	xfirm.MMfragments.co.06.1	0.0086
UTWENTE	frag_art_title	0.0030
UTWENTE	frag_star_casterms	0.0009
UTWENTE	frag_star_casterms_srcmeta	0.0009

MMimages Figures 4 shows the interpolated recall precision graphs for the submitted runs for the *MMimages* task, Table 6 shows the mean average preci-

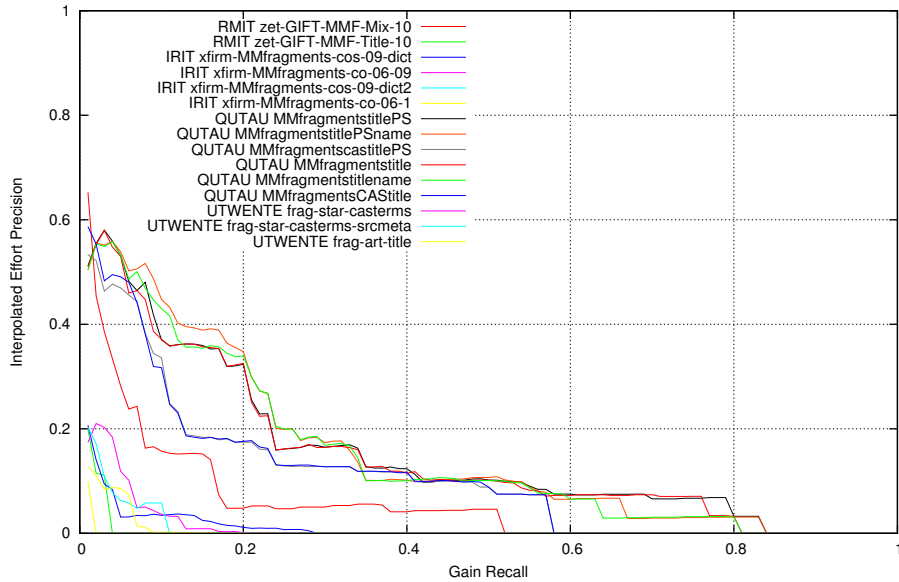


Fig. 3. MMfragments: Effort-precision/gain-recall (ep/gr), overlap not penalised

sion scores. Also for *MMimages*, the top performing runs do not use any image analysis or visual processing. They are simple text based baselines.

6.2 Participants

Below we briefly discuss the approaches taken by the groups participating in the multimedia track at INEX 2005.

CWI/UTWENTE For the *MMfragments* task CWI/UTWENTE limited their system to return only fragments that contain at least one image that was part of the multimedia collection. They did not use any further multimedia processing and experimented with traditional text based approaches. They extended the wikipedia collection with the metadata from the wikipedia_IMG collection; for each of the images in the collection, they included the text from the corresponding image in the wikipedia_IMG collection inside the `<image>` tag. For the *MMimages* task, CWI/UTWENTE experimented with different textual query variants, including adding the metadata from example images to the text query.

IRIT IRIT participated in both the *MMfragments* and *MMimages* tasks of the INEX 2006 MM track. For *MMfragments* topics, the XFIRM "Content Only" and "Content And Structure" methods intended for adhoc retrieval are used

Table 6. Mean average precision (MAP) for submitted *MMimages* runs

group	run	MAP
UTWENTE	frag_article_title	0.4040
UTWENTE	img_cas_noMM	0.3891
QUTAU	Uva_Castitle_Fusion	0.3648
QUTAU	HOT_CasTitle_06	0.3612
RMIT	zet-GIFT-MMI-Title-05	0.3174
RMIT	zet-GIFT-MMI-Title-10	0.3153
RMIT	zet-GIFT-MMI-Mix-05	0.2808
QUTAU	HOT_Title_Fusion	0.2727
RMIT	zet-GIFT-MMI-Mix-10	0.2599
QUTAU	Uva_Title	0.2440
IRIT	CASMETHOD	0.2372
RMIT	zet-GIFT-MMI-Title-00	0.2281
RMIT	zet-GIFT-MMI-Mix-00	0.2232
IRIT	ImagesMethodV2	0.2152
IRIT	ImagesMethod1.1	0.2119
IRIT	COMethod	0.1084

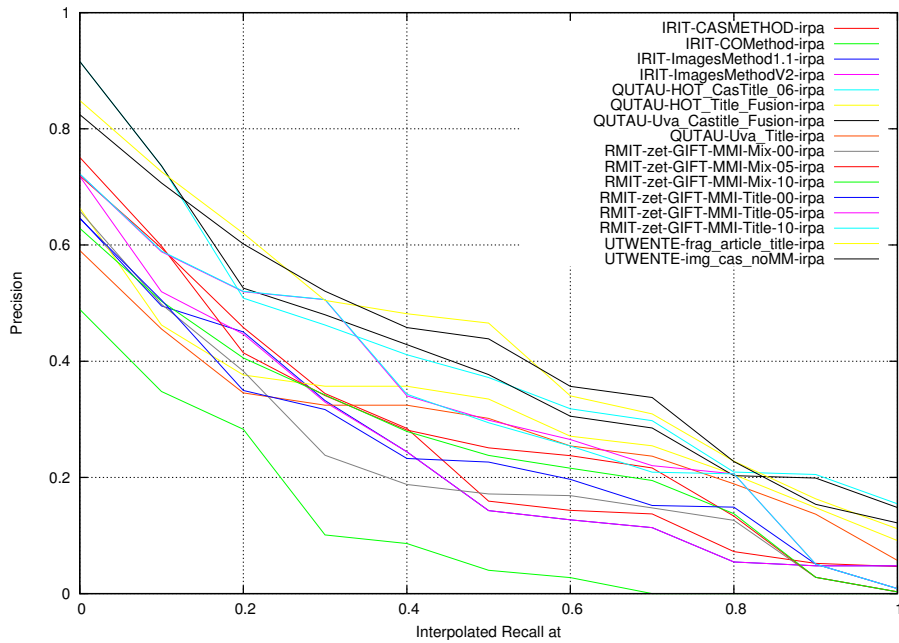


Fig. 4. *MMimages*: Interpolated Recall Precision Averages

to process queries. For *MMimages* topics, a new method using 4 sources of evidence (descendants nodes, brothers nodes, ancestors nodes and image name) is proposed to evaluate images relevance. This method is compared with the XFIRM CO and COS methods. Results show that better results are obtained by the XFIRM CO and COS methods. This can be explained by the structure of the *MMimages* collection, where only ascendant nodes can be used to evaluate image nodes relevance. Future work includes the processing of queries by example and the addition of other sources of evidence (like images classification scores and image features vectors) to evaluate image nodes relevance.

QUTAU In QUTAU's research, a text and an image-based search engines are used concurrently and the results are obtained by fusing two or more independent result sets. Therefore, the primary contribution is to answer the challenging question on how to fuse multiple results to form an optimal query results for users. This involves the fusion of the XML document retrieval results of multiple search algorithms on the same collection. Queries consist of both text (keywords) and/or example images. Two document retrieval algorithms are used: a text-based retrieval using a TF-IDF variant, and an image-based retrieval using image feature similarity.

RMIT For the *MMimages* task, RMIT used the Zettair search engine³ for XML retrieval and combined these text based results with content based image retrieval based on GIFT. The results of the two systems are fused using a simple linear interpolation. An equally weighted combination of the two systems is compared to either alone. They report slightly higher scores for the combined run over the text only run.

For *MMimages*, RMIT did not apply any image analysis. For this task, two text only runs were submitted. One <title> only run, and a run using <title>, <castitle> and <description>. The latter gave a better performance.

RSLIS RSLIS did not submit any official runs for the track, but they did help with additional assessments, and plan to use the tracks data for future studies. Their main idea is to apply Ingwersen's principle of polyrepresentation[2] to image retrieval. The hypothesis in polyrepresentation is that overlaps between the results of the most different representations are most relevant. RSLIS regards content based and text based retrieval as the most different types, since content based retrieval only takes low level features into account, while text based retrieval has the potential of capturing high level semantic information. Their aim is find out, which specific combinations of representations give the best performance. They will use results retrieved by GIFT for content based retrieval and topX NEXI queries, pointing to specific parts of the documents for retrieving a number of different text based results.

³ <http://www.seg.rmit.edu.au/zettair/>

7 Conclusions and Outlook

The INEX 2006 multimedia track, used new collections this year, based on wikipedia data. The main collections are the XML-ised wikipedia data (with images) as in the Ad Hoc track and XML versions of the metadata documents that wikipedia holds for each of these images. In addition we provided the participants with a set of resources that were either starting points for or results of visual processing. The total set of data provided makes for a nice collection of related resources.

The number of participants in the multimedia track was disappointing with only four groups submitting runs. This makes it hard to draw general conclusions from the results. What we could see so far is that the top runs in both tasks, *MMfragments* and *MMimages*, did not make use of any of the provided visual resources. More detailed analyses of the results and the participants' system descriptions is needed to see if groups managed to improve over a textual baseline using visual indicators of relevance. Also, a topic by topic analysis could shine some light. Perhaps perhaps these techniques did contribute for only a limited number of topics and hurt for others.

Another point on the future research agenda is the study of assessments and pool quality. For some topics in the pools we have duplicate and deeper assessments. This allows us to study the stability of the pool as well as inter-assessor agreement. We plan to investigate these for the final version of this paper.

For next year's multimedia track, we hope to draw more participants, from inside as well as outside INEX. The set of related collections and resources, makes this track an interesting playing ground, both for groups with a background in databases or information retrieval, and for groups with a deeper understanding of computer vision or image analysis.

References

1. L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 2006.
2. P. Ingwersen and K. Järvelin. *The Turn, Integration of Information Seeking and Retrieval in Context*, chapter 5, pages 206–214. Springer, 2005.
3. R. U. S. of Computer Science and I. Technology. Wikipedia cbir system for the multimedia track. <http://www.cs.rmit.edu.au/>.
4. C. Snoek, M. Worring, J. van Gemert, J. Geusebroek, and A. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *ACM Multimedia Conference*, 2006.
5. M. Theobald, R. Schenkel, and G. Weikum. An efficient and versatile query engine for topx search. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 625–636. VLDB Endowment, 2005.
6. A. Trotman and B. Sigurbjörnsson. Narrowed extended xpath i (NEXI). In N. Fuhr, M. Lalmas, S. aadia M alik, and Z. Szlavik, editors, *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8*,

- 2004, *Revised Selected Papers*, volume 3493. Springer-Verlag GmbH, may 2005.
<http://www.springeronline.com/3-540-26166-4>.
7. J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, C. G. Snoek, and A. W. Smeulders. Robust scene categorization by learning image statistics in context. In *CVPR Workshop on Semantic Learning Applications in Multimedia*, New York, USA, June 2006.

Kyungpook National University at INEX 2006: Interactive Track

Youngmi Jung and Heesop Kim

Department of Library & Information Science, Kyungpook National University

Daegu, 702-701, South Korea

yomjung@hanmail.net; heepsop@knu.ac.kr

Abstract. The objectives of this study are to investigate the behavior of users when interacting with elements of XML documents and experiment the effect of user characteristics and subject knowledge on search outcome in XML searching. In order to achieve the study objectives, we use a baseline system and a number of 2006 CO topics which provided by the organizers of the INEX 2006. This system is included features like implicit relevance feedback and bookmarking of elements. Twenty undergraduate students in various backgrounds participate in the experiment and need to identify documents which are relevant for completing topics. Data is collected through system logging (for example, number of query terms, query types, number of retrieved documents, precision ratio, time spent, and so on) and through several questionnaires filled by searchers, demographics, extend of subject knowledge on topics, preference on search outcome, and so on. We will use statistical analysis tool such as SPSS (Win10.0) for summarizing and describing collected data.

The Interactive Track at INEX 2006

Birger Larsen¹, Saadia Malik² and Anastasios Tombros³

¹Royal School of Library and Information Science, Denmark

²University Duisburg-Essen, Germany

³Queen Mary University of London, UK

{ blar@db.dk, malik@is.informatik.uni-duisburg.de, tassos@dcs.qmul.ac.uk }

Abstract. In this paper we describe the planned setup of the INEX 2006 interactive track. As the track has been delayed and data collection has not been completed before the workshop, the track will continue into 2007. The details, including the schedule, will be decided upon at Dagstuhl.

1 Introduction

The overall motivation for the interactive track at INEX is twofold. First, to investigate the behaviour of users when interacting with components of XML documents, and secondly to investigate and develop approaches for XML retrieval which are effective in user-based environments. The format of the track is deliberately of an exploratory nature, and has relatively broad aims rather than addressing very specific research questions. Element retrieval is still in its infancy and many basic questions remain unanswered.

As with the main ad hoc track, a major change this year is the move from the corpus of IEEE CS journal articles to the Wikipedia XML corpus of encyclopaedia articles [1]. As the wikipedia corpus is different in a number of ways, we have chosen to repeat some of the conditions studied in previous years in order to investigate if the results achieved there will also apply to the new collection. In addition, we put more emphasis on the search tasks this year and also on investigating the differences and similarities between element retrieval and passage retrieval (as recommended at the SIGIR 2006 Workshop on XML Element Retrieval Methodology¹). Finally, we have attempted to ease the burden of experimenters and searchers by an online experiment control system that handles administration and collection of electronic questionnaires, selection of tasks and logins to the search system, etc.

As in previous years, minimum participation in the track does not require a large amount of work, as a baseline system is provided by the track organisers. The bulk of the time needed will be spent on running the experiments, approximately 1.5 hours per searcher. Due to, among other things, a complex system setup, the track has been delayed and data collection has not been completed before the workshop. The track will therefore continue into 2007.

¹ See <http://www.cs.otago.ac.nz/sigirmw/> for the proceedings and presentation slides, and in particular the paper by Trotman & Geva.

1.2 Interactive track tasks

This year we offer 2 different tasks in the track for participating groups. A minimum number of searchers must be recruited for one of these, but is up to each group to decide which task they wish to participate in, or whether they will take part in both tasks. The two tasks are described in the following sections.

2 Task A – Element versus Passage retrieval

In this task, each searcher works with six search tasks in the Wikipedia collection. Two system versions will be tested: one based on Passage retrieval and one based on Element retrieval. The recruiting of minimum 6 searchers is required for participation in Task A.

2.1 Document corpus

The document corpus used in Task A is the 4.6 GB corpus of encyclopaedia articles from extracted from Wikipedia [1]. The corpus consists of more than 650,000 articles formatted in XML.

2.2 Search system

The system to be used in Task A is a Java-based retrieval system built within the Daffodil framework [2] and is provided by the track organisers. The search system interface is similar to the one used by the track in 2005 in Task A. Two system versions will be tested: one based on a Passage retrieval backend² and one on an Element retrieval backend³. Both versions have similar search interfaces - the main difference between them lies in the returned retrieval entities: The passage retrieval backend returns non-overlapping passages derived by splitting the documents linearly. The element retrieval system returns elements of varying granularity based on the hierarchical document structure. In both versions, the passages/elements are grouped by document in the result list and up to three high ranking passages/elements are shown per document (see Fig. 1 for an example of the result list in the element version of the system). The system attempts to indicate the parts of the documents that may be useful for the searcher in several ways. In the result list, selected parts are listed under each document and small icons indicate the degree of potential usefulness. The same icons are used in the overview of the document when viewing the full text. Finally, these parts are highlighted in the text of the documents – a green background colour indicates a stronger belief in the usefulness than a yellow.

² The Passage retrieval backend runs on CSIRO's Panoptic™/Funnelback™ platform. See <http://www.csiro.au/csiro/content/standard/ppsf...html> for more information.

³ The Element retrieval backend runs on Max Planck Institute for Informatics' TopX platform. See [6] for more information.

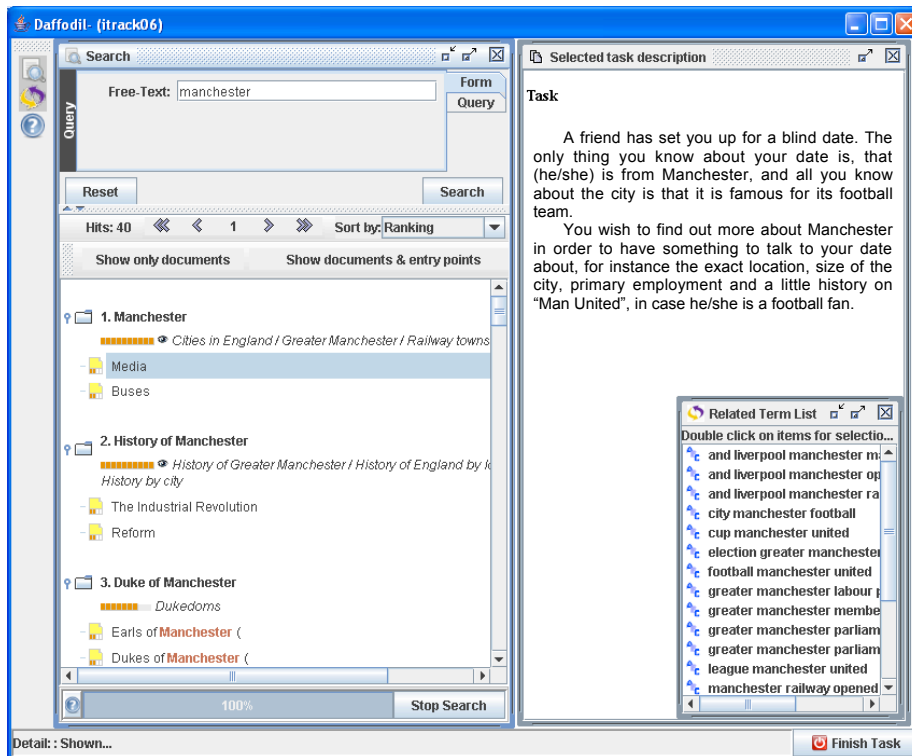



Fig. 1. Selected task, query box and result list from the INEX2006 interactive track system. (Modified version of the Daffodil Digital Library system [2])

When a searcher chooses to examine a document, both systems show the entire full text of the document with background highlighting for highly ranked passages/elements (see Fig 2). In addition to this, the element version shows a Table of Contents drawn from the XML formatting, and the passage system an overview of the retrieved passages. This also allows to highlight the currently viewed part of the document. Other parts of the document can easily be viewed by clicking a different part in the overview. Any part of the document which has already been viewed is indicated with a small eye icon ().

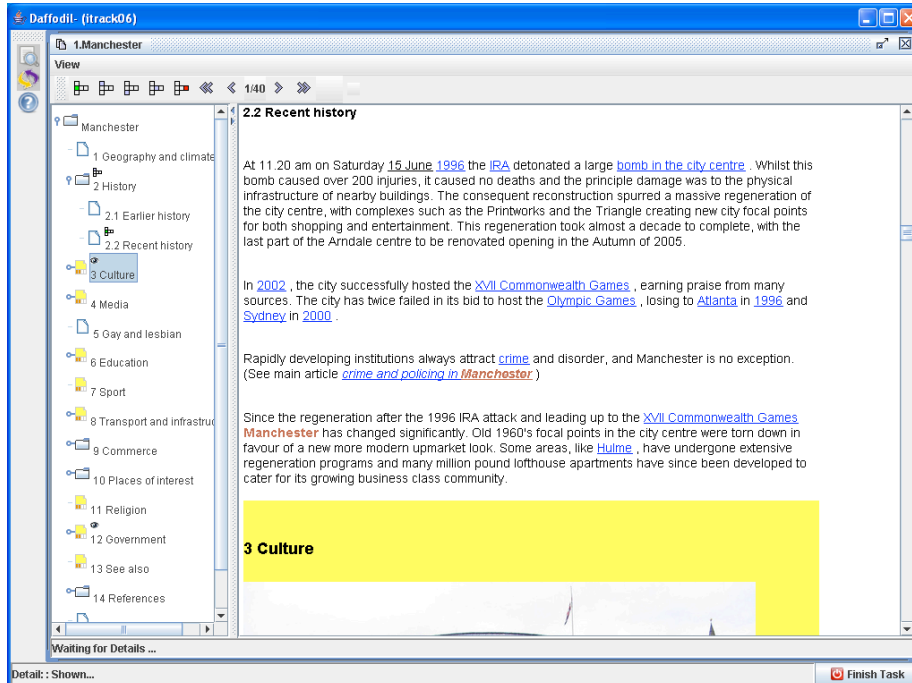


Fig. 2. Detail view of a document from the INEX2006 interactive track system. (Modified version of the Daffodil Digital Library system [2])

2.3 Relevance assessments

It is an important aspect of the study to collect the searcher's assessments of the relevance of the information presented by the system. We have chosen to use a relevance scale based on work by Pehcevski et al. [5] (and also used in Task C of last year's interactive track). This scale balances the need for information on the granularity of retrieved elements, allows degrees of relevance and is fairly simple and easy to visualise.

Searchers are asked to select an assessment score *for each viewed piece of information* that reflects the usefulness of the seen information in solving the task. Five different scores are available at the top left-hand side of the screen shown as icons:



The scores express two aspects or dimensions in relation to solving the task:

1. How much **relevant information** does the part of the document contain?
It may be *highly relevant*, *partially relevant* or *not relevant*.
2. How much **context is needed** to understand the element?
It may be *just right*, *too large* or *too small*.

This is combined into the five scores illustrated as:

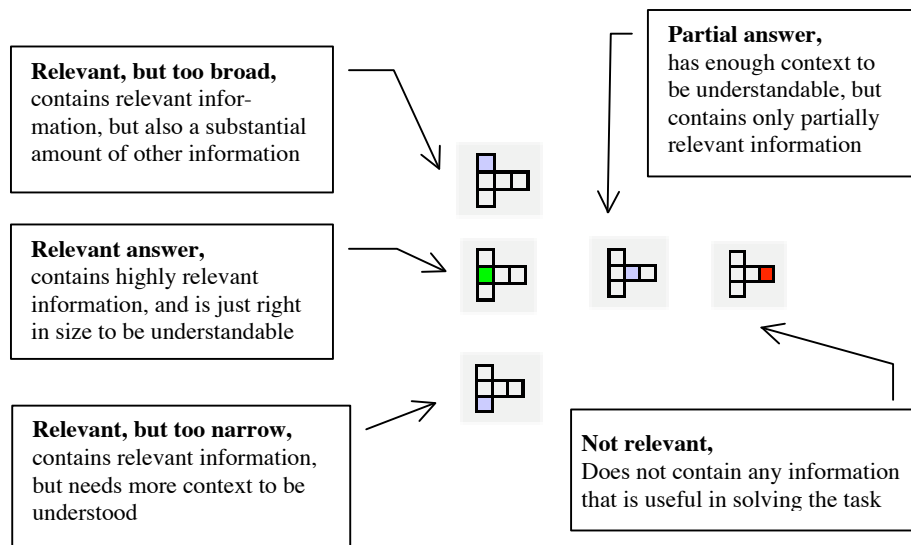


Fig. 3. INEX2006 interactive track relevance assessment scale based on Pehcevski and Thom [5].

In the interactive track the intention is that each viewed element should be assessed with regard to its relevance to the topic by the searcher. This will, however, not be enforced by the system as we believe that it may be regarded as intrusive by the searchers [4]. Please note that in contrast to the assessments made for the ad hoc track, there is no requirement on the searchers to view each retrieved element 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 searchers from interacting with the retrieved elements in a natural way.

2.4 Logging

Logs of the search sessions are saved to a database for greater flexibility and stability [3]. The log data will comprise of one session for each task the searcher carries out. For each session, the log registers the events in the session, both the actions performed by the searcher and the responses from the system. A log viewer is

available and can be a useful tool for the experimenter to monitor the progress of the searching from another computer and to spot potential problems in the experiment.

2.4 Search tasks

For the 2006 interactive track we have chosen to put more emphasis on investigating the effect of different search task types [7]. Thanks to the work of Elaine Toms (Dalhousie University, Canada), Luanne Freund (University of Toronto, Canada) and Heather L. O'Brien (Dalhousie University, Canada) we have a multi-faceted set of 12 tasks this year with three task types (*Decision making*, *Fact finding* and *Information gathering*) further split into two structural kinds (*Hierarchical* and *Parallel*). See Appendix A for the tasks and more information about them.

The 12 tasks are split into six categories allowing the searchers a choice between two tasks, and at the same time ensuring that each searcher will perform one of each type and structure. This allows the topic to be more “relevant” and interesting to the searcher. Because of the encyclopaedic nature of Wikipedia with most topics concentrated in a few documents, we have chosen to allow fairly short time to solve each task and instead have each searcher tackle more tasks. A maximum time limit of 10 minutes will apply. Sessions can finish before this if searchers feel they have completed the task.

2.5 Experimental design

Contrary to previous years we will use an online experimental control system in an attempt to ease the burden of experimenters and searchers. The control system handles administration and collection of electronic questionnaires, selection of tasks, login to the search system etc. The experimenter thus needs only to log in once per searcher.

2.6 Experimental matrix

A minimum of 6 searchers from each participating group need to be recruited. Each searcher will search on one simulated work task from each category (chosen by the searcher). The order in which task categories are performed by searchers over the two system versions will be permuted in order to neutralise learning effects. This means that one complete round of the experiment requires 6 searchers.

For information, the basic experimental matrix looks as follows:

	S1	S1	S1	S2	S2	S2
Rotation 1	I-H	F-H	D-H	D-P	F-P	I-P
Rotation 2	F-P	I-P	D-P	D-H	I-H	F-H
Rotation 3	F-H	D-H	I-H	I-P	D-P	F-P

	S2	S2	S2	S1	S1	S1
Rotation 4	I-H	F-H	D-H	D-P	F-P	I-P
Rotation 5	F-P	I-P	D-P	D-H	I-H	F-H
Rotation 6	F-H	D-H	I-H	I-P	D-P	F-P

Where:

Element (S1) vs. Passage (S2) systems

Decision (D) vs. Fact (F) vs. Info (I) tasks

Hierarchical (H) vs. Parallel (P) tasks

These rotations will be related to the searcher logins and the control system will handle their administration.

2.7 Experimental procedure

The experimental procedure for each searcher is outlined below.

1. Experimenter briefs the searcher, and explains format of study. The searcher reads and signs the Consent Form
2. The experimenter logs the searchers into the control system. Tutorial of the system is given with a training task given by the system, and the experimenter hands out and explains the System features document
3. Any questions answered by the experimenter
4. The control system administers the Before-experiment Questionnaire
5. *Task descriptions for the first category administered, and a task selected*
6. *Before-each-task questionnaire administered*
7. *Task begins by clicking the link to the system. Max. duration 10 minutes, with reminder by the system.*
8. *After-each-task questionnaire administered*
9. Steps 5-8 repeated for the five other tasks
10. Post questionnaire administered.

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, the work on the tasks and completion of questionnaires should be performed in one, continuous session in an undisturbed environment.

A Consent Form gives information to the searchers about the experiment and their role in it. Basic information about system information and how to assess elements for relevance are given in the System features document.

3 Task B - Own element retrieval system or own interface

This task allows groups to test either:

- I. Their own fully working element retrieval system, or
- II. Interfaces for element retrieval using the TopX element retrieval as a backend.

For both I and II some work is needed from the participating group. The track organisers can provide, e.g. the search tasks, questionnaires, the experimental matrix etc. from Task A. It would also be possible to use Daffodil as a baseline with either the Element or Passage retrieval versions. For groups that do not have a fully functioning element retrieval system, option II allows to concentrate on building an element retrieval interface and use the TopX engine as a backend. A simple API is available for TopX that can accept queries and return a set of ranked elements. A large variety of element retrieval interface features could be built on top of this. This allows a large range of issues related to element retrieval be investigated depending on the goals of any individual groups. Groups that participate in Task B do not have to recruit searchers for Task A.

The scope in Task B is therefore different, with much larger degree of freedom for participating groups. The experimental procedure from Task A may be altered and modified to fit the requirements of the local groups and the issues they want to explore. In addition, there is no requirement that logs and other data must be submitted to the organisers as in Task A.

4 Concluding remarks

As the track has been delayed and data collection has not been completed before the workshop the track will continue into 2007. The details and the schedule will be decided upon at Dagstuhl.

References

1. Denoyer, L., Gallinari, P. (2006): The Wikipedia XML corpus. *SIGIR Forum*, 40(1):64-69.
2. Fuhr, N., Klas, C.P., Schaefer, A., Mutschke, P. (2002): Daffodil: An integrated desktop for supporting high-level search activities in federated digital libraries. In *Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, p. 597-612.
3. Klas, C.P., Albrechtsen, H., Fuhr, N., Hansen, P., Kapidakis, S., Kovács, L., Kriewel, S., Micsik, A., Papatheodorou, C., Tsakonas, G., Jacob, J. (2006): A logging scheme for comparative digital library evaluation. In *Proceedings of the 10th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, p. 267-278.
4. Larsen, B., Tombros, A. and Malik, S. (2005): Obtrusiveness and relevance assessment in interactive XML IR experiments. In: Trotman, A., Lalmas, M. and

- Fuhr, N. eds. *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, held at the University of Glasgow*. Dunedin (New Zealand): Department of Computer Science, University of Otago, p. 39-42.
5. Pehcevski, J., Thom, J. A. and Vercoustre, A.M. (2005): Users and assessors in the context of INEX: Are relevance dimensions relevant? In: Trotman, A., Lalmas, M. and Fuhr, N. eds. *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, held at the University of Glasgow*. Dunedin (New Zealand): Department of Computer Science, University of Otago, p. 47-62.
 6. Theobald, M., Schenkel, R., Weikum, G. (2005): An efficient and versatile query engine for TopX search. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, p. 625-636.
 7. Toms, E.G., Freund, L., Kopak, R., Bartlett, J.C. (2003): The effect of task domain on search. In *Proceedings of the 2003 Conference of the Centre for Advanced Studies on Collaborative Research*, p. 303-312.

Appendix A – Search Tasks

In this year’s track, 12 search tasks that are to be performed with the Wikipedia dataset. The tasks are loosely based on INEX 2006 topics. They are, however, modified versions of INEX topics to ensure that they:

- are not limited in computer science topics only
- are not of limited regional interest
- are not too simple or too complex (i.e. require more than a simple page in order to be answered, or do not have too many relevant elements)

The twelve tasks are split into three different types:

- **Fact finding**, where the objective is to find "specific accurate or correct information or physical things that can be grouped into classes or categories for easy reference."
- **Information gathering**, where the objective is to collect miscellaneous information about a topic
- **Decision making**, where the objective is to select a course of action from among multiple alternatives

The tasks are also split into two categories, depending on the “structure” of the search task:

- **Parallel**, where the search uses multiple concepts that exist on the same level in a conceptual hierarchy; this is a breadth search (and in a traditional Boolean likely was a series of OR relationships)
- **Hierarchical**, where the search uses a single concept for which multiple attributes or characteristics are sought; this is a depth search, that is a single topic explored more widely

Each task also has an associated **domain**, which is the broad subject area to which a topic belongs.

Based on these classifications, we can represent the tasks for the track in the following matrix:

ID	Task	Domain	Type	Structure
1	Your community is contemplating building a bridge across a span of water measuring 1000 M in order to ease traffic congestion. There will be a presentation this evening about the type of bridge proposed for the project. To date, many types of bridges have been discussed: "folding bridge," "suspension bridge," "retractable bridge," and "bascule bridge". In order to be well informed when you attend the meeting, you need information on what type of bridge would best suit the community's needs, bearing in mind that the solution must accommodate vehicles and be sturdy enough to withstand a body of water that can be rough and frozen over with ice in winter.	Engineering	Decision Making	Hierarchical
2	Your friends who have an interest in art have been debating the French Impressionism exhibit at a local art gallery. One claims that Renoir is the best impressionist ever, while the other argues for another. You decide to do some research first so you can enter the debate. You consider Degas, Monet and Renoir to construct an argument for the one that best represents the spirit of the impressionist movement. Who will you choose and why?	Art	Decision Making	Hierarchical
3	As a tourist in Paris, you have time to make a single day-trip outside the city to see one of the attractions in the region. Your friend would prefer to stay in Paris, but you are trying to decide between visiting the cathedral in Chartres or the palace in Versailles, since you have heard that both are spectacular. What information will you use to make an informed decision and convince your friend to join you? You should consider the history and architecture, the distance and different options for travelling there.	Travel	Decision Making	Parallel

4	As a member of a local environmental group who is starting a campaign to save a large local nature reserve, you want to find some information about the impact of removing the trees (logging) for the local pulp and paper industry and mining the coal that lies beneath it. Your group has had a major discussion about whether logging or mining is more ecologically devastating. To add to the debate, you do your own research to determine which side you will support.	Environment	Decision Making	Parallel
5	A friend has just sent an email from an Internet café in the southern USA where she is on a hiking trip. She tells you that she has just stepped into an anthill of small red ants and has a large number of painful bites on her leg. She wants to know what species of ants they are likely to be, how dangerous they are and what she can do about the bites. What will you tell her?	Natural science-Health	Fact Finding	Hierarchical
6	You enjoy eating mushrooms, especially chanterelles, and a friend who is an amateur mushroom picker indicates that he has found a good source, and invites you along. He warns you that chanterelles can be confused with a deadly species for which there is no known antidote. You decide that you must know what you are looking for before you going mushroom picking. What species was he referring to? How can you tell the difference?	Food	Fact Finding	Hierarchical
7	As a history buff, you have heard of the quiet revolution, the peaceful revolution and the velvet revolution. For a skill-testing question to win an iPod you have been asked how they differ from the April 19th revolution.	History	Fact Finding	Parallel
8	In one of your previous Web experiences, you came across a long list of castles that covered the globe. At the time, you noted that some are called castles, while others are called fortresses, and Canada unexpectedly has castles while Denmark has also fortresses! So now you wonder: what is the difference between a fortress and a castle? So you check the Web for a	History-Travel	Fact Finding	Parallel

	clarification, and to find a good example of a castle and fortress in Canada and Denmark.			
9	A close friend is planning to buy a car for the first time, but is worried about fuel costs and the impact on the environment. The friend has asked for help in learning about options for vehicles that are more fuel efficient and environmentally friendly. What types of different types of engines, manufacturers and models of cars might be of interest to your friend? What would be the benefits of using such vehicles?	Cars	Info gathering	Hierarchical
10	You recently heard about the book "Fast Food Nation," and it has really influenced the way you think about your diet. You note in particular the amount and types of food additives contained in the things that you eat every day. Now you want to understand which food additives pose a risk to your physical health, and are likely to be listed on grocery store labels.	Food	Info gathering	Hierarchical
11	Friends are planning to build a new house and have heard that using solar energy panels for heating can save a lot of money. Since they do not know anything about home heating and the issues involved, they have asked for your help. You are uncertain as well, and do some research to identify some issues that need to be considered in deciding between more conventional methods of home heating and solar panels.	Home heating	Info gathering	Parallel
12	You just joined the citizen's advisory committee for the city of St. John's, Newfoundland. With the increase in fuel costs, the city council is contemplating supplementing its power with alternative energy. Tidal power and wind power are being discussed among your fellow committee members. As you want to be fully informed when you attend the next meeting, you research the pros and cons of each type.	Energy	Info gathering	Parallel

Context revisited - element retrieval behaviour and genre dependency

Ragnar Nordlie, Nils Pharo

Faculty of Journalism, Library and Information Science, Oslo University College, Norway
{ragnar.nordlie, nils.pharo}@jbi.hio.no

Extended abstract

For the last two years, we have conducted user studies to investigate the implications of one fundamental proposition underlying XML-assisted retrieval, at least as understood in the INEX context, namely that users of information retrieval systems are better assisted if they are presented with search results which provide primary access to relevant parts of the documents rather than or in addition to the full text of relevant documents. The validity of this proposition depends on at least two factors: that users are actually able to identify relevance upon being shown parts of documents at various levels of granularity; and if so, that they behave accordingly, i.e. that they will not feel compelled, for some reason, to view the whole article even if they have been shown and have looked at the relevant parts. In studies based on data from the 2004 and 2005 INEX Interactive Track we have used various approaches towards investigating our hypotheses: that users are more assured in their relevance judgements when they are based on full articles, and that their preferred search behaviour includes verifying relevance judgements by looking at the context of the viewed element.

In our paper “Context matters: an analysis of XML documents”, we investigated the relevance assessments of the users in the 2004 INEX Interactive experiments, expressed in a three-part scale of “usefulness” and “specificity”, respectively. We found that users looked relatively more frequently at whole articles than at elements of articles, that they were more likely to judge a full article “useful” than an element of the same article, and that there was no corresponding tendency to judge an element more “specific” than the full article. Overall, there was a general tendency to judge elements as more relevant than full articles, but this did not hold for judgements of elements and the full text of the same article.

A major weakness in the 2004 study was that users had apparent difficulties in dealing with the combined relevance measure that were used, so that the relevance judgements were unreliable. The 2005 user assessments made use of a single measure of relevance, providing a clearer picture of users’ ability to judge relevance regardless of level of granularity. We based our study on an investigation of

- what text elements users initially ask to see from their result list
- what text elements users choose to look at in their interaction with the search result

- in what sequence they choose to look at the elements
- how consistent they are in their judgements of elements on different levels.

We found that users overwhelmingly seem to choose the full article as their primary entry point; that given an opportunity to browse the search results on varying levels of granularity, they look at the elements on different levels with a frequency relatively proportional to the elements' frequency in the collection, but that users agree on their relevance judgements far more in the case of full articles than when they judge elements on higher level of granularity; and that there is a strong tendency to access the full document as the last element when several elements from the same article is accessed, regardless of the relevance scores awarded to the lower-level elements. It seems that giving users access to the most relevant text elements on lower levels of granularity is valuable, but only if the full article is present to provide context.

Both the 2004 and 2005 INEX interactive experiments have used a database consisting mainly of traditional scientific articles, with a length and structure typical of their genre. The 2006 experiments, conducted on a database of Wikipedia articles, will provide an opportunity for studying the genre-dependency of our previous findings. Shorter articles and more fact-oriented queries may have a strong influence on user behaviour. We will also be able to study the effect of topic familiarity on both relevance judgements and navigation behaviour. A combined study of users' relevance assessments and actual search behaviour on documents from distinctly different digital genres will provide a rich background for determining the potential usefulness of a retrieval system based on XML-partitioned documents.

Evaluating Tasks by Type and Form

Elaine Toms¹, Luanne Freund², Chris Jordan¹,
Tayze Mackenzie¹, Sandra Toze¹ and Heather O'Brien¹

¹ Centre for Management Informatics, Dalhousie University
Halifax, Nova Scotia, Canada

² Faculty of Information Studies, University of Toronto
Halifax, Nova Scotia, Canada

{etoms, sandra.toze, hlobrien}@dal.ca; luanne.freund@dal.ca; chris.jordan@acm.org

Abstract. The purpose of this research is to assess the affect of selected task characteristics on search outcomes.

Keywords: tasks, information searching,

1 Introduction

A long term goal of our research is to examine the influence of context on search. While context can include many characteristics of users, their work environment, the work tasks they are performing, and the resources that are available to them, in this research we focus on one characteristic: the work task.

While there are multiple views on conceptualizing task (see Campbell, 1988), we prefer the definition of Byström & Hansen (2005): a task is defined as a particular item of work with a recognizable beginning and end, a meaningful purpose and a practical goal. In addition to purpose and goal, tasks have many attributes that are either conceptual in nature such as complexity, specificity, and clarity, or structural in physical form such as hierarchical versus parallel, attribute versus alternative, number of concepts or dimension represented, and number of negative elements, i.e., concepts that are not pertinent to outcomes. While tasks tend to be unique to their environment, e.g., software engineering (Freund, Toms and Waterhouse, 2005) and bioinformatics (Bartlett & Toms, 2005), little research has explored the conceptual and/or structural differences among tasks and their effect on the search process and search outcomes. Notably, Trifts and Toms (2006) found that tasks classified as Attribute (those in which people look for and compare characteristics of objects, e.g., RAM, disk size, processor), and those classified as Alternative (those in which people look for and compare different objects, e.g., Apple versus a Windows computer) are treated very differently by users.

In this research, we isolated two characteristics of task, one that focuses on purpose of the task and one on the structure of the task. Do the characteristics of the task

influence user behaviour? Is there a relationship between attribute(s) and search outcome? Can task characteristics partially predict search outcome?

2 Methods

2.1 Overview

To respond to the research questions, we developed a combined within-and between-subject human experiment. Forty-eight people will search for 6 search tasks using a customized interface to Wikipedia.

2.2 Task Types

1. Structure: characteristics of the *format* of the task
 - a) Hierarchical: task is a single concept for which multiple attributes or characteristics are sought; this is a depth search in which a single topic is explored more broadly
 - b) Parallel: task contains multiple concepts that exist on the same conceptual level of a hierarchy; this is a breadth search.
2. Type: different *objectives* of the task
 - a) Fact finding: the objective is to locate specific details;
 - b) Information gathering: the objective is to collect miscellaneous information about a topic;
 - c) Decision making: the objective is to select a course of action from among multiple alternatives.

Twelve tasks were used in the study: 6 hierarchical and 6 parallel, and four each for the three levels of Type. The tasks were also balanced so that each of the three levels of Type included a even mix of the levels of Structure. In addition, the tasks were classified by Domain, e.g., engineering, art and travel, but domain was not controlled.

2.3 Metrics

The performance of the tasks was measured in several ways: a) user satisfaction as measured by a series of questions asked on completing each task, b) efficiency (e.g., time taken to do the task, number of queries submitted), and c) effectiveness (e.g., completeness of the answer). In addition, the search outcomes were characterized in a number of ways, and the relationships among those attributes and task variables were explored.

2.4 Procedures

This study was conducted in the lab. Participants interacted with a WiIRE like system that is in essence a participant self-serve system that runs off the Web. Each participant ran through the same routine:

1. introduction to the study,
2. consent to participate,
3. a tutorial on the system,
4. task assignment and pre-task questions
5. time (ca. 10 minutes) to complete the task which included finding the appropriate page(s), and drafting a likely answer
6. post-task questions about the experience
7. post session questionnaire

All data was captured to a database, and a log file.

2.5 Status

Some data will be collected prior to the December meeting and the remainder in January.

A Taxonomy for XML Retrieval Use Cases

Miro Lehtonen¹, Nils Pharo², and Andrew Trotman³

¹ Department of Computer Science, University of Helsinki, Finland
Miro.Lehtonen@cs.Helsinki.Fi

² Faculty of Journalism, Library and Information Science, Oslo University College,
Norway

nils.pharo@jbi.hio.no

³ Department of Computer Science, University of Otago, Dunedin, New Zealand
andrew@cs.otago.ac.nz

Abstract. Despite the active research on XML retrieval, it is a great challenge to determine the contexts where the methods can be applied and where the proven results hold. Therefore, having a common taxonomy for the use cases of XML retrieval is useful when presenting the scope of the research. The taxonomy also helps us design more focused user studies that have an increased validity. In the current state, the taxonomy covers most of common uses of applying Information Retrieval to XML documents. We are delighted to see how some of the use cases match the tasks of the INEX participants.

1 Introduction

There are several approaches to creating a hierarchical classification for the use cases, depending on which features or facets we consider important. The factors that distinguish use cases from each other include the following:

XML A continuum from marked-up text including articles, books, web pages to structured data extracted from a database. The element names shift from presentation-specific names towards metadata-like content descriptors as we move marked-up text towards the other end of the continuum.

Content Books and other big documents, chapters, articles, web pages, document fragments and other incomplete documents. The size of the document usually makes a difference when presented to the user. Some are too big, whereas others may be too small, and the system has to work accordingly.

Queries Content-Only (CO), Content-And-Structure (CAS). Depending on the kind of XML, the document structure can be present in the queries as structural hints or requirements. Allowing XML structure in the queries usually requires user interfaces that support such querying.

Information need Answer to a question (QA), topic description, topic-based inventory of the document collection.

Presentation of search results Links to standalone documents, best entry points, aggregated/assembled documents, heatmaps.

2 Hierarchical classification for use cases

One of the most important characteristics of XML is its ability to describe content with metadata which is a part of the markup. The tags around words, phrases, paragraphs, and even whole documents, define the interpretation of the enclosed content — only the interpreters vary. The purpose of the XML markup is the deciding factor at the top level of the hierarchy, so that in Class A, XML is computer-interpreted for display, in Class B, XML is human-readable, and in Class C, XML is interpreted and further processed by computer.

A Layout-oriented document types

XML documents of this nature are often conversions from other formats. In the past it has been common for academic publishers to use automated programs to generate the XML from typesetting files, whereas the legacy documents of enterprises are conversions from word processor formats or even HTML [1]. However, XML in both cases is the archival format: the XML “variant” of the document is definitive and all other formats are derivative.

XML: Most tag names describe either 1) the presentation of the content, e.g., `<i>` for italics, `` for boldface, or 2) document structure, e.g., `<p>` for paragraph, `<sec>` for section. Differences between document types (DTDs, XML Schemas) are not a great challenge for IR applications where finding the relevant content comes first and publishing it comes later.

Queries: Keywords, keyphrases (Content-Only, CO). Including structural hints in the queries is not meaningful as far as element names are concerned. Layout and structure-related element names do not represent any information need as they are not related to the topic of any CO query.

Presentation of search results: The document order of the original documents is significant, so that the content of a single document or fragment cannot be reordered.

The size of the document is the second most distinctive feature in the class of layout-oriented documents as it directly affects the presentation of the search results as well as the overall system tasks.

A.1 Book search

One of the advantages of using XML-IR for book search (also for article search in A.2) is that the documents are long and XML-IR can locate relevant fragments for the user — reducing the load on the user.

Example document types:

- Docbook⁴.

⁴ <http://www.docbook.org/>

Presentation of search results:

Example documents:

- Relevant chapters or sections of the book as standalone documents,
- Links (with summaries) to the full-text of whole books where the relevant content is highlighted.

User tasks: Learning and reading about a topic until information need is satisfied.

System tasks: Identifying relevant passages and best entry points.

A.2 Article search

Example documents:

- Web pages,
- IEEE journals in XML format (part of the INEX test suite 2002-2005).

Presentation of search results: Links (with summaries) to the full-text of whole articles. Optionally, the relevant content is highlighted.

User tasks: Learning and reading about a topic until information need is satisfied.

System tasks: Identifying relevant articles, relevant passages, and best entry points.

A.3 Fragment search

Example documents:

- Single-sourced documentation.

The source documents are typically too small or too dependent on other documents (incomplete) to be published on their own.

Presentation of search results: Customised according to fragment size, content, and final publication format.

User tasks: Finding all relevant fragments to be included in a publication, e.g. technical document, or to be used as background material.

System tasks: Identifying relevant fragments and combining them into whole documents.

B Content-oriented document types

XML: The majority of the non-optional tag names describe the text content of the corresponding elements, e.g., `<action>`, `<reason>`, `<recommendation>`. Differences between document types (DTDs, XML Schemas) have a direct impact on the query evaluation which may require manual mappings between incompatible structures. Automatic and ontology-based methods are also worth considering.

Queries: Keywords, keyphrases, and structure (Content-And-Structure, CAS). Without structural hints in the query, the results may be highly ambiguous, because the interpretation of the content is dependent on the XML names.

Presentation of search results: The relevant results are rarely presented in the original order even when they come from the same document. For each relevant answer, the content is ordered by rules or templates. Each XML document, in turn, may hold more content than what would ever be published as a single publication.

The second most distinctive feature of content-oriented document types is the level where the content is described. If the content-oriented element names are mostly at the level of paragraphs and sections, for example, `<instructions>` or `<weatherforecast>`, we may apply methods for *semantic search* [2] to the documents (B.1). If the content is mostly described at the inline-level, the typical element names include `<city>`, `<person>`, and `<code>`. The most appropriate style of search is then for entities (B.2). If all the element names describe the content, we are ready for data retrieval (B.3).

B.1 Semantic search

Example documents:

- Documentation with descriptive element names (metadata) and full-text content,
- The Guideline Elements Model (GEM) [3].

Presentation of search results:

- The relevant parts of the original documents reformatted according to the query.
- Standalone XML fragments including the relevant content which is extracted from the source document.

User tasks: Finding topic descriptions.

System tasks: Identifying relevant elements, vague matching of content and structural search conditions, organising the matches into whole documents.

B.2 Entity search

Example documents:

- Annotated documents

Presentation of search results: Standalone fragments, possibly formatted.

User tasks: Question asking.

System tasks: Question Answering: Identifying relevant elements, duplicate detection, vague matching of data (target elements, element and attribute names) and structural search conditions on them.

B.3 Data retrieval

Thanks to its interoperability, XML is the appropriate interchange format, for example between two different (relational) database formats. Consequently, most databases provide an XML view of the data that is equivalent to the native representation format and available for querying.

Example documents:

- XML databases, relational data for XML.

Presentation of search results: Template-driven formatting.

User tasks: Database publishing, template-driven document assembly.

System tasks: Identifying relevant elements.

C Process-oriented document types

What is typical of searching process-oriented documents is that the query is matched against one part of the XML document (metadata), whereas other parts are expected as answers (data).

C.1 Multimedia search

Example document types:

- SVG, SMIL.
- Other documents that may contain multimedia

User tasks: Finding multimedia.

System tasks: Ranking multimedia by metadata descriptions.

C.2 Feed search

The first implementations of feed search date back to 2003 when Feedster launched their online system⁵.

Example document types:

- RSS, OPML.

User tasks: Finding relevant postings.

System tasks: Indexing and querying streaming XML, monitoring and filtering XML feeds.

3 Conclusion

Creating the hierarchical classification for the use cases of XML retrieval is only a first step in developing the taxonomy. The next step is to describe each class in more detail, e.g., by identifying the challenges and opportunities of each class. This work is on-going and the authors appreciate any feedback and contributions that advance this work.

References

1. Chidlovskii, B., Fuselier, J.: Supervised learning for the legacy document conversion. In: DocEng '04: Proceedings of the 2004 ACM symposium on Document engineering, New York, NY, USA, ACM Press (2004) 220–228
2. Chu-Carroll, J., Prager, J., Czuba, K., Ferrucci, D., Duboue, P.: Semantic search via XML fragments: a high-precision approach to IR. In: SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (2006) 445–452
3. ASTM International: E2210-02 Standard Specification for Guideline Elements Model (GEM)-Document Model for Clinical Practice Guidelines. (2003)

⁵ <http://www.feedster.com/>

Inter-assessor agreement at INEX 06

Nils Pharo¹, Andrew Trotman², Shlomo Geva³, Benjamin Piwowarski⁴

¹Faculty of Journalism, Library and Information Science, Oslo University College, Norway
nils.pharo@jbi.hio.no

²Department of Computer Science, University of Otago, Dunedin, New Zealand
andrew@cs.otago.ac.nz

³Faculty of Information Technology, Queensland University of Technology, Australia
s.geva@qut.edu.au

⁴Department of Computer Science, University of Chile
bpiwowar@dcc.uchile.cl

Extended abstract

A basic requirement of IR test collection is to have a pool of documents with relevance assessments. Such relevance assessments are typically performed by topic experts. The procedure at INEX is having the participating groups submitting topics which are later returned to topic creators for relevance assessment. Usually the participants perform relevance assessments of their own topics, but some topics are assessed by more than one assessor.

At the INEX 06 workshop an experiment was conducted in order to measure inter-assessor-agreement. Since the INEX 06 collection consists of articles from Wikipedia it is relatively easy to design topics (simulated work tasks) of general interest, thus making it easier for other than topic creators to perform relevance judgments. Participants at the workshop were asked to perform relevance assessments of a selection of topics, which were logged in the online assessment system. The assessors also answered questionnaires related to the experiment.

The main purpose of this experiment was to measure the level of agreement between many assessors judging the same topics. We present the findings and relate them to background factors.

The findings from our analysis can be used for several purposes: 1) they can help us design the procedures for relevance assessments in future INEX experiments; 2) they provide a valuable understanding of the general factors influencing relevance judgements used in test collections; and 3) they can be used to argue for the validity of using the test collection approach as a method for testing the retrieval efficiency of IR systems.

XML-IR Users and Use Cases

Andrew Trotman¹, Nils Pharo², Miro Lehtonen³

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

²Faculty of Journalism, Library and Information Science, Oslo University College, Norway
nils.pharo@jbi.hio.no

³Department of Computer Science, University of Helsinki, Finland
miro.lehtonen@cs.helsinki.fi

Abstract. We examine the INEX *ad hoc* search tasks and ask if (or not) it is possible to identify any existing commercial use of the task. In each of the tasks: thorough, focused, relevant in context, and best in context, such uses are found. Commercial use of CO and CAS queries are also found. Finally we present abstract use cases of each *ad hoc* task. Our finding is that XML-IR, or at least parallels in other semi-structured formats, is in use and has been for many years.

Introduction

Many of the teething troubles in setting up realistic XML-IR experiments have been blamed on the lack of user grounding. Trotman [17] suggests that the standard methodology for XML-IR experiments should be based on examining user behavior and that it is essential to identify such a group of users. Unfortunately, he stops there and does not identify a single user.

At INEX 2006 Trotman and Pharo ran a thought experiment called the User Case Studies Track [20]. They asked participants to hypothesize users of XML-IR and to imagine how such users would interact with an information system. They assumed such users did not exist and would not exist for several years.

Almost immediately Dopichaj [6] identified the use of XML-IR in book search and separately Lehtonen [11] identified commercial systems dating back to several years before the first INEX workshop. In the words of Lehtonen “Rather than trying to find the users, we are tempted to ask an even more interesting question: How did we lose sight of the users of XML retrieval?”.

In this contribution we examine the existing INEX *ad hoc* tasks and demonstrate that for each track there is already a commercial use – they are already grounded in a user base. We take those search engines identified in the studies of Dopichaj and of Lehtonen, and connect the methodology of INEX. We then present hypothetical use cases for each INEX task.

Ironically, just as Dopichaj and Lehtonen were demonstrating the long-term prior existence of XML-IR users, Trotman was suggesting that users might prefer passages to elements [18] and emphasizing the importance of Passage Retrieval. We do not

believe that the method of identifying the semantic unit is of material value in the case studies herein – that is, we do not believe the user is concerned with whether they are presented with a series of paragraphs or an element.

INEX *ad hoc* tasks

INEX initially identified only one *ad hoc* task, the thorough task. Subsequent rounds added the focused task, then the fetch & browse task (which became the relevant in context task). Finally the best in context task was added in 2006. The best in context task (also known as the best entry point (BEP) task) was one of the tasks studied in the Focus experiments [10] prior to the first INEX, and the task for which it could be easiest for the established search engines to adopt on the web.

Thorough Retrieval

The purpose of the thorough task is to identify all relevant elements in the document collection and to rank those relative to each other. That is, regardless of nesting, and regardless of the document in which the element has been identified, the task is to rank all elements with respect to topical relevance.

At the beginning of INEX 2006 there was no absolute certainty of the credibility of the task to a user community. It is clearly stated that “there are no display-related assumptions nor user-related assumptions underlying the task” [5]. However, as soon as these assumptions are made, the task may reduce to finding BEPs or to focused retrieval.

The task continues because it is considered important by some participants as a benchmark for improvements [18]. It is the only task that has been held at every INEX. It is also considered by some to be a system oriented approach to XML element ranking: from the thorough set of relevant elements, a set of disjoint elements (for focused retrieval) could be chosen, or a set of good entry points into a document could be identified.

Focused Retrieval

An endeavor to create a more user-centered task resulted in a task in which overlapping elements were forbidden from occurring in the result list. Evidence from the INEX Interactive track supported the view that users did not want to see the same text repeated in their results lists [16]. Adding the exclusion of overlap from the results list added a new problem to the search strategy, the identification of the *just right* sized element to return to the user.

At INEX 2006, and for the focused task, a hypothetical user of an XML-IR system was described. This user views the results list top-down, they prefer smaller elements to larger ones, and they are mostly concerned with elements ranked highly in the results list [5], they also do not want to see the same information returned multiple times.

The focused task has been criticized for being context-free [14] – that is, the user is presented with the results of their search but cannot evaluate the authority of the information because no context is given. As an example, a section of an academic paper might fulfill the user’s information need, but without knowing who wrote the paper (and were it was published) the user can’t be certain of the accuracy of the content. The context is important, and is needed.

Relevant In Context (Fetch & Browse)

In the fetch & browse task the search engine must first identify relevant documents and rank these relative to each other and with respect to topical relevance. Within this ranking, those relevant elements within the document are identified. The subtle difference between relevant in context and fetch & browse is that the former is focused whereas the latter is thorough.

At INEX 2006 a real user was imagined. This user considers the document to be the natural unit of retrieval, but wants the relevant elements within the document identified for quick perusal and browsing [5] – perhaps by subtle highlighting.

Best in Context

Considering how an information system using the relevant in context strategy might be implemented leads to the final task. The user performs a search, chooses a result from the results lists, and expects to be taken to the relevant passage within the chosen document. Again, at INEX 2006 a real user was theorized [5].

This task was previously investigated in the Focus experiments [10]. There they asked assessors to identify the best entry point from which a reader should start reading in order to satisfy their information need.

Theoretical Tasks

Since the first INEX workshop tremendous effort has been made to identify a realistic task for XML-IR, and it is hard to argue such a task has been found. We believe that relevant in context would be of enormous interest if a web search engine provider chose to implement similar technology for HTML on their cached results. Best in context might be easily implemented in a search engine that considered anchors inside a web page as well as whole web pages. Alternatively, if and when semantically and structurally stronger XML-languages are widely adopted in place of HTML, such technology would be valuable. Although XHTML is based on XML rather than SGML, the web still is based on quite a weak markup language, and sadly search engines do not identify relevant parts of documents.

This leaves a gap. INEX has identified four tasks; three are believed to be user driven. But, are these tasks purely theoretical or are there any users?

INEX *ad hoc* Queries

It is poignant to ask how a user might phrase their query before even being presented with their search results – this is a question that has also received much attention (see, for example, van Zwol [23]). We believe it is likely to remain a topic of debate as different users are likely to want to interact with an XML search engine in different ways. A database administrator might prefer SQL to interact with a given database whose users prefer a point and click interface. Similarly, an administrator of an XML information system might prefer XPath [4] while a user of said system might prefer a keyword interface.

INEX identifies two types of queries, those that contain Content And Structural hints (CAS queries) and those that contain Content Only (CO queries) words. The former are specified in the INEX language NEXI [21], and the latter are simple keyword lists.

Several other methods of specifying queries were tested at INEX (see Trotman [22] for details), but it is the loose semantics and the simplicity that make NEXI an appealing language for XML-IR.

Content Only then Content And Structure

Again at INEX we see a hypothetical user. This user does not know (or might not want to use) the structure of the documents in the collection. This user issues a query containing only keywords and is returned a set of elements (ordered according to the task).

Just as a user of a web search engine might refine their query by adding keywords, the hypothetical user adds not keywords but structural constraints to refine their query. Of course, they might choose to include the constraints from the beginning if they suspected it would help.

Other models

Van Zwol examined a graphical user interface for choosing structural constraints for CAS queries [23]. In his interface the user selects structures from lists and are able to specify keywords to be constrained to those structures. These graphical queries are easily converted CAS queries in the NEXI language. Baeza-Yates [1] used block-like graphical language to represent the nesting of structures. Query by example, the specification of XML fragments containing keywords was examined by Carmel *et al.* [3].

Theoretical queries

Comparative studies of the methods of asking the queries are hard to do in an environment in which there are no known users. What can be done, however, is to examine the requirements of the query language. It should optionally allow the user

to specify structure, it should optionally allow the user to specify keywords, and it should optionally allow the user to specify the granularity (or target elements) of the search result that is, the following kinds of queries:

1. Keywords only (no structural constraints or target elements specified)
2. Keywords limited to structures (no target elements specified)
3. Keywords limited to structures with target elements specified
4. Structural constraints and target elements (no keywords)
5. Structural constraints (no keywords or target elements)
6. Target elements (no keywords or structural constraints)

The last three of which (typically) require no interpretation of the user's information need and are so outside the realm of the INEX *ad hoc* task. The term *keyword* refers to single keywords as well as phrases of several words or any other string-valued search condition.

User interfaces for queries need to be limited to text-only interfaces. In the preceding section graphical query languages and list-box query interfaces were discussed. Such interfaces do not alter the kinds of queries that can be asked, only how they are asked. We believe there is room for investigation of novel query interfaces for XML, a field in its infancy.

Existing Search Models

In this section some search engines already identified as being (or hypothesized as being) XML-IR are examined. In some cases it is not possible to state with any certainty that XML is used as they are proprietary and vendors are not at liberty to disclose this information.

Version Control (Thorough)

To find an example of thorough retrieval it is necessary to find a collection of documents in which both the whole document and the document elements are atomic. That is, the document must be a loosely coupled collection of elements. [14]. Two such genre have been identified: books (discussed below); and version control.

A version control system such as RCS allows the user to manage individual source code files and also to manage whole software development projects. In some systems the user is able to search through the comments associated with a file, with a directory of files, or with a project.

In the Microsoft version control system SourceSafe, for example, the user can add comments to (label) a file, or a directory. Any comments associated with the directory also apply to any subdirectories and files of that directory. A directory label will be seen when a file's comments are examined. These comments could be (but are likely not) stored in an XML file in which the structure of the XML matches the structure of the file system.

A user searching these comments expects the version control system to identify each and every relevant comment, regardless of where in the structure that comment occurs. If a comment is given to a file, and to a directory then it is reasonable to expect it to occur multiple times in the results list. If the comment applies to a directory then the user expects to see the comment in a results list only once. In other words, the search engine should be thorough. In a version control system the context is implicit (the given archive) and so it is not needed.

News Summarization (Focused)

The awkwardness in finding an application of focused retrieval is that the context is not presented. As with thorough retrieval, if the context is implicit or otherwise already known, then context is not needed. The credibility of news stories published by BBC news, for example, can be taken for granted. The BBC news reader need not concern themselves with the authorship as the editor has already ensured the credibility of the content.

Not needing to know the greater context of a document does not mean elements are meaningful outside the context of the body text. To find an application of this aspect of focused retrieval it is necessary to look beyond the user, and at an information system as a whole. Such a system would present the user with information and not documents. News summarization is one example.

The Columbia Newsblaster summarizes news pages from multiple sources, often in HTML. To do this it must extract the text of the document from the decoration. Evans *et al.* [7] describe the elaborate mechanism that is used to do so. Focused retrieval could be used to separate the relevant text from the decoration.

Multi-document text summarizers such as MultiGen [13] are used by Newsblaster to summarize a collection of document on a single topic. Focused retrieval could be used to identify parts of documents for summarization. Such an approach might result in an increase of summarization performance as there could be an associated reduction in non-relevant material being summarized.

Book Search (Relevant in Context or Fetch & Browse)

Dopichaj [6] identifies book search as a commercial (and in-use) application of XML retrieval. Specifically he identifies Books24x7 and Safari as successful Internet sites that rely on this (or very similar) technology. His examples could not have been further from expectation; they use a thorough retrieval strategy.

Books24x7 provides an interface in which books are first ranked by topical relevance, and then within that, parts of books are ranked. Dopichaj provides an example where (within a single book) both a chapter and a section of that chapter are listed. This interface is relevant in context but results are thorough; it is the Fetch & Browse strategy. It is not clear why Fetch & Browse was dropped from INEX, but perhaps it should be reinstated.

The granularity of search on Books24x7 is not known. It is possible that only three levels of the document tree are searchable: book, chapter, and section. Some INEX

participants [19] take the view that many of the common elements (such as typographically marked elements or section headings) seen in the document tree are of no material value for retrieval and should not even be indexed. Others [12] take the view that the best elements for retrieval can be pre-determined and so only index a small subset of the elements in a document. Of the latter group, some [12] build a separate index for each element type and merge after searching each.

When examining the table-of-contents of a single book on Books24x7, chapters of that book which are relevant to the user's needs are marked as being so. By extension, if the relevance of each chapter (and section of each chapter) were also indicated then the table of contents would be a heat map of relevance across the document.

We believe that the document heat map is an important grounding for XML-IR. At each point in the document a relevance score is computed and this is displayed either alongside the document or with a table-of-contents like overview of the document. The heat map is a thorough paradigm. A given document has a certain heat and then within that each section and each subsection of that has a given heat. For evaluation purposes it is convenient to order these elements first by document then by topical relevance within document.

Extending Web Search (Best in Context)

Lehtonen [11] makes the point that the user does not need to be aware of the fact that XML is used in the underlying system for it to be XML retrieval. He provides examples of web interfaces to XML collections. It is reasonable to ask what other aspects of the web could be likened to XML retrieval, or more to the point what aspects of XML retrieval are already in use on the web. The best in context task is one example.

It is not uncommon to see a web page with many internal links. Particularly in long documents (for example academic papers) a table-of-contents with links to entry points in that document is the norm. The web rendering of papers by BioMedCentral [2] for example, includes this kind of table-of-contents which they refer to as an outline. In some cases these links are inserted automatically by software that converts XML to HTML.

It is also not uncommon for web search engines such as Google [8] to present the user with a list of results that includes one sentence query-biased summaries (or snippets) of the identified documents. This is done so that the user can see, in context, where the search terms lie within the document.

It is, however, uncommon for information systems to provide both behaviors. It is not possible for a web search engine to instruct the web browser to load a given page and locate an arbitrary snippet at the top of the window – such behavior does not exist in the browser. But, given a set of entry points into the document (each marked as an HTML anchor) it is entirely possible for the search engine to instruct the browser to load the page and to locate the last entry point before the snippet.

In this example the entry points are chosen by the author of the web page and inserted manually. But this does not make it a reasonable criticism of the analogy. There are essentially no true structures in an HTML document and the best an author

can do is to create these with anchors (cascading style sheets (CSS), div and span elements, and XHTML aside). Of course, if the authors were using XML then they would have structures available to use, but just as with the HTML example, they would manually have to choose to insert the structures (and the same is true with CSS). There seems to be no getting around the issue that entry points are manually chosen in both HTML and XML.

Existing Query Models

One is tempted to suggest that identifying any single implementation of XPath in a commercial product is sufficient evidence of its user base but it is of more value to identify mass use of the searching paradigms.

Keyword Only

Examples of keyword only interfaces abound the Internet (for example Google) and before this they abounded CD-ROMs.

Keywords Limited to Structures

Close inspection of the search interface provided by Google reveals that searches can be restricted to specific meta-data fields. This behavior is in use by search engines that utilize this search engine to search only their site. A query can, for example be restricted to the University of Otago web site by adding the search term "site:otago.ac.nz", requiring that the meta-data for the web page contain a structure called site and that that structure contain the expression otago.ac.nz.

Structured information systems such as PubMed [15] also provide similar behavior, in this case a user can restrict their search terms to occurring in a multitude of different structures including the title, the keywords, or even the abstract of a document.

Keywords Limited to Structures with Target Elements Specified

Although now defunct, BioMedNet once used an SGML [9] search engine that relied on fully normalized reference concrete syntax (it was essentially an XML search engine). The site provided a number of services including job listings, a store for biology supplies, magazines, academic journals, and abstracts from Medline.

The user interface to the search engine provided the user with the ability to search only the journal collection, only a single journal, only the jobs, only the store, or everything. Of course, the user also supplied keywords and could limit these to any structure seen in any of the SGML elements. This behavior is directly analogous to the user supplying the target elements and provided keywords optionally limited to structures (even though not implemented in this way).

Search Forms

Many search engines provide a so-called advanced interface in which the user specifies their information need by selecting from list boxes. Lehtonen [11] provides examples of such interfaces in which keywords are not needed, as well as ones in which they are.

One can easily lose sight of a goal when it lies directly under one's own nose. The bibliographic citation manager used by the authors of this contribution (EndNote) provides search functionality in which the user chooses fields from a list box and enters keywords to be limited to those fields.

XML-IR Use Cases

Formal use-cases are used to document the interaction of a user and an information system. Such use cases are often written in an environment in which the system does not yet exist, or in which the system is being modified (and consequently does not fully exist). They are design documents.

XML-IR is already in use, but this need not detract from the benefits of writing use cases. For one, a description of how it is being used could lead to insights as to how it might be used.

Cockburn [Cockburn 2001] examines effective and ineffective use cases and provides a set of reminders. Reminder 6 is to "Get the Goal Level Right". By this he is reminding us not to get bogged down in the detail of how exactly something happens, but rather to describe that it does. The exact details can sometimes be filled in with another use case while at other times it is obvious. How the user enters their query in a web search engine is less important than that they must do so.

Much of the fine detail of the interaction between the user and the information system is identical for the different XML retrieval tasks identified at INEX. The user sits at a computer, enters a query, and is presented with results. Because these details are the same each time, and essentially the same is seen in web search, it is unnecessary to include or repeat them – the level is wrong.

Cockburn [Cockburn 2001] also examines several different formats for presenting use cases. These range from the *fully dressed* format that is highly formal with numbered steps to a drawn *diagram* format to a *casual* format that is paragrammatic. It is important to choose the appropriate style for both clarity and to avoid ambiguity.

The use cases included herein are in the *casual* format. They serve not to dictate the design of a system but rather are illustrative of the user's needs, and why an XML-IR system of a particular kind is needed. As the interaction of the user and the search engine is obvious and ubiquitous, the use cases reduce to a description of the user's need and why a particular strategy satisfies their needs.

The Thorough Use Case

The user needs maintenance information to maintain some modular engineering equipment, but cannot be certain of the exact modules until in the field. Such is the

case when a modular system evolves over time but old and new pieces remain interchangeable. The product has an electronic set of service manuals that are constructed hierarchically with the details of different versions of the same module compiled into the same XML document.

The user believes it is possible to identify the component in the field from the service manuals, but to do so needs exhaustive details on each variant of a given components. The service manual is designed to extract the appropriate XML elements to form a coherent page regardless of the variant of that component.

The user consults the collection which provides a ranked list of information elements for all variants of the component.

The user browses and selects from the result list the document fragments that match the modular component in the field.

The user identifies the version of the component.

The information system constructs the service pages for the component.

Sure of a valid identification and having the service manual page the user services the part and is finished.

The Focused Use Case

The user has been asked to present a report summarizing the many different views of a given debated topic. They are, consequently, interested in reading opinions without being interested in whose opinion it is. Such might be the case if a secondary school pupil were asked to present a report on the different views of the effects of global warming.

The user believes it is necessary to obtain the information from multiple sources and from many different collections.

They consult such a meta-search engine which provides a ranked list of information elements from multiple sources.

The user browses and selects from the result list and views accompanied information.

Uninterested in the source of the information, as it is opinion they are seeking, they take notes from relevant information elements.

The user combines the notes from the multiple sources into a coherent report.

Sure of having covered the multitude of opinions, the user is finished.

The Relevant in Context (and Fetch & Browse) Use Case

The user needs factual information to solve a dispute. Such was the original reason for the collection of facts now readily available as the Guinness World Records [Guinness Book of Records, Norris and Ross McWhirter (eds) 1975].

The user believes it is possible to obtain the information from a single source in a community built collection of facts (such as Wikipedia, or Guinness).

They consult such a collection which provides a ranked list of information elements.

The user browses and selects from the result list and views accompanied information.

Knowing the facts are under dispute, the user chooses to verify the credibility of the information. The information system provides two methods for doing this:

Either the user consults a discussion list associated with the information and uses their personal experience and judgment in following the postings

Or the user consults the context of the information and uses their personal experience to judge the reliability of the source and therefore the credibility of the information.

Sure of the facts and the credibility of the source, the user is finished.

The Best in Context Use Case

The user is writing an academic paper and recalls some facts they wish to cite, but is unable to recall the source. The user is a regular and frequent user of an online academic digital library which houses many of the journals they frequently read. Such was the case for the authors of this paper during preparation.

The user is looking for a single source of the information from a reputable source (such as a journal on ScienceDirect).

They consult such a collection which provides a ranked list of information elements. As the information units are long (typically 8-30 printed pages), the result list includes query biased summaries (snippets).

The user browses and selects from the result list.

The information system presents the user with an article, pre-scrolled to the right location so that the snippet is on-screen in context, for the user to read.

The user discovers that their understanding of the details is not sufficiently detailed and chooses to read more of the context.

As the information unit is on-screen, the user scrolls and reads at their leisure to ensure they sufficiently understand the details.

Sure of their facts, and with a citation to them, the user is finished.

Conclusions

There are currently four *ad hoc* tasks at INEX: thorough, focused, relevant in context, and best in context. For each task it is reasonable to ask if it is a purely academic task being studied for academic reasons, or if such systems exist and therefore might be improved by research in XML-IR.

For thorough retrieval, internet book search as well as version control are identified as existing uses of the technology. For focused, news summarization. The relevant in context task is already in use in internet book search, and an application of best in context is suggested. Use cases for each of the four tasks are presented.

Accepting that it is reasonable to rank the tasks on credibility, we suggest the relevant in context task is most credible (as it has been show to exist), followed by best in context, and then focused, then thorough.

XML-IR users certainly exist. We believe the use cases are reasonable, and we hope that future research in XML-IR will consider the existing users as well as the hypothetical users for whom some of the INEX tasks were targeted.

References

- [1] Baeza-Yates, R., Navarro, G., & Vegas, J. (1998). A model and a visual query language for structured text. In *Proceedings of the String Processing and Information Retrieval: A South American Symposium*, (pp. 7-13).
- [2] BioMedCentral. (2006). Biomedcentral. Available: <http://biomedcentral.com> [2006, 21 November].
- [3] Carmel, D., Maarek, Y. S., Mandelbrod, M., Mass, Y., & Soffer, A. (2003). Searching XML documents via XML fragments. In *Proceedings of the 26th ACM SIGIR Conference on Information Retrieval*, (pp. 151-158).
- [4] 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>.
- [5] Clarke, C., Kamps, J., & Lalmas, M. (2006, to appear). INEX 2006 retrieval task and result submission specification. In *Proceedings of the INEX 2006 Workshop*.
- [6] Dopichaj, P. (2006). Element retrieval in digital libraries: Reality check. In *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology*, (pp. 1-4).
- [7] Evans, D., Klavans, J. L., & McKeown, K. R. (2004). Columbia newsblaster: Multilingual news summarization on the web. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL-2004)*.
- [8] Google. (2006). Google. Available: <http://google.com> [2006, 21 November].
- [9] ISO8879:1986. (1986). *Information processing - text and office systems - standard generalised markup language (SGML)*.
- [10] Kazai, G., & Ashoori, E. (2006). What does shakespeare have to do with INEX? In *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology*, (pp. 20-27).
- [11] Lehtonen, M. (2006). Designing user studies for XML retrieval. In *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology*, (pp. 28-34).
- [12] Mass, Y., & Mandelbrod, M. (2004). Component ranking and automatic query refinement for XML retrieval. In *Proceedings of the INEX 2004 Workshop*, (pp. 73-84).
- [13] McKeown, K. R., Barzilay, R., Evans, D., Hatzivassiloglou, V., Kan, M.-Y., Schiffman, B., & Teufel, S. (2001). Columbia multi-document summarization: Approach and evaluation. In *Proceedings of the Document Understanding Workshop (DUC 2001)*.
- [14] O'Keefe, R. A. (2004). If INEX is the answer, what is the question? In *Proceedings of the INEX 2004 Workshop*, (pp. 54-59).

- [15] PubMed. (2006). Pubmed. Available: <http://ncbi.nlm.nih.gov> [2006, 21 November].
- [16] Tombros, A., Larsen, B., & Malik, S. (2004). The interactive track at INEX 2004. In *Proceedings of the INEX 2004 Workshop*, (pp. 410-423).
- [17] Trotman, A. (2005). Wanted: Element retrieval users. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, Second Edition*, (pp. 63-69).
- [18] Trotman, A., & Geva, S. (2006). Passage retrieval and other XML-retrieval tasks. In *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology*, (pp. 43-50).
- [19] Trotman, A., & O'Keefe, R. A. (2003). Identifying and ranking relevant document elements. In *Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX)*.
- [20] Trotman, A., & Pharo, N. (2006). User case studies track. INEX. Available: <http://inex.is.informatik.uni-duisburg.de/2006/usercase.html> [2006, 13 November].
- [21] Trotman, A., & Sigurbjörnsson, B. (2004). Narrowed Extended XPath I (NEXI). In *Proceedings of the INEX 2004 Workshop*, (pp. 16-40).
- [22] Trotman, A., & Sigurbjörnsson, B. (2004). NEXI, now and next. In *Proceedings of the INEX 2004 Workshop*, (pp. 41-53).
- [23] van Zwol, R., Baas, J., van Oostendorp, H., & Wiering, F. (2005). Query formulation for XML retrieval with bricks. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, Second Edition*, (pp. 80-88).

What XML-IR Users Want

Alan Woodley¹, Shlomo Geva¹, Sylvia L. Edwards²

¹School of Software Engineering and Data Communication, ²School of Information Systems
Faculty of Information Technology, Queensland University of Technology
GPO Box 2434, Brisbane, Queensland, Australia
{ap.woodley@student.qut.edu, s.geva@qut.edu.au}

Abstract. It is assumed that by focusing on retrieval at a granularity lower than documents that XML-IR systems will better satisfy users' information need than traditional IR systems. Participates in INEX's Ad-hoc track develop XML-IR systems based upon this assumption, using an evaluation methodology in the tradition of Cranfield. However, since the inception of INEX, debate has raged on how applicable some of the Ad-hoc tasks are to real user tasks. The purpose of the User-case Studies track from to explore the application of XML-IR systems from a user perspective. This paper outlines QUT's involvement in this task. For our involvement we conducted a user experiment using an XML-IR system (GPX) and three interfaces: a standard keyword interface, a natural language interface (NLPX) and a query-by-template interface (Bricks). Following the experiment we interviewed the users about their experience and asked them - in comparison with a traditional XML-IR system - what type of tasks would they use an XML-IR for, what extra information they would need and how would they want to see XML-IR results presented. It is hoped that the outcomes of this study will bring us closer to understanding what users want from XML-IR systems.

On the unsupervised classification of text-centric XML document collections

Antoine Doucet¹ and Miro Lehtonen²

¹ IRISA-INRIA
Campus de Beaulieu
F-35042 Rennes Cedex
France

`firstname.lastname@irisa.fr`

² Department of Computer Science
P. O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland
`firstname.lastname@cs.helsinki.fi`

Abstract. In this paper, we present the results of the clustering experiments related to the participation of the University of Helsinki in the Document mining track of the Initiative for the Evaluation of XML Retrieval 2006 (INEX 2006). We start with a review of related work, after which we experiment with a number of techniques that were developed at a time when no performance evaluation framework was available. The techniques are built on top of the vector space model, and enhanced with different types of features, either separately (textual, structural), or in a 2-step approach (first structural, then textual). We present the corresponding results in the context of the INEX 2006 document mining track. We then extend our contribution to the integration of a measure of “textitude” of a structured document in the document description process. Because we require no DTD, our techniques are particularly suited for experiments with several different document collections, such as the IEEE journal collection and the Wikipedia collection, used in the mining track 2006.

The evaluation of clustering consists of comparing automatic unsupervised classification instances to a given “gold-standard”. Finding such an ideal classification is very difficult, as many ways to split a document collection may be equally valid and arguable. However, we believe that the gold-standards used in the evaluation of the INEX mining track are heavily oriented towards the textual content of the document, and far less towards their structural content. Therefore, it came as no surprise that our best results were obtained when we exclusively used textual features. The paper discusses this issue and develops on why the results should be analyzed very carefully.

1 Introduction

No DTD for the Wikipedia documents. Perfectly fine since our methods do not require one. Looking at the totality of the INEX 2006 clustering submissions,

this seems to be a specificity of our system, because most of the INEX clustering runs were submitted versus the IEEE collection (at this point, however, and as long as we do not know more about the other systems, we cannot be certain that this explanation is correct).

2 Related work

3 Run description

Our approach relies on the vector space model. Every document of the collection is represented by a set of values corresponding to different features of that document...

We based our experiments on the clustering technique *k – means*.

3.1 Text features only

3.2 Tag features only

3.3 Text and tag features

3.4 The 2-step approach: Tag features first, Text features next Isim - 2003

Previous experiments suggested that a 2-step approach permits to obtain better results, by putting aside structural outliers before running the textual (semantic) classification [1].

3.5 Integrating a new structural indicator

The T/E measure The T/E measure is a structural indicator of the proportion of “mixed content” in an XML fragment. In previous research, it has given us the Full-Text Likelihood of an XML element, based on which, the element could be excluded from a full-text index [2]. Although the values of the T/E measure are in the continuous range from 0 to ∞ , the interpretation has come with a projection into a binary value space, where values greater than 1.0 provide evidence of full-text content. When treated as a feature for clustering XML documents, the projection is unnecessary. Therefore, the whole value space of the T/E measure is available.

Integrating the T/E measure into the vector space model

4 Evaluation

4.1 Collection description

IEEE
Wikipedia

4.2 Definitions.

Micro average, macro average, etc.

4.3 Table summary with micro and macro avgs

Table 1. INEX - IEEE Collection

Features	Micro F1	Macro F1
Text	.348789	.290407
Tags	.132453	.081541
Text+Tags	.253369	.203533
Tags→Text, 0.8	.270350	.222365
Text + TE	.348828	.290379

Table 2. INEX - Wikipedia Collection

Features	Micro F1	Macro F1
Text	.444455	.210621
Tags	.221829	.072834
Text+Tags	.372376	.128239
Tags→Text, 0.8	.406129	.155034
Tags→Text, 0.9	.413439	.159473
Text + TE	.427438	.183567

Other clustering runs (from other INEX mining participants):

- INEX:
 - Team 3, 5 runs (results = one of them beats our best - the others are close to the results of our tags only submission - our worst)
 - Team 4, 3 runs (results = worse than all of ours)
- Wikipedia:
 - Team 4, 1 run (results = worse than all of ours)

All the submissions, ours and those of other participants, returned the same number of categories as in the ideal classifications: 18 for the IEEE collection and 60 for the Wikipedia collection.

4.4 Result analysis

INEX-IEEE Several issues with the ideal classification, the one we should aim at. Its categories, disjoint, are the journals a document was published in.

Two issues:

Calls for papers, tables of contents, etc., should they really belong to the same categories as the articles they contain?

More generally, we do not think that the themes of the papers of the 18 IEEE journals are mutually exclusive. This needs to be reflected in the evaluation.

The main problem at the moment is that, to obtain better results, we need to contradict a number of apparently reasonable assumptions.

Wikipedia.

What's gold? The main problem is, what do people (users) want?

Is there a conflict between the goals of XML mining and the “correct” clusters, central to the evaluation of XML clustering?

5 Conclusion - Future work

References

1. Doucet, A., Ahonen-Myka, H.: Naive clustering of a large xml document collection. In: Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), Schloss Dagstuhl, Germany (2002) 81–87
2. Lehtonen, M.: Preparing Heterogeneous XML for Full-Text Search. *ACM Transactions on Information Systems* **24** (2006) 1–21

XML Document Mining using Contextual Self-Organizing Maps for Structures

M. Kc¹, M. Hagenbuchner¹, A.C. Tsoi², F. Scarselli⁴, M. Gori⁴, and A. Sperduti³

¹ University of Wollongong, Wollongong, Australia

² Monash University, Melbourne, Australia

³ University of Padova, Padova, Italy

⁴ University of Siena, Siena, Italy

Abstract. XML is becoming increasingly popular as a language for representing many types of electronic documents. The consequence of the strict structural document description via XML is that a relatively new task on mining documents based on structural and/or content information has emerged. In this paper we investigate (1) the suitability of new unsupervised machine learning methods for the clustering task of XML documents, and (2) the importance of contextual information for the same task. It is found that the proposed machine learning methods are suitable for the clustering of structured data by winning an international competition on clustering of XML data. It is also found that XML structure is a causal decomposition of information and thus does not require contextual processing.

1 Introduction

The eXtensible Meta Language (XML) is a tool for describing many types of electronic documents. XML is related to HTML (hypertext marked up language) but is much more flexible and strict in its definition. As such, XML is increasingly utilized to represent a wide range of electronic documents ranging from e-books, Web pages, to multi-media contents. This is due to the fact that XML provides a flexible document format which allows cross-platform compatibility. Thus, documents created as a result of a UNIX application can be viewed or edited on by any other systems with an XML capable application. Data mining on such types of documents become increasingly important as a consequence. The striking feature that makes data mining on XML documents so interesting is that XML provides a strict structural document description and hence, XML document mining can be predominantly a structure mining task.

XML provides meta-tags which encapsulate content. These meta-tags can be nested and define the property of the content. As a simple example, the following line of XML code: `<A>HelloWorld` gives a document which has the string “Hello World” as its content where the word “World” shares the property `<A>` of the word “Hello” but also is being assigned the property ``. For example, “Hello” may be in italic font, while “World” may be in italic and bold font. It

is not possible to have overlapping XML tags such as `<A>`. The consequence is that XML provides a tree-like decomposition of a document's content. In other words, any XML document can be represented by a tree-like graph structure.

Recent developments in machine learning produced neural network models which are capable of encoding graph structured information. This paper is concerned with the specific task of clustering XML documents. To the best of our knowledge, there is only one known neural network method that is capable of clustering structured information in an unsupervised fashion. The method is known as the Self-Organizing Map for Structured Data (SOM-SD) and its extension: Contextual SOM-SD (CSOM-SD) which can deal with contextual information on graph structured data. These methods are recent advances of the popular Self-Organizing Map (SOM) model introduced by T.Kohonen some 20 years ago.

At the INEX 2005 workshop it was shown for the first time that a SOM-SD based model can produce exceptional performances on XML classification tasks by winning the 2005 international competition by having the best overall performance compared with the performance of other approaches. This paper applies the SOM-SD and CSOM-SD approaches to a new and relatively large XML clustering task: to investigate the suitability of such machine learning methods, a task that has never before been executed in this manner. This paper will also answer the question of whether the contextual processing of information can lead to improvements in this XML mining task. This question can be answered by comparing the SOM-SD which processes information in a strict causal manner with results from applying the CSOM-SD which processes the same data in a contextual fashion.

The structure of this paper is as follows: Section 2 provides a brief introduction to unsupervised machine learning approach and describes the basic idea of SOM. A detailed description of the SOM-SD method is provided in Section 3, and its CSOM-SD extension is given in Section 4. A specific description of the learning task, the approach, and experimental results are given in Section 5. Some conclusions are drawn in Section 6.

2 Machine learning and Self-Organizing Maps

The term *machine learning* (ML) refers to computer models which are capable of learning from examples. Artificial Neural Networks (ANNs) for example is one branch of machine learning where the task is to develop a model which is inspired by the ability of a (human) brain to learn using the connection strengths among the biological neurons to represent the information contained in the inputs. Such an approach has the obvious advantage of not requiring a system to hard-code information and thus is much more flexible in its application. However, ML is generally known to be an inexact science in that the answer of a ML method can only be *approximately* correct. In practice, this approximation is extensively exploited as it allows to obtain a system that remains stable in a noisy (learning

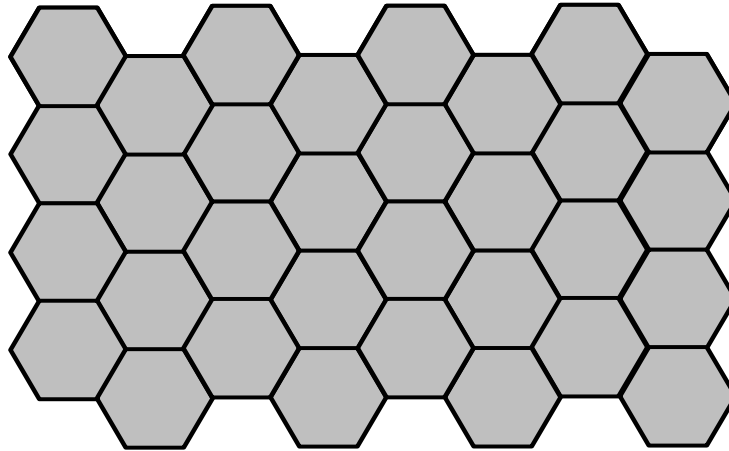


Fig. 1. The architecture of a simple Self-Organizing Map.

and application) environment. For example, ML methods are popularly applied in human-machine interfaces such as speech recognition systems. No two voice signals are ever produced exactly identical due to the analogue nature of sound. Thus, a voice recognition system needs to be able to recognise ever differing signals in a possibly noisy environment. A ML method is trained on a set of sample signals together with a desired output signal. An iterative training approach is commonly applied with the aim to adjust internal system parameters such that for a given input signal the desired output signal is produced (as closely as possible). Such learning methods are generally asymptotically convergent which means that the desired output can never be reached accurately in finite time. However, most ML methods, especially ANN methods, are proven to be able to approximate any input-output relationship to any arbitrary precision provided some not too restrictive but generic conditions are satisfied, which makes ML methods work very well in practice.

This paper is concerned with one particular branch of ANNs known as *unsupervised neural networks*. Unsupervised neural network methods are trained on input data for which target information is not available. The general application of such methods is to cluster data such that data that share certain properties are grouped together. The most famous and most widely used of the unsupervised neural network methods is the Self-Organizing Map and its many variants originally developed by [6].

The basic idea of SOM is simple. The SOM defines a mapping from high n -dimensional input data space onto a regular lower q -dimensional array (often two-dimensional array) called a *display map*. The intersection of the two dimensional grid in the display map is represented by an entity called a *neuron*. Every neuron i of the display map is associated with an n -dimensional codebook vector $\mathbf{m}_i^T = (m_{i1}, \dots, m_{in})^T$, where the superscript T denotes the transpose of a

vector, and $m_{ij} \in \mathcal{R}$. The neurons on the map are connected to adjacent neurons by a neighborhood relation, which defines the topology, or the structure, of the map. The most common topologies in use are rectangular and hexagonal [6]. Adjacent neurons belong to the neighborhood N_i of the neuron i . Neurons belonging to N_i are updated according to a neighborhood function $f(\cdot)$ such as a *Gaussian-bell* or a *Mexican-hat* function. Typically, the topology and the number of neurons remain fixed from the beginning of the training process. The number of neurons determines the granularity of the mapping, which has an effect on the accuracy and generalization capability of the SOM [6]. Figure 1 shows the architecture of a two-dimensional SOM of size $8 \times 4 = 32$. Each hexagon represents a neuron, and since each neuron has six neighbors, the topology of this map is hexagonal. Not shown in Figure 1 is that each of these neurons is associated with an n -dimensional codebook vector.

The vector \mathbf{m}_i are updated by a training process. During the training phase, the SOM forms an elastic cover that is shaped by input data. The training algorithm controls the cover so that it strives to approximate the density of the underlying data. The reference vectors in the codebook drift to the areas where the density of the input data is high. Eventually, only few codebook vectors lie in areas where the input data is sparse. The result is that the data is clustered. The training algorithm for the weights associated with each neuron in the q -dimensional lattice ($q = 2$ in our case) can be trained using a three step process as follows:

Step 1 Competitive step. One training sample $\mathbf{u} \in \mathcal{R}^n$ is randomly drawn from the input data set and its similarity to the codebook vectors is computed:

$$r = \arg \min_i \|\mathbf{u} - \mathbf{m}_i\| \quad (1)$$

where $\|\cdot\|$ is the Euclidean distance.

Step 2 Cooperative step. After the best matching codebook \mathbf{m}_r has been found, all codebook vectors are updated. \mathbf{m}_r itself as well as its topological neighbours are moved closer to the input vector \mathbf{u} in the input space. The magnitude of this adjustment is governed by the learning rate α and by a neighborhood function $f(\Delta_{ir})$, where Δ_{ir} is the topological distance between \mathbf{m}_r and \mathbf{m}_i . As the learning proceeds and new input vectors are given to the map, the learning rate gradually decreases to zero according to a specified function. Along with the learning rate, the neighborhood radius decreases as well ⁵. The updating algorithm is given by:

$$\Delta \mathbf{m}_i = \alpha(t) f(\Delta_{ir})(\mathbf{m}_i - \mathbf{u}) \quad (2)$$

where α is a learning coefficient, $f(\cdot)$ is a neighborhood function which controls the amount the weights of the neighbouring neurons is updated. The

⁵ Generally, the neighborhood radius in SOMs never decreases to zero. Otherwise, if the neighborhood size becomes zero, the algorithm has no longer topological ordering properties ([6], page 111).

neighborhood function $f(\cdot)$ can take the form of a Gaussian function:

$$f(\Delta_{i_r}) = \exp\left(-\frac{\|\mathbf{l}_i - \mathbf{l}_r\|^2}{2\sigma(t)^2}\right) \quad (3)$$

where σ is the neighborhood radius, and \mathbf{l}_r is the location of the winning neuron, and \mathbf{l}_i is the location of the i -th neuron in the lattice. Other neighborhood functions are possible.

Steps 1 and 2 together constitute a single training step and they are repeated until the training process ends. The number of training steps must be fixed prior to training the SOM because the rate of convergence in the neighborhood function and the learning rate are calculated based on this information.

The SOM, and in fact all conventional ML methods, require data to be available in vectorial form. This includes recurrent neural network methods which can process continuous signals by processing shifting fixed sized sub-sections of a continuous signal one at the time. A recent extension [2] produced Self-Organizing Maps which can process graph structured data without requiring the pre-processing of such data. This is being addressed in the following section.

3 Self-Organizing Maps for Structures

Work on SOM methods capable of mapping structures was inspired by developments with supervised neural networks for graphs [1]. The basic idea behind these approaches is to process individual nodes of a graph rather than the graph as a whole, and to provide a mechanism to incorporate structural dependencies between the nodes. This is achieved by adding a set of *states* which stores information about the activations of the network when processing a given node, and to use the states associated with neighboring nodes as an additional inputs when processing the current node. The recursive application of this approach ensures that information about the encoding of any node in a graph is passed on to any other node in the graph as long as there is a path connecting these nodes. In other words, the idea allows to encode a graph as a whole by processing individual nodes one at a time.

When applied to SOM, the idea allows the processing of labelled directed acyclic graphs (labelled tree structures) [2] by producing the network input \mathbf{x}_v for every node v in a graph through the concatenation of the label \mathbf{u}_v and states $\mathbf{y}_{ch[v]}$ so that $\mathbf{x}_v = [\mathbf{u}_v, \mathbf{y}_{ch[v]}]$. The $\mathbf{y}_{ch[v]}$ are simply the mappings (the coordinates) of the children of v .

Training a SOM-SD is performed in a similar manner to the classical approach [6]. The difference is that for computing the similarity in Equation 1, the measure needs to be weighed so as to account for the hybrid data in the vector \mathbf{x}_v . Also, some components (the states) of the input vector need to be updated at every training step. The resulting training algorithm is as follows:

Step 1 A node v is chosen from the data set. When choosing a node special care has to be taken that the children of that node have already been processed.

Hence, at the beginning the terminal (sink) nodes of a graph are processed first, the root (source) node is considered last. Then, vector \mathbf{x}_v is presented to the network. The winning neuron r is obtained by finding the most similar codebook entry \mathbf{m}_r as follows:

$$r = \arg \min_i \|(\mathbf{x}_v - \mathbf{m}_i)\mathbf{\Lambda}\| \quad (4)$$

where $\mathbf{\Lambda}$ is a $n \times n$ dimensional diagonal matrix. Its diagonal elements $\lambda_{11} \cdots \lambda_{pp}$ are set to μ_1 , all remaining diagonal elements are set to μ_2 . The constant μ_1 influences the contribution of the data label component to the Euclidean distance, and μ_2 controls the influence of the states (the node's children coordinates) to the Euclidean distance.

Step 2 After the best matching neuron has been found, the winning codebook vector and its neighbours are updated as in Equation 2.

Step 3 The coordinates of the winning neuron are passed on to the parent node which in turn updates its vector \mathbf{y}_{ch} accordingly. This step neglects the potential changes in the state of the descendants due to the weight change in step 2. This approximation does not appear to cause any problems in practice [2].

Cycles of Steps 1 to 3 are executed repeatedly until a given number of training iterations is performed, or when the mapping precision has reached a given threshold.

The optimal choice of the values μ_1 and μ_2 depends on the dimension of the data label \mathbf{l} , the magnitude of its elements, the dimension of the coordinate vector \mathbf{c} , and the magnitude of its elements. The Euclidean distance in Equation 4 is computed as follows:

$$d = \sqrt{\mu_1 \sum_{i=1}^p (u_i - m_i)^2 + \mu_2 \sum_{j=1}^{2o} (y_{ch_j} - m_{n+j})^2} \quad (5)$$

where o is the number of children whose states are fed into the node v . This value changes with each node according to the connectivity of the node v . The factor 2 is due to the fact that each child node will have two coordinates. Hence, it becomes clear that the sole purpose of μ_1 and μ_2 is to balance the influence of the two terms in the behaviour of the learning algorithm. Ideally, the influence of the data label and the coordinate vector on the final result is equal. This can be computed by statistically analysing the training dataset [2].

It can be observed that the SOM-SD processes data in a strict causal manner from the terminal nodes towards the root node. Processing in the reverse direction, i.e. from the root node towards the terminal nodes is also possible but rarely applied. It is not possible to process nodes in both directions or in a random order since otherwise in Step 2 it is possible that not all states of all of a particular selected node's neighbors are available. An extension which circumvents this problem is described in the following section.

4 Contextual Self-Organizing Maps for Structures

The SOM-SD processes nodes in an inverse topological order (i.e. from the terminal nodes towards the root node) which is required to guarantee that the states of all dependencies are available at any time during network training. This limits the ability of the SOM-SD to discriminate certain sub-structures. For example, the mapping of a graph with a single node **A** would be exactly the same for a node **A** that occurs as a leaf in another graph with many nodes. This is because no information about parent nodes can be included when computing a mapping and hence, any identical sub-structure would be mapped onto the same location independent of the contextual arrangement in which such a sub-structure occurs.

In order to capture differing contextual arrangement of nodes in a graph it is necessary to allow the inclusion of information about the mappings of its parent nodes as well as the mapping of child nodes at any time during training. A solution to this problem would bring a far reaching advantage: it would become possible to process undirected cyclic graphs, a much more general class of graphs than tree structures.

A first solution was proposed in [3]. In [3] it is observed that while the mapping of parent nodes is not available at a time instant t that it is available by retrieving the mappings at time $t - 1$. Given the asymptotic nature of the training algorithm it can be assumed that mappings do not change significantly between two iterations⁶, and hence, the utilization of mappings from a previous iteration in order to fill the states that are not available at a current time should be a valid approximation.

An advancement of this idea removed two obstacles: (1) the initialization problem (i.e. at time $t = 0$ there is no mapping from time $t - 1$ available), and, (2) the need to memorize the mappings of the time instant $t - 1$ by recursively computing a stable point in the state space [4]. The method proposed in [4] allows to process nodes in a random order thus removing the necessity of having an acyclic directed graph, and removes the need to sort the nodes in a particular order before processing can commence. The proposed mechanism to compute the states of all parents and children of a randomly selected node is by computing a stable point in the mapping of nodes as follows:

A Select a node from the dataset. Generate a k dimensional vector. Initialize the first p elements with the data label that is attached to this node. Initialize all the states of offsprings $\mathbf{y}_{ch[v]}$, and all the states of parents $\mathbf{y}_{pa[v]}$ with zero since these states are as yet unknown. Initialize all remaining elements with $(-1, -1)$ (corresponding to missing parents or missing children.)

B Find the best matching codebook entry.

Steps **A** and **B** are executed for each node in the training set. This computes every node's state as far as it is possible at this stage. A stable point is computed by recursively executing the following steps: (a) select a node from the dataset; (b) generate a k -dimensional vector with the first p elements initialized with the

⁶ This is particularly the case during the final stages of the learning procedure.

data label that is attached to this node, initialize $\mathbf{y}_{ch[v]}$ with the states of this node’s offsprings (which is available from the previous iteration), the $\mathbf{y}_{pa[v]}$ with the states of this node’s parents (which are available from the previous iteration), and all other elements with $(-1, -1)$ as before. Once every node in the training set has been considered, this is called “one iteration”. The algorithm iterates until the number of changes of node states during an iteration do not decrease further. With this mechanism in place, it becomes possible to train a CSOM-SD as follows:

Step 1 Compute the stable fixed point.

Step 2 Train the SOM-SD on every node in the training set as usual ensuring that the vector components $\mathbf{y}_{ch[v]}$ and $\mathbf{y}_{pa[v]}$ are updated by using the states from the previous iteration, and ensuring that the components \mathbf{l} , $\mathbf{y}_{ch[v]}$, and $\mathbf{y}_{pa[v]}$ are weighted so as to balance their influence on the Euclidean distance measure similar to that described in Section 3.

Iterate **Step 2** at least N_d times, where N_d is the maximum number of training iterations.

Note that this approach, given a sufficiently large map, ensures that identical substructures which are part of different graph structures are mapped to different locations. It can be assumed that a map properly trained will map vertices to the same or nearby location only if the underlying graph structure is similar. Note also that the complexity of the training algorithm increases linearly with the size of the training set, the size of the network, and the maximum number of iterations. Hence, the approach provides a mechanism which may be capable of finding similar graphs (inexact graph matching) in linear time.

Note also that the stable fixed point is computed only once. One may consider the possibility of computing the stable fixed point every time the network parameters are adjusted (i.e. after processing a node). However, this would increase the computational complexity to a quadratic one, and may lead to a moving target problem.

It was shown in [4] that the CSOM-SD algorithm is stable in practical applications and it was found in a number of experiments that the CSOM-SD is consistently less sensitive to initial network conditions and training parameters when compared to SOM-SD. There is as yet no explanation of such behaviours.

5 Experiments

The clustering task for the INEX dataset consists of two components: clustering using only the structural information, and clustering using both the structure and textual content. Processes addressed in this Section include dataset analysis, data pre-processing, training using structural information only, training using both structural and textual information, and comparisons of results with other approaches.

5.1 Data Analysis

The INEX dataset includes XML formatted documents, each from one of 18 different journals, covering both transactional and non-transactional journals and across various topics in computer science. However, there are up to five journals that belong to the same structural (transactional or non-transactional) and semantic (topics) grouping, therefore distinct differences cannot be expected from documents of several journals. Furthermore, the journals are unbalanced in the number of documents they contain in the training dataset, therefore, this learning task is high in complexity, yet contains features that are commonly found in real-world problems.

The documents used in the training process is the training portion of the dataset, which consists of 6,053 documents, and the number of XML tags in each document ranges widely, from 9 to 7,024, with a total of 3,966,123 tags. To represent the structure of a document, a tree could be used where each node in the tree represents the occurrence and location of XML tags. This will result in large trees where the maximum depth is 19 and the maximum out-degree is 1,023. Another observation of the INEX dataset is that there are a total of 165 unique tags, and some tags occur with a high frequency in a number of documents, but not all tags occur in all documents.

The documents used in the testing process is the testing portion of the dataset, which consists of 6,054 documents, and the proportion of documents in each journal is comparable to the training data, to ensure that the rules learned from training can be applied to the test data and that a similar level of performance can be expected.

5.2 Data Pre-processing

As described in the data analysis, the training data has a total of almost 4 million nodes and a maximum out-degree of 1,023. The XML documents are represented by trees. Nodes denote XML tags and the arc between a parent and a child represents the encapsulation of a tag in another tag. Given a learning method which processes each node individually, and requires the inclusion of the states of all offsprings, it is found that such data could result in very long training durations. While the learning method does not necessarily require pre-processing steps, some pre-processing is applied in order to improve the turn around time for the experiments. Four pre-processing approaches were considered and are compared.

- 1. Consolidating repetitive sub-structures:** This was suggested by an approach taken at the 2005 INEX challenge [5, 7], which consolidates repetitive sub-structures. However, due to the large variation of structures and tag positions in this year's INEX documents, this approach did not significantly decrease the expected training times. Another drawback of this approach is that the repeating sub-structures may be a significant feature, and compressing that feature could impact the clustering performance.

2. **Constructing tag-based connectivity graph:** To construct a connectivity graph based on the unique tags in each document instead of using a tree to represent the document structure. In the connectivity graph, each unique tag is represented by a node, so multiple occurrences of the same tags are merged. Also, care was taken to unfold the cyclic portions of the connectivity graph using the depth-first links visited method.
3. **Extracting header segment of documents:** Segment documents by using only the structure of a chosen segment. The document is first segmented by the nodes in the first level of the structure tree. This identified the sub-structures: FNO, DOI, FM, BDY and BM, where FNO and DOI do not contain any XML tags, and BM does not occur in all documents, so that leaves us with FM and BDY. The substructure where the maximum out-degree occurred was in the BDY segment, so we decided to take the FM segment, which is small enough to train without any processing of its structure.
4. **Developing a basic framework:** This approach considered the use of a simplistic framework. While the training duration using the structure of document headers may be reasonable for structure only clustering, it will be far too long for clustering based on structure and textual information, where the challenge of effectively encoding high dimensional textual information into the structure needs to be addressed. As a result, a framework is developed where only the key tags of documents are included to minimize and control the number of nodes per document. The key tags refer to significant structural elements of each document such as *Title*, *Abstract*, *Body*, and *Conclusions*. Thus, the result is a reduction of the structural representation of each document to at most four nodes. This is explained in some more detail in Section 5.4.

5.3 Training using Structural Information

For the clustering using only structural information, the documents went through the header structure extraction process (the third approach in Section 5.2). This approach has the least structural modification, therefore is expected to closely resemble the original document structure.

The SOM provides a discrete mapping space. The size of the map needs to be determined prior to a training session. In order to obtain an indication of suitable network sizes, it is recommended to consider the analysis of the training set. It is found that the set which will be used for the experiments consists of 108,523 nodes. Each node is the root of a sub-structure. Thus, there are 108,523 sub-structures in the dataset, 2,275 of these sub-structures are unique, while 48,620 of the nodes are found in a unique contextual arrangement within the graphs. Since, each pattern is represented by a codebook, the number of codebooks should be large enough to represent every different feature of any pattern. In fact, usually the number of codebooks is selected approximately equal to the number of different sub-structures in SOMSD and unique graph-node pairs in CSOM-SD. In other words, the statistical analysis of the dataset suggests that

Table 1. The training parameters used for the structure only learning task.

Clustering method	Map size	Learning rate	Iteration	Radius	μ_1	μ_2	μ_3	Training time
SOM-SD	8800	0.7	150	15	0.05	0.95	0	16 hours
CSOM-SD	8800	0.7	100	15	0.005	0.095	0.9	16 hours

a SOMSD would require at least 2,275 codebook vectors in order to provide the means to differentiate the mappings of the unique features in the input dataset. It furthermore tells us that a CSOM-SD would require at least 48,620 codebooks. Hence, it is seen that the CSOM-SD should be able to differentiate the mappings much more effectively than when compared to the SOM-SD.

For first experiments we utilized maps which consisted of 8,800 codebook vectors. Training was conducted using both SOM-SD and CSOM-SD. The training parameters used are as shown in Table 1.

The initial training used a map size based on the number of unique substructures in the document structure tree. However, the map size that provides a good performance within allowable time is slightly larger than their initial map size which in this case is 8,800.

The use of appropriate parameters for training is essential for achieving good clustering performance. Usually, the best training parameters are identified through a trial-and-error process. It was observed that an initial learning rate of 0.7 and a large number of iterations is best for most training runs. However, the balance between high number of iterations and long training duration is a challenging task.

Another observation is that although previous clustering experience indicated that best performance can be obtained when the radius is a half of the length of the training map’s side, which was not the case for this clustering task. For the structures used in clustering the INEX dataset, the radius that delivers best clustering performance remains small regardless of the map dimension. This could be due to the fact that nearby clusters are somewhat independent of each other and should be handled differently; therefore updating an extended radius causes the performance to decline.

The most influential parameter is perhaps the use of an appropriate weight value for the node label (μ_1), as it also determines the weight of the children nodes (μ_2) in the structure. The selection of appropriate weight for the node label is dependent on the importance of the node label in the structure used.

The parameters used for SOM-SD were taken as the starting point for CSOM-SD training and the map size is approximately the same as the number of unique nodes. Then, adjustments were made based on the observations of SOM-SD and CSOM-SD in the work from the previous challenge, which indicated the following:

- The number of training iterations for CSOM-SD does not need to be as high as that used for SOM-SD.
- The map size for CSOM-SD should be much larger than the map size provided for SOM-SD.

The difference between the number of unique substructures and unique nodes indicated that for this particular structure, a CSOM-SD experiment that will require a map size that is about 15 times larger than that used for the SOM-SD. However, due to the time constraint, the CSOM-SD experiments conducted used only a map size as large as 8,800.

Similar to SOM-SD, the appropriate weight values are important for good CSOM-SD clustering performance. The difference in performance with various weight values on parent nodes (μ_3) is quite significant. In an example where all parameters remain constant and only the weights were adjusted, an increase of weight on parent node from 0.5 to 0.9 increased the micro F1 clustering result by 2%.

Performance measures will be given by F1. Macro and micro statistics are given. Macro averaging is calculated for each cluster and then averaged while micro averaging is computed over all neurons and then averaged. $F1$ is defined as $\frac{2PR}{P+R}$ where P is the precision and R is the recall. $F1$ can be computed if target information is available for the training and test dataset by computing R and P as follows:

Recall: Assuming that for each XML document d_j the target information $y_j \in \{t_1, \dots, t_q\}$ is given. Since each XML document is represented by a tree, and since both the SOM-SD and CSOM-SD consolidate the representation of a tree in the root node, we will focus our attention just on the root of the tree. With r_j we will refer to the input vector for SOM-SD or CSOM-SD representing the root of the XML document d_j . Then the index is computed as follows: the mapping of every node in the dataset is computed; then for every neuron i the set $win(i)$ of root nodes for which it was a winner is computed. Let $win_t(i) = \{r_j | r_j \in win(i) \text{ and } y_j = t\}$, the value $R_i = \max_t \frac{|win_t(i)|}{|win(i)|}$ is computed for neurons with $|win(i)| > 0$ and we obtain $R = \frac{1}{W} \sum_{i, |win(i)| > 0} R_i$, where $W = \sum_{i, |win(i)| > 0} 1$ is the total number of neurons which were activated at least once by a root node.

Precision: $P = \sum_{i, |win(i)| > 0} \frac{R_i \cdot |win(i)|}{W}$.

5.4 Training using Structure and Textual Information

For the task of clustering using structure and textual information, the input data is very different to the data for structure only clustering. The documents went through a simple framework filtering process (approach 4 in Section 5.2). This approach uses a maximum of 4 nodes to represent the document structure, so that a high dimension of textual information can be added without requiring excessive training time.

Table 2. The training parameters used for the clustering task of both structural and textual information.

Clustering method	Map size	Learning rate	Iteration	Radius	μ_1	μ_2	μ_3	Training time
SOM-SD using 1-D textual label	40000	0.7	120	20	0.002	0.95	–	1.5 hours
SOM-SD using 3-D textual label	40000	0.7	120	20	0.001	0.999	–	10 hours
CSOM-SD using 1-D textual label	90000	0.7	100	10	0.003	0.332	0.665	20 hours

The clustering training was conducted using both SOM-SD and CSOM-SD. The training parameters used are as shown in Table 2.

As Table 2 shows, the simplicity of the framework allows various textual information to be added to the structure. The 1-dimensional textual label simply uses the number of unique keywords in the various segments of the document, whereas the 3-dimensional textual label contains information about the 3 keywords with the highest frequency in each document segment. Here, the keywords are dictionary words, where the dictionary lists all words found in the training set but has common words such as “the”, “is”, “a”, “in”, etc. removed.

The initial SOM-SD training used a map size based on the number of unique sub-structures in the document structure tree which was quite small. However, since the training time is not long, the map size was increased to spread out overlapping nodes on the map and obtain better clustering results.

Similar to structure only clustering, the provision of appropriate training parameters is important, so the best combination of learning rate, number of training iterations and map updating radius have been attempted to arrive at the combination for each input data.

Again, the most influential parameter is the use of an appropriate weight value for the node label (μ_1), as it also determines the weight of children nodes (μ_2) in the structure. The selection of appropriate weight for the node label is dependent on the importance of the node label in the structure used. For this task, the existence and types of children nodes in this simple framework is quite important, therefore the weight on the node label (μ_1) should remain low.

The difference between the unique nodes and the unique sub-structures for this framework is approximately 4 times larger, so the map size for CSOM-SD is expected to be 4 times larger than its SOM-SD counterpart. Although the performance obtained through the CSOM-SD is expected to be better than SOM-SD, however, the long training time and the resulting accuracy is a trade-off.

Table 3. Test results for structure only clustering; a comparison.

Micro F1	Macro F1	Team	Method used
0.38	0.34	Our team	SOM-SD
0.37	0.33	Our team	CSOM-SD
0.27	0.22	Other team 1	not yet known
0.13	0.08	Other team 1	not yet known

Table 4. Test results and comparisons when clustering using XML structure and document content.

Micro F1	Macro F1	Team	Method used
0.35	0.29	Other team 1	not yet known
0.35	0.29	Other team 1	not yet known
0.25	0.20	Other team 1	
0.13	0.09	Our team	CSOM-SD with 1-D textual label
0.13	0.08	Our team	SOM-SD with 3-D textual label
0.11	0.07	Our team	SOM-SD with 1-D textual label
0.09	0.04	Other team 2	not yet known
0.09	0.04	Other team 2	not yet known
0.07	0.04	Other team 2	not yet known

5.5 Comparisons

The test results for structure only clustering are given in Table 3, ordered by micro F1 in descending order. The test results for structure and textual clustering are shown in Table 4. Note that we are unaware as to which approaches were used by the other teams since the various approaches are to be presented at the INEX'06 workshop.

As the tables showed, our team obtained the best performance in the structure only clustering. In fact, the performances obtained by using XML structure only also out-performed all the other teams involved in the structure and content clustering task (see Table 4). Although the CSOM-SD method does not appear to perform as well as the SOM-SD method, but bearing in mind that the map size provided for CSOM-SD is the same as the size used for SOM-SD, the CSOM-SD result may easily be improved.

Our performance in the structure and content could not achieve the similar standard, and we attribute this poor performance to the use of a significantly simplified structure which allowed the addition of textual information. However, the addition of textual information did not seem to add too much value to the clustering performance. This is perhaps due to the dramatic reduction of dimensionality for the textual information, which was carried out in order to keep the turn around time of the experiments reasonable.

6 Conclusions

It was shown that the SOM-SD can be a suitable tool for the clustering of relatively large sets of documents. While in general it is not necessary to pre-process the data when using this method, we found that pre-processing can help to substantially reduce training times. It is noted, however, that the network, once trained, can respond very quickly to large sets of test patterns.

We have furthermore demonstrated that XML structure is causal. This means that a contextual processing of such data is unlikely to bring any advantages, and thus, machine learning methods can be optimized by learning the structure in one direction only.

This paper addressed the inclusion of textual information into the clustering task. The results obtained were considerably worse than when learning XML structure only. We attribute this to an oversimplification of the training data through an aggressive pre-processing step which was engaged to achieve results in a timely fashion. It remains to be demonstrated how the SOM-SD method can perform when supplied with a richer set of structural and textual information.

Acknowledgments

The work presented in this paper received financial support from the Australian Research Council in form of a Linkage International Grant and a Discovery Project grant.

References

1. P. Frasconi, M. Gori, M. Maggini, E. Martinelli, and G. Soda. Inductive inference of tree automata by recursive neural networks. In *Proceedings of the Fifth Congress of the Italian Association for Artificial Intelligence*, pages 425–432, Rome, Italy, September 1997. Springer-Verlag.
2. M. Hagenbuchner, A. Sperduti, and A. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14(3):491–505, May 2003.
3. M. Hagenbuchner, A. Sperduti, and A. Tsoi. Contextual processing of graphs using self-organizing maps. In *European symposium on Artificial Neural Networks*, Poster track, Bruges, Belgium, 27 - 29 April 2005.
4. M. Hagenbuchner, A. Sperduti, and A. Tsoi. Contextual self-organizing maps for structured domains. In *Workshop on Relational Machine Learning*, 2005.
5. M. Hagenbuchner, A. Sperduti, A. Tsoi, F. Trentini, F. Scarselli, and M. Gori. Clustering xml documents using self-organizing maps for structures. In N. F. et al., editor, *LNCS 3977, Lecture Notes in Computer Science*, pages pp. 481–496. Springer-Verlag Berlin Heidelberg, 2006.
6. T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995.
7. F. Trentini, M. Hagenbuchner, A. Sperduti, A. Tsoi, F. Scarselli, and M. Gori. Clustering xml documents using self-organizing maps for structures. In *INitiative for the Evaluation of XML Retrieval (INEX)*, <http://xmlmining.lip6.fr/articles.html>, 28-30 November 2005.

FAT-CAT: Frequent Attributes Tree based Classification

Jeroen De Knijf

Universiteit Utrecht, Department of Information and Computing Sciences
PO Box 80.089, 3508 TB Utrecht
The Netherlands
jknijf@cs.uu.nl

Abstract. The natural representation of XML data is to use the underlying tree structure of the data. When analyzing these trees we are ensured that no structural information is lost. These tree structures can be efficiently analyzed due to the existence of frequent pattern mining algorithms that works directly on tree structured data. In this work we describe a classification method for XML data based on frequent attribute trees. From these frequent patterns we select so called emerging patterns, and use these as binary features in a decision tree algorithm. The experimental results show that combining emerging attribute tree patterns with standard classification methods, is a promising combination to tackle the classification of XML documents.

1 Introduction

In recent years there has been a growing interest from the knowledge discovery and data mining community in analyzing and mining XML data. The main reasons for this are the growing amount of semi-structured data and the widespread adoption and use of XML as the standard for semi-structured data. Frequent tree mining is a data mining technique that is able to exploit the information on the structure of the data present in XML-databases. Frequent tree mining is an instance of frequent pattern mining, specialized on tree structured data. Some well known algorithms are described in [2, 12, 13]. Briefly, given a set of tree data, the problem is to find all subtrees that satisfy the minimum support constraint, that is, all subtrees that occur in at least $n\%$ of the data records.

Classification is a major theme in data mining; the goal of classification is to predict the class of objects based on their attributes. Within the frequent pattern mining paradigm different classification approaches have been developed. In the work of Li et. al [10] classification is based on association rules i.e., it computes frequently occurring items associated with a class label. A drawback of this approach when applied to XML data is that the structure of XML data is lost. Frequent pattern based classification techniques that are suited for structured data are described in the work of Zaki and Aggarwal [14], Geamsakul et al. [8] and Bringmann and Zimmermann [5]. The first method computes frequent trees, orders these based upon some rule strength measures, and uses a decision list

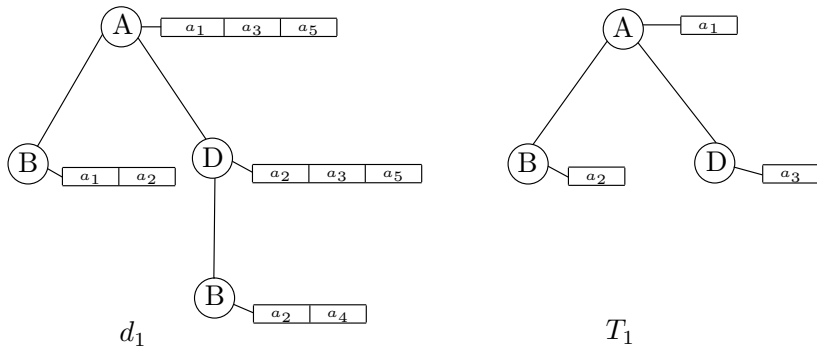


Fig. 1. T_1 is an induced subtree of the rooted ordered attribute tree d_1

approach to build a classifier. The other two methods compute interesting tree or graph patterns and use these as binary features in a standard classification method such as decision trees [11].

A drawback of the current frequent pattern based classification algorithms is that the existence of attributes associated with structured data is ignored, and hence potentially useful information is neglected. In previous work [9] we have described FAT-miner an efficient algorithm to mine tree structured data where attributes play an important role in the mining process. In this work we describe FAT-CAT; a classification method for XML data, based on frequent attribute trees. We applied this classification method to the Wikipedia [6] classification dataset, and discuss the results. Furthermore, these results are compared with those of a frequent pattern approach, that ignores the attributes of the dataset.

2 The Mining Algorithm

In this section we provide the basic concepts and notation used in this paper and describe the mining algorithm.

2.1 Preliminaries

A labeled rooted ordered attribute tree $T = \{V, E, \leq, L, v_0, M\}$ is an acyclic directed connected graph which contains a set of nodes V , and an edge set E . The labeling function L is defined as $L : V \rightarrow \Sigma$, i.e., L assigns labels from alphabet Σ to nodes in V . The special node v_0 is called the root of the tree. If $(u, v) \in E$ then u is the parent of v and v is a child of u . For a node v , any node u on the path from the root node to v is called an ancestor of v . If u is an ancestor of v then v is called a descendant of u . Furthermore there is a binary relation ' \leq ' $\subset V^2$ that represents an ordering among siblings. The size of a tree is defined as the number of nodes it contains; we refer to a tree of size k as a k -tree. The set of attributes is denoted by $\mathcal{A} = \{a_1, \dots, a_n\}$, where each attribute takes its

value from a finite domain. We further assume that there is an ordering among the attributes; i.e., $a_j \prec a_k$. To each node v in V , a non-empty subset of \mathcal{A} is assigned; we call this set the attributes of v . More formally: $M : V \rightarrow \mathcal{P}(\mathcal{A}) \setminus \{\emptyset\}$.

Given two labeled rooted attribute trees T_1 and T_2 we call T_2 an induced subtree of T_1 and T_1 an induced supertree of T_2 , denoted by $T_2 \preceq T_1$, if there exists an injective matching function Φ of V_{T_2} into V_{T_1} satisfying the following conditions for any $v, v_1, v_2 \in V_{T_2}$:

1. Φ preserves the labels: $L_{T_2}(v) = L_{T_1}(\Phi(v))$.
2. Φ preserves the order among the siblings: if $v_1 \leq_{T_2} v_2$ then $\Phi(v_1) \leq_{T_1} \Phi(v_2)$.
3. Φ preserves the parent-child relation: $(v_1, v_2) \in E_{T_2}$ iff $(\Phi(v_1), \Phi(v_2)) \in E_{T_1}$.
4. Φ preserves the attributes: $\forall v \in V_{T_2} : M(v) \subseteq M(\Phi(v))$.

In figure 1 an example data tree (d_1) is shown together with one of its subtrees. Let $D = \{d_1, \dots, d_m\}$ denote a database where each record $d_i \in D$, is a labeled rooted ordered attribute tree. Let $C = \{c_1, \dots, c_k\}$ be the k classes in the data. With D_{c_i} we denote the set of records in the database that has class label c_i , likewise with $D_{\bar{c}_i}$ the set of records in the database is denoted that has a class label different from c_i . For a given labeled rooted ordered attribute tree T , we say T occurs in a transaction d_i if T is a subtree of d_i . Let $\sigma_{d_i}(T) = 1$ if $T \preceq d_i$ and 0 otherwise. The support of a tree T in the database D is then defined as $\psi(T) = \sum_{d \in D} \sigma_d(T)$, that is the number of records in which T occurs one or more times. Likewise, the support within a class c_i of a tree T is defined as $\psi(T|c_i) = \sum_{d \in D_{c_i}} \sigma_d(T)$, that is the number of records with class label c_i in which T occurs one or more times. T is called frequent within the class c_i if $\psi(T|c_i)/|D_{c_i}|$ is greater than or equal to a user defined minimum support (*minsup*) value.

In general, the goal of frequent tree mining algorithms is to find all frequently occurring subtrees in a database. In the current setting, we are interested in all frequently occurring subtrees within the classes. Besides the frequency within a class constraint, an additional requirement is that a pattern is an emerging pattern for its class, i.e. patterns that occur often in one class and rarely in any other class. These patterns can have very discriminating properties, which is useful for classification. The parameters used for minimal support and whether a patterns is an emerging pattern will be discussed in section 3. The naive method to compute the desired patterns is to compute all subtrees, and select from these previously computed patterns the ones that are frequent. However, this is unfeasible from a computational point of view; the number of subtrees of a tree T is exponential in the size of T . To compute all the frequent patterns, the anti-monotonicity property of the support function ψ is used: $T_i \preceq T_j \Rightarrow \psi(T_i) \geq \psi(T_j)$. With this observation infrequent patterns can be pruned, which reduces the search space drastically.

2.2 FAT-miner

The main idea for the attribute tree mining algorithm is to split the mining process in a global mining stage and a local one. Loosely speaking the global mining

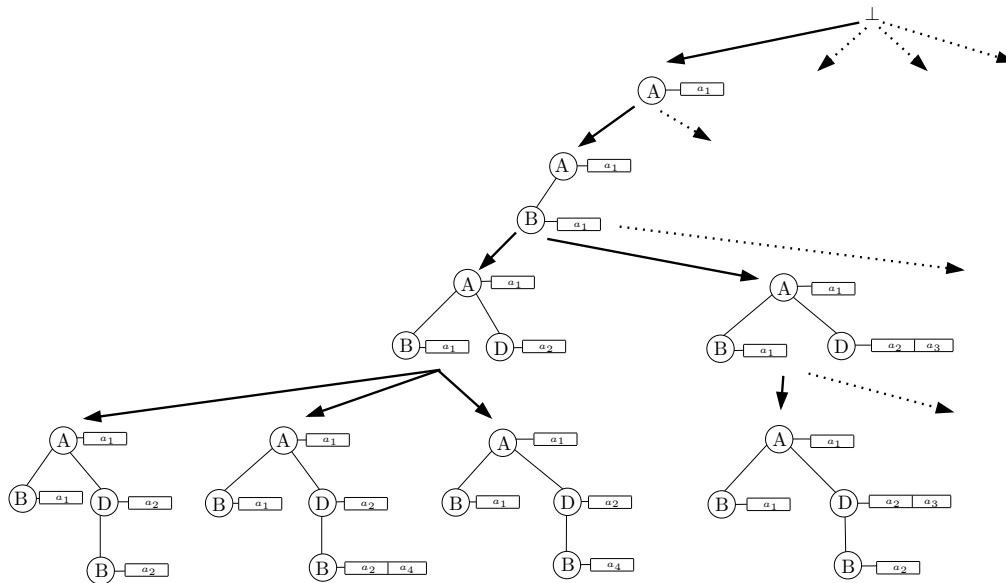


Fig. 2. Part of the enumeration tree for patterns on the tree d_1 (shown in figure 1).

part consist of a slightly modified rooted tree mining algorithm as described in the work of Asai et. al [2]: Enumeration of all frequent subtrees is accomplished by using the rightmost extension technique, that is, a $(k - 1)$ -tree is expanded to a k -tree by adding a new node *only* to a node on the rightmost branch of the $(k - 1)$ -tree. The local mining must be performed for every node of the subtrees. It boils down to the computation of frequent attribute sets from all the attributes to which the node of the subtree is mapped in the database. In this setting, the local mining algorithm is slightly different from existing frequent itemset mining algorithm [1, 3]. This difference is mainly due the fact that the node labels can occur multiple times in a tree. The local and global mining methods have to be combined: the idea is that whenever a candidate tree T (of size k) is generated, the local mining algorithm determines the first frequent attribute set of v_k , say A_1 . If there is none, then T must be pruned. Otherwise all supertrees of T are computed, where the attribute set of v_k equals A_1 . When there are no more frequent extensions of T left, the next frequent itemset of v_k is computed and this frequent pattern is then further extended. For both the global and local mining a depth-first search through the enumeration lattice is used. In figure 2, this principle is illustrated: the partial enumeration tree of d_1 is shown.

In figure 3 the pseudo code for the local mining algorithm is given. In the function `LocMine` in case of a non-empty attribute set X , we represent with a_k the attribute which comes last in the ordering of the attributes in X . Recall that we earlier defined that the set of attributes has as element with the highest

```

Function LocMine (  $X, OCL$ )
  if  $X = \emptyset$ 
  then
     $l \leftarrow 1$ 
  else
     $l \leftarrow k + 1$ 
  do
    while  $l \leq n$ 
      if  $support(X \cup a_l) \geq minsup$ 
      then
        ( $X \leftarrow X \cup \{a_l\}$ )
        return( $X, ComputeOcc(X, OCL)$ )
      else
         $l \leftarrow l + 1$ 
     $Y \leftarrow X$ 
    if  $X \neq \emptyset$ 
    then
       $X \leftarrow X \setminus \{a_k\}$ 
       $l \leftarrow k + 1$ 
    while  $Y \neq \emptyset$ 
  return ( $\emptyset, \emptyset$ )

Function ComputeOcc(  $X, OCL$ )
   $out \leftarrow \emptyset$ 
  for each  $d_i \in OCL$ 
  for each  $\Phi_{d_i}^j \in d_i$ 
  if  $X \subseteq M(\Phi_{d_i}^j(v_{k+1}))$ 
  then
     $out \leftarrow out \cup \Phi_{d_i}^j$ 
  return  $out$ 

```

Fig. 3. The local mining algorithm

```

Function Expand-Trees( $T, D$ )
   $out \leftarrow \emptyset$ 
  do
    ( $Aset, Nocc$ )  $\leftarrow$  LocMine( $M(T), occ(T, D)$ )
    if  $|Nocc| \geq minsup$ 
    then
       $M(v_{k+1}) \leftarrow Aset$ 
       $occ(T, D) \leftarrow Nocc$ 
       $out \leftarrow out \cup T$ 
       $C_{k+1} \leftarrow$  candidates generated from  $T$ 
      for each  $c_{k+1} \in C_{k+1}$ 
         $out \leftarrow out \cup$  Expand-Trees ( $c_{k+1}$ )
    while  $|Nocc| \geq minsup$ 
  return  $out$ 

Algorithm FAT-miner(database  $D$ )
   $out \leftarrow \emptyset$ 
   $C_1 \leftarrow$  candidate one patterns
  for each  $T \in C_1$ 
     $out \leftarrow out \cup$  Expand-Trees( $T, D$ )
  return  $out$ 

```

Fig. 4. The global mining algorithm.

order a_n . First, for every possible extension of X with a higher ordered attribute than a_k , the frequency is determined. If one of these extensions is frequent, the function **ComputeOcc** is called. This function determines all mappings in a list (occurrence list), for which the rightmost node of the mapping covers the

frequent extension. If none of the previous extensions is frequent, a_k is removed from X and X is again extended with attributes.

In the global mining algorithm, as described in figure 4, candidate trees are generated by adding a node on the rightmost path of the tree. For each candidate one-pattern the function `Expand-Trees` is called. This function first calls the local mining function, which determines the next frequent attribute set. If there is one, this attribute set is assigned to the right-most node of the current tree, and the result is added to the solution. Then the occurrence list is updated and this tree, with the updated occurrence list, is further extended. Otherwise the current tree is pruned.

3 XML Document Classification with FAT-CAT

The global procedure for the classification of XML documents is as follows:

1. Compute the frequent patterns for the different classes on the training set.
2. Select from these frequent patterns the emerging patterns.
3. The emerging patterns are used to learn the classification model on the training set.
4. Evaluate the classification model on the test set.

To compute the frequent patterns we still have to determine the minimum support value. This, must be done very precisely: when it is set too high, only patterns that are already ‘common knowledge’ will be found. On the other hand, if it is set too low we have to examine a massive number of patterns if they are useful for the classification task. When dealing with multiple classes in frequent pattern mining, the question arises whether a single support value is sufficient for all different parts of the database. Given two parts D_{c_i} and D_{c_j} , it may be the case that the structural similarity in one part is much higher than the structural similarity in the other part. Hence, in the part with the higher structural similarity a higher support value is preferred. Therefore we have chosen to use k different minimum support values; one for each class label. To determine an appropriate minimum support value for each class, we started with a high support value, and lowered it gradually until a sufficient number of high quality patterns was produced. As sufficient criterion we used that there were at least ten different frequent patterns T of which each has the following property: $\psi(T|c_i)/\psi(T) \geq 0.6$, i.e. the probability of a particular class given pattern T must be greater than or equal to 0.6. This property is also known as the confidence of the rule $T \rightarrow \text{class}$. When this procedure for finding proper support values was applied to the training set, for 53 out of 60 class labels we founded a sufficient number of high quality frequent patterns; these 53 classes together contained roughly about 99% of all XML documents in the dataset.

Having settled the support values for all classes, we still need to select ‘interesting’ patterns. Since the overall goal is to classify XML documents, we are more specifically interested in patterns that describe local properties of particular groups in the data. Notice that not all computed frequent patterns have this

property. These interesting patterns are often called discriminative or emerging patterns [7], and are defined as patterns whose support increases significantly from one class to another. For emerging patterns to be significant, different measures are used. One measure of interest is by what factor observing a pattern changes the class probability, compared to the class prior probability, i.e., $P(\text{class}|T)/P(\text{class})$ also known as the lift of the rule $T \rightarrow \text{class}$. Another commonly used measure is the confidence of the rule $T \rightarrow \text{class}$. However, this measure is too tight for the purpose of multi-class classification. This is mainly because the patterns produced would jointly only cover a relatively small part of the records. In this work we used the lift measure. As a result the emerging patterns on the training set covered 99% of the records in the test set; compared with only 51% coverage when the confidence measure was used.

In order to learn a classification model, the next step is to construct binary features for every XML document. Each feature indicates the presence or absence of an emerging pattern in a record. We used binary decision trees [11] to learn a classification model, more specifically the implementation provided by Borgelt [4]. The classification model was build constructed with all parameters set to the default value, except for the confidence level pruning value, which was set to 0.9.

4 Experimental Results

Besides the evaluation of the classification model on XML documents, we also experimentally investigate whether the inclusions of attribute values improves the performance of the classifier. For this purpose, we trained two classifiers following the procedure described earlier: one where we used the tree structure and the attributes associated to the nodes of the tree, and a second one where only the tree structure was used. These classifiers were trained and evaluated on the Wikipedia XML dataset [6] as provided for the document mining track at INEX 2006. The collection consists of a training set and a test set, which both contain 75,047 XML documents with 60 different class labels. The distribution of the classes over the test set is shown in figure 5. The classifiers were trained on a random sample of approximately one third of the original training set. A sample was taken such that the mining algorithms used were able to run this dataset on a desktop computer with 500MB of main memory. Besides the memory constraint, the running time for the tree mining algorithms already took quite some time (about a week).

The sample used consisted of 25,127 trees and 8,310 distinct node labels; of these nodes 389 contained attributes. To model XML data as attribute trees, a dummy attribute was assigned to each node in the database that had no attributes. The average number of nodes in the trees equals 85, with each attribute counted as a single node, the average size of the trees was 128.

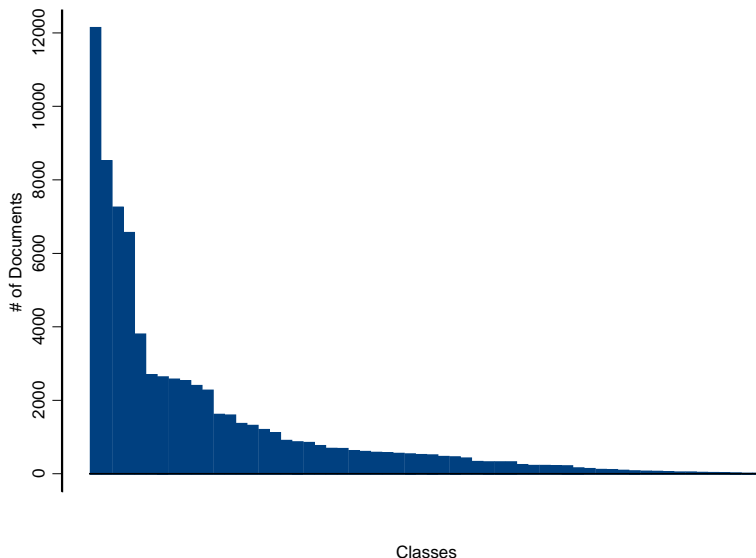


Fig. 5. Distribution of the XML documents over the different classes, classes are sorted according to the number of documents they contain.

	ATR	NOATR
Micro-average F1	0.479802	0.338321
Macro-average F1	0.527043	0.338920

Fig. 6. Micro-average F1 and Macro-average F1 classification results on the test set.

4.1 Results

The document mining track evaluation measures are precision, recall, micro-average F1 and macro-average F1. Macro-average F1 gives equal weight to each class, while micro-average F1 is a per document measure, so it is heavy influenced by larger classes. Furthermore, the $F1$ measure is the harmonic mean of precision and recall: $\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$. We use ATR as abbreviation of the classifier in which we used the tree structure and the attributes associated to nodes, likewise NOATR is the classifier in which we only used the tree structure. The macro and micro-average F1 scores on the test set are shown in figure 6. As expected the scores for the ATR classifier are substantially higher than the scores for NOATR. A closer investigation of the patterns used for the ATR classifier, revealed that the main advantage of including attribute values is that these val-

ues often describe the document to which a link points. For example in figure 7 and figure 8 example emerging patterns are shown. However, due to the very common structural properties of these patterns, only the attribute values in-sures that these are emerging patterns. In the example shown in figure 7, the attribute value points to a Wikipedia page listing all record labels. In the second example shown in figure 8, the attribute value points to a picture that was the US Navy Jack for some time. Clearly, both values are good indicators of their classes (“Portal:Music/Categories” and “Portal:War/Categories”).

```

<section>
<title>See also</title>
  <normallist>
    <item>
      <collectionlink xlink:type="simple" xlink:href="193135.xml">List of record labels</collectionlink>
    </item>
  </normallist>
</section>

```

Fig. 7. An emerging pattern found on the training set (for clarity displayed with text). This pattern describes class 1474144 (“Portal:Music/Categories”) and has a support of 402 in its class and 0 for all other classes.

```

<figure>
  <image xlink:type="simple" xlink:href=" ../pictures/USN-Jack.png" </image>
</figure>

```

Fig. 8. An emerging pattern found on the training set. This pattern, describing class 2879927 (“Portal:War/Categories”) has a support of 48 in its class and 0 for all other classes.

In figure 9 and figure 10 the recall and precision values per class are plotted, for both classifiers. In both cases the classes were sorted according to the highest value for the ATR classifier. In almost every class the recall values for the ATR classifier are better than the recall values for the NOATR classifier. However, in three classes the recall value for the ATR classifier was lower than the recall values for the NOATR classifier. Hence, for some classes the inclusion of attributes has a negative influence on the classification performance. Furthermore, there are a couple of classes where both classifiers were unable to retrieve any document belonging to that class. Especially classes that contained a small number of documents suffered from this. It is interesting to note the large different in performance between different classes, for example: the ATR classifier for class 148035 (“Portal:Art/Categories”) achieved a recall score of 0.262338 and a precision score of 0.312677 while, for class 2257163 (“Portal:Pornography/Categories”) a recall score of 0.944444 and a precision score of 0.924686 was achieved. In order to accomplish a higher performance for the first class, we experimented

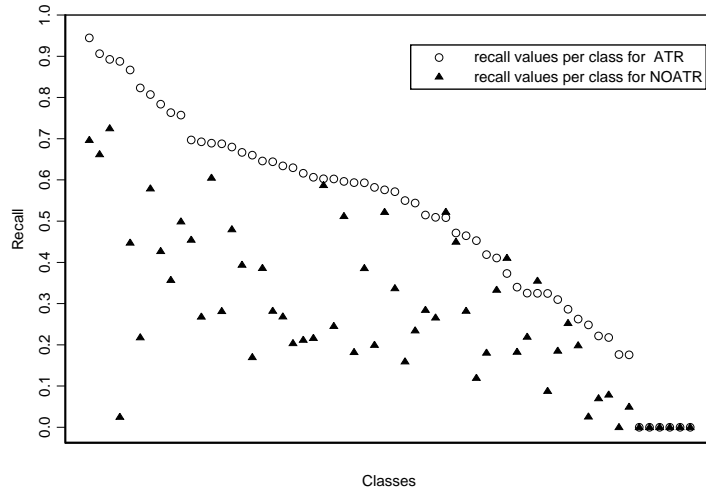


Fig. 9. The recall values per class, both for the experiments where we used the attributes of the data and those where the attributes were ignored.

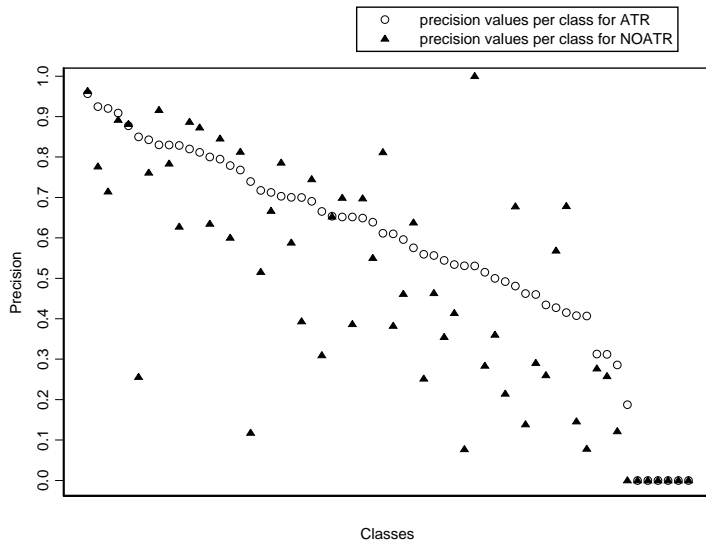


Fig. 10. The precision values per class, both for the experiments where we used the attributes of the data and those where the attributes were ignored.

with lowering the support value for the class. Unfortunately, the additional patterns did not result in better performance. This suggests, that in the current framework, using only the structure of XML documents is insufficient for classification. Generally speaking, the precision values achieved are better for the ATR classifier than for the NOATR classifier. The most likely explanation for the fact that the NOATR classifier achieved a higher precision for fifteen classes is that the recall achievement of this classifier are rather poor for these classes. However, this does not hold for all classes, for example on class 1507239 ("Portal:Aviation/Categories") the NOATR classifier achieved higher scores both for recall and precision. A closer investigation of the emerging patterns for this class is needed to give a possible explanation.

5 Conclusion

In this work we presented FAT-CAT; an XML classification approach based on frequent attribute trees, and compared these with a frequent tree approach. We have shown that the inclusion of attributes improves the performance of the classifier. Furthermore, we analyzed the added value of including attributes into the classification process and describe weaknesses of the used approach.

Further research includes the combination of the current classification approach with more context oriented classification techniques, such as text mining.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499, 1994.
2. T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In *SIAM Symposium on Discrete Algorithms*, 2002.
3. R. Bayardo. Efficiently mining long patterns from databases. In A. T. Laura and M. Haas, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 85–93, 1998.
4. C. Borgelt. A decision tree plug-in for dataengine. In *Proc. 6th European Congress on Intelligent Techniques and Soft Computing*, 1998.
5. Björn Bringmann and Albrecht Zimmermann. Tree² - decision trees for tree structured data. In *European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 46–58, 2005.
6. L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 2006.
7. G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 43–52, 1999.
8. W. Geamsakul, T. Yoshida, K. Ohara, H. Motoda, H. Yokoi, and K. Takabayashi. Constructing a decision tree for graph-structured data and its applications. *Fundamenta Informaticae.*, 66(1-2):131–160, 2005.
9. J. De Knijf. FAT-miner: Mining frequent attribute trees. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, 2007. to appear.

10. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 80–86, 1998.
11. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
12. K. Wang and H. Liu. Discovering structural association of semistructured data. *Knowledge and Data Engineering*, 12(2):353–371, 2000.
13. M. J. Zaki. Efficiently mining frequent trees in a forest. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2002.
14. M. J. Zaki and C. C. Aggarwal. Xrules: an effective structural classifier for XML data. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 316–325, 2003.

XML Structure Mapping

Application to the PASCAL/INEX 2006 XML Document Mining Track

INEX 06 Preproceedings paper *

Francis Maes, Ludovic Denoyer and Patrick Gallinari

LIP6 - University of Paris 6

Abstract. We address the problem of learning to map automatically flat and semi-structured documents onto a mediated target XML schema. We propose a machine learning approach where the mapping between input and target documents is learned from examples. Complex transformations can be learned using only pairs of input and corresponding target documents. From a machine learning point of view, the structure mapping task raises important complexity challenges. Hence we propose an original model which scales well to real world applications. We provide learning and inference procedures with low complexity. The model sequentially builds the target XML document by processing the input document node per node. We demonstrate the efficiency of our model on two structure mapping tasks. Up to our knowledge, there are no other models yet able to solve these tasks.

1 Introduction

Semantically rich data like textual or multimedia documents tend to be encoded using semi-structured formats. Content elements are organized according to some structure that reflects logical, syntactic or semantic relations between these elements. For instance, XML and, to a lesser extent, HTML allow us to identify elements in a document (like its title or links to other documents) and to describe relations between those elements (e.g. we can identify the author of a specific part of the text). Additional information such as metadata, annotations, etc., is often added to the content description leading to richer descriptions.

The question of heterogeneity is central for semi-structured data: documents often come in many different formats and from heterogeneous sources. Web data sources for example use a large variety of models and syntaxes. Although XML has emerged as a standard for encoding semi-structured sources, the syntax and semantic of XML documents following different DTDs or schemas will be different. For managing or accessing an XML collection built from several sources, a correspondence between the different document formats has to be established.

* This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

Note that in the case of XML collections, the schemas themselves may be known or unknown depending on the source. For HTML data, each site will develop its own presentation and rendering format. Thus even in the case of HTML where the syntax is homogeneous across documents, there is a large variety of formats. Extracting information from different HTML web sites also requires to specify some type of mapping between the specific Web sites formats and the predefined format required by an application.

Designing structure mappings, in order to define correspondences between the different schemas or formats of different sources is thus a key problem to develop applications exploiting and accessing semi-structured sources. This problem has been addressed for some times by the database and to a lesser extent by the document communities for different conversion tasks and settings. Anyway, the real world solution is to perform a manual correspondence between heterogeneous schemas or towards a mediated schema via structured document transformation languages, like XSLT. Given the multiplicity and the rapid growth of information sources, manually specifying the correspondence between sources is clearly a bottleneck to the process of document integration and reuse. Automating the design of these transformations has rapidly become a challenge.

This work was realized in the context of the PASCAL/INEX XML Document Mining Challenge¹. This challenge proposes, as an extension to XML categorization and clustering, a track concerning the Structure mapping task. The goal of this task is to learn to transform HTML/flat document to an XML mediated schema as described previously.

We propose here to learn the transformation from examples. The learning system relies on a training set provided by the user. Each training example is made of an input document and the corresponding target document. The system will directly learn the transformation from these examples. Input documents may come with heterogeneous structures or simply with no structure at all. The manual specification of document mappings is replaced here with the development of a training set of transformed documents. This allows to consider problems where the input schema is not explicitly given or cases where this schema is too general so that no explicit mapping can be defined (this is the case for many HTML conversion applications). Besides, the proposed method only requires to provide a set of transformed documents and this task will be much easier than the manual development of a transformation script.

The structure mapping task we are solving is described in section 2. Our solution is detailed in part 3 and experiments performed on two different real world tasks are presented and discussed in section 4.

2 Structure Mapping Task

2.1 Description

The structure mapping task addresses the problem of learning document transformations given a set of examples. The task is seen as a supervised learning

¹ <http://xmlmining.lip6.fr>

problem where inputs and outputs are semi-structured documents. Input documents may come from different sources and may take different formats like flat text, wikitext, HTML or different XML schemas. Output documents are expressed in XML. Given the set of learning examples, the aim is to learn an inference procedure able to convert any input document of the same family.

The structure mapping task encompasses a large variety of real applications like:

- *Semantic Web*: conversion from raw HTML to semantically enriched XML. Example sources include forums, blogs, wiki-based sites, domain specific sites (music, movies, houses, ...).
- *Wrapping of Web pages*: conversion from the relevant information in web-pages to XML.
- *Legacy Document conversion*: conversion from flat text, loosely structured text, or any other layout oriented format (*e.g.* PDF) to XML.

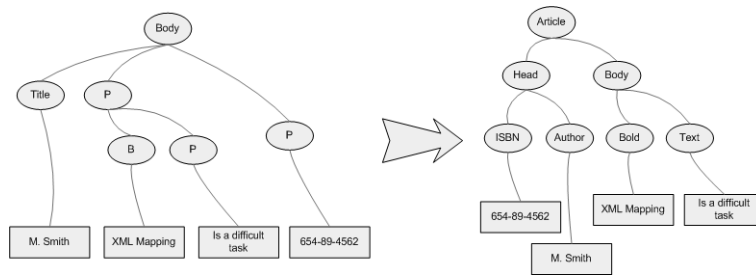


Fig. 1. Example of web page wrapping. This is a case of HTML to XML Structure Mapping. Nodes can be relabelled, inserted, suppressed and moved.

Figure 1 illustrates an example of XML to XML conversion. As can be seen in this simple example the structure mapping task involves many different kind of elementary transformations, *e.g.* relabelling, node creation and suppression, node displacement. These actions can have global consistency constraints, *e.g.* conserving textual content order or being valid with respect to a DTD.

2.2 Formalization

Structure mapping consists in transforming $d \in D_{in}$ into an XML document $d^* \in D_{out}$ where D_{in} is the set of possible input documents and D_{out} is the set of possible output documents. For example, D_{in} is the set of all documents that are valid given one or multiple specific XML schemas specific DTDs. As training data, a user provides a set of pair $\{(d_i, d_i^*)\}_{i \in [1, N]}$ where N is the number of examples, d_i is an input document in D_{in} and d_i^* is the corresponding output document in D_{out} .

A structure mapping model is a function $f_\theta : D_{in} \rightarrow D_{out}$ that maps input documents into target documents. The quality of a structure mapping can be measured with a user supplied loss-function. This function is a dissimilarity measure between output documents : $\Delta : D_{out} \times D_{out} \rightarrow [0, 1]$. Good models will produce low loss. Learning is done by finding the parameters θ that minimize the empirical risk on the training set:

$$\theta^* = \operatorname{argmin}_\theta \frac{1}{N} \sum_{i=1}^N \Delta(f_\theta(d_i), d_i^*) \quad (1)$$

Structure mapping models have to deal with two major difficulties. First, they have to support a large variety of transformation. The family f_θ must thus be expressive enough to express them. The other difficulty is related to the size of D_{out} which is exponential in the length of documents. A such problem makes full exploration of the output space intractable. The usual solution for this type of problem is to use dynamic programming techniques in order to efficiently explore the space of possible solutions. For the document transformation problem addressed here, the complexity of the inference step is so high that even dynamic programming does not lead to scalable solutions [1].

3 Proposed Model

One way to break the complexity of the structure mapping task, is to decompose it into simpler subproblems. In order to accomplish this, we propose to model the *sequential* process of structure mapping. The whole mapping is decomposed into a sequence of local decision steps. The mapping is fulfilled by iterating over input nodes and taking one decision for each of them. The structure mapping loss depends thus on the entire sequence of chosen decisions. In order to model this decision process, we rely on the *Deterministic Markov Decision Process* (DMDP) framework. This framework allows us decompose the structure mapping loss into individual per-decision losses. Moreover it is possible to use well known Reinforcement Learning (RL) algorithms [2] which solves the learning problem.

3.1 Deterministic Markov Decision Process

Markov Decision Processes (MDP) [3] provide a mathematical framework for modelling sequential decision-making problems. They are used in a variety of areas, including robotics, automated control, economics and in manufacturing. As a special case of them, we consider in this paper Deterministic Markov Decision Process. Before describing the model, we first briefly establish some notations related to DMDPs.

A DMDP is a tuple $(S, A, \delta(.,.), r(.,.))$ where S is the state space, A is the action space, $\delta: S \times A \rightarrow S$ is the transition function and $R: S \rightarrow \mathfrak{R}$ is the reward function. At any time the process is in a given state $s \in S$. In each state s there are several actions in A from which the model must choose. For a state s and an

action a , the transition function defines the next state $\delta(s, a)$. The model earns a reward $r(s)$ for each state s visited. A decision maker is called a *Policy*. A Policy π is a function that maps states to action probabilities. $\pi(s, a) \in [0, 1]$ denotes the probability to select action a in state s . Learning algorithms attempt to find a policy that maximizes cumulative reward over the course of the problem.

We show now how to model structure mapping as a DMDP. Figure 2 gives an example of such an DMDP. The main points of our model are:

- The input document is processed node per node. We denote n_i the input document node with index i . In the case of an HTML/XML source, the nodes n_i are the leafs of the tree. In the case of flat segmented text, the nodes n_i are the text segments.
- When processing input node n_i , the state contains the partial output document corresponding to input nodes n_1, \dots, n_{i-1} .
- For each input node, there are two possible kind of actions: skip this node, or use it to create new nodes in the partial output document.
- Each decision is given a reward such that the sum of all rewards corresponds to the structure mapping loss.

We now present the detail of the structure mapping DMDP.

States A state contains the input document d , the index of the current input node t and the current output document \hat{d}_t . The initial output document \hat{d}_1 contains a single node: the XML root. Each state corresponds to a different partial output document. At each time step, the model has to take a decision for input node n_t . All input nodes have been processed when $t > \|d\|$. In this case, the state is final and contains the complete output document. We note these state s_{\perp} and the corresponding output document \hat{d} . \hat{d} is the prediction made by our structure mapping model.

Actions In a given state s , the model has to choose what to do with input node n_t . The model can choose between the **Skip** and the **Create(Tags, Positions)** actions. **Skip** makes the system simply continues with the next input node. In this case n_t will not appear in the output document. **Create(., .)** is used to select the content of n_t and to use it to create new nodes in the output. **Tags** defines the sequence of tags from the root node to the content node that is created.² **Positions** defines where the new nodes will be connected on the existing tree. This set of actions is very similar to the one used in [4] for the natural language parsing task. The difference with parsing is that for structure mapping, we allow to throw input nodes and to change the order of the leaves during the transformation. In order to reduce the number of possible actions, some external information (like DTD or XML schema) can be used to constraint **Tags** and **Positions**.

² This parameter can range on all the tags sequences that were observed at least once in the training set.

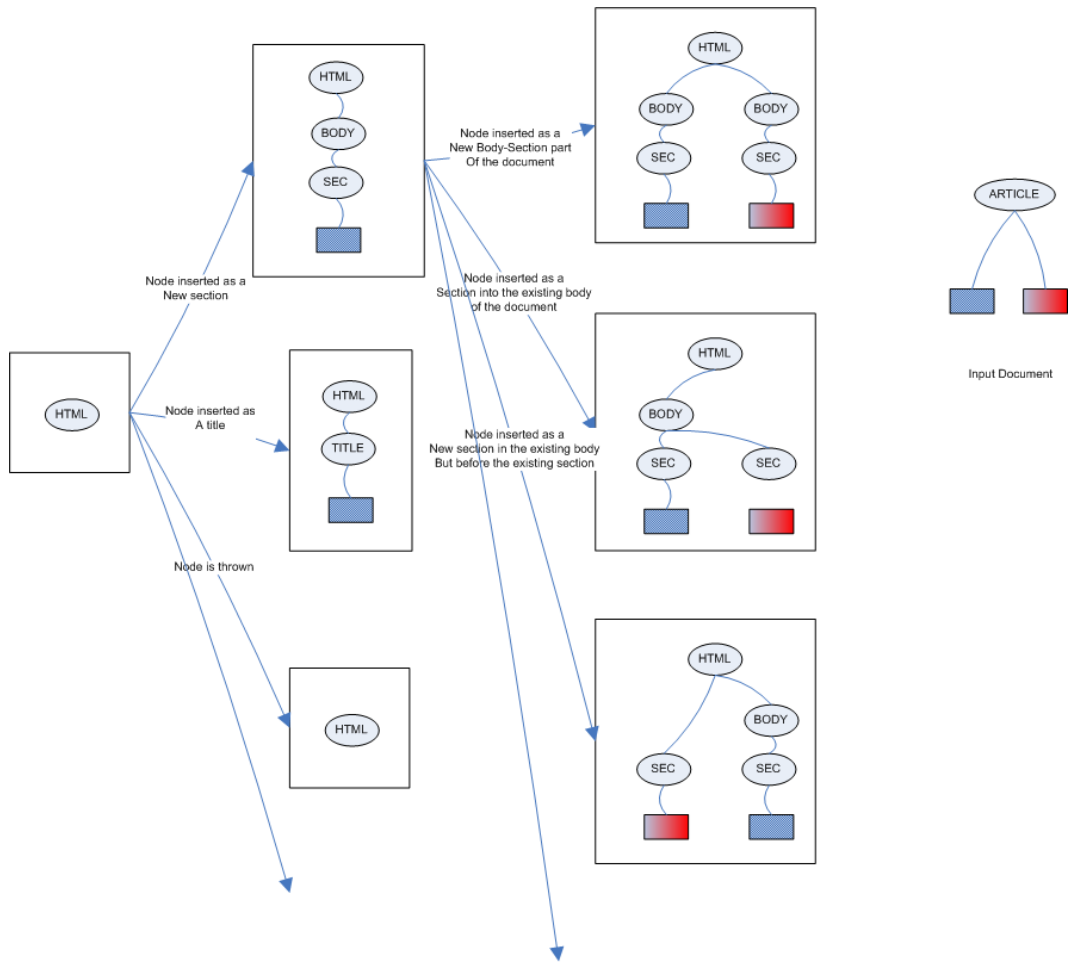


Fig. 2. This figure is an example of DMDP. The states correspond to partial output trees. The initial state (at the left) is the state with the empty document (one HTML node). Nodes to the right are potential output documents. The edges are the possible actions. In this DMDP, we first choose an action for the first input node (blue square node), and then choose an action corresponding to the second input node (red square leaf).

Transitions The transition function increments the input node index t , applies the action a on the current partial output document and returns the resulting state.

Rewards Learning algorithms attempt to maximize the cumulative reward over the course of the problem. Therefore we have to design reward functions such that maximizing cumulative reward corresponds to minimizing the loss function $\Delta(d, \hat{d})$. We propose two reward functions which have this property. In the simplest case, the reward is only given once the output document is finished. Our reward is null for all states except the final ones. In final states, the reward is the negative loss between the output document and the true document:

$$r_1(s_t) = \begin{cases} -\Delta(\hat{d}, d^*) & \text{if } s_t \text{ is a final state,} \\ 0 & \text{otherwise.} \end{cases}$$

The cumulative reward of r_1 is:

$$\begin{aligned} R_1(s_1, \dots, s_{\parallel d\parallel}, s_{\perp}) &= \sum_{i=1}^{\parallel d\parallel} r_1(s_i) + r_1(s_{\perp}) \\ &= r_1(s_{\perp}) \\ &= -\Delta(\hat{d}, d^*) \end{aligned}$$

In order to help learning convergence, we also introduce a reward that is defined on all states. In this case the action that transforms \hat{d}_{t-1} into \hat{d}_t is given reward

$$r_2(s_t) = \begin{cases} -\Delta(\hat{d}_1, d^*) & \text{if } t = 1, \\ -\Delta(\hat{d}_t, d^*) + \Delta(\hat{d}_{t-1}, d^*) & \text{otherwise.} \end{cases}$$

The cumulative reward of r_2 is:

$$\begin{aligned} R_2(s_1, \dots, s_{\parallel d\parallel}, s_{\perp}) &= \sum_{i=1}^{\parallel d\parallel} r_2(s_i) + r_2(s_{\perp}) \\ &= -\Delta(\hat{d}_1, d^*) + \Delta(\hat{d}_1, d^*) - \Delta(\hat{d}_2, d^*) \dots \\ &\dots + \Delta(\hat{d}_{\parallel d\parallel}, d^*) - \Delta(\hat{d}, d^*) \\ &= -\Delta(\hat{d}, d^*) \end{aligned}$$

It is thus clear that maximizing the cumulative reward R_1 or R_2 is equivalent to minimizing the example loss of the structure mapping model. Notice that our reward functions are only available when the exact output document d^* is known. This means that the reward can not be computed on the testing set. Although this is unusual in MDPs this is not a problem: learning aims at finding a policy that maximizes the cumulative reward on the training set. Once the policy is learned it can be used on the testing set without any reward feedback.

3.2 Reinforcement learning

Markov Decision Processes can be solved using well known reinforcement learning algorithms. These algorithms learn the parameters θ of a policy π_θ such that π_θ maximizes the expected cumulative reward. We are interested in seeking a deterministic policy for our decision process. A simple way to model such policies is to use an *Action-Value Function* that scores each possible action a in a given state s . More precisely we define a function $Q : S \times A \rightarrow \mathfrak{R}$ where $Q_\theta(s, a)$ is the expectation of the cumulative reward that will be earned by choosing a in state s . The deterministic policy is defined in terms of the Q function:

$$\pi_\theta(s, a) = \begin{cases} 1 & \text{if } a = \operatorname{argmax}_{a'} Q_\theta(s, a') \\ 0 & \text{otherwise.} \end{cases}$$

The Q function is modelled over all possible input documents and all possible states so it can't be stored as if - in a table for example. It must be approximated. Using value function approximators is an active research area [5]. A simple and efficient solution is to use linear approximators:

$$Q_\theta(s, a) = \langle \theta, \phi(s, a) \rangle$$

where $\phi(s, a)$ is a vectorial description of the state-action pair, θ is a vector of weights and $\langle ., . \rangle$ is the classical dot product. $\phi(s, a)$ is a vector that jointly describes the current state and the action we want to evaluate. Many different representations ϕ can be used. We show in table 1 the type of features we have used. The size n of the feature vectors directly influences the training time and the performance of our model. θ contains one weight per feature available in $\phi(s, a)$.

Several well known algorithms exist for learning the parameters θ , *e.g.* Q-Learning, Sarsa(0), Sarsa(λ). In our experiments we have used Sarsa(0). Briefly the idea is to iteratively correct θ such that $Q_\theta(s_t, a_t)$ approximates the sum of the immediate reward $r(s_{t+1})$ and the next cumulative expected reward $Q_\theta(s_{t+1}, a_{t+1})$. Actions are chosen following an ϵ -greedy policy: with probability ϵ the policy chooses a random exploratory action, and with probability $1 - \epsilon$ it chooses the action maximizing $Q(s, a)$. See [2] for more details.

3.3 Complexity

In order to perform inference the model has to evaluate all $Q(s, a)$ values for each state s it visits. This gives us the inference complexity $O(\|d\| * \|\hat{A}_s\|)$ where $\|\hat{A}_s\|$ corresponds to the mean number of possible actions for each state s . This complexity is much lower than the $O(\|d\|^3 * \|\hat{A}_s\|)$ complexity of most classical dynamic programming tree inference procedures. Inference is thus tractable on large document transformations.

4 Experiments

4.1 Tasks and Corpora

We present here experiments performed in the context of the INEX Structure Mapping Challenge. The challenge focuses on two corpora. The first one is the INEX corpus which is composed of 12017 scientific articles in XML format. Each document comes from a journal (18 possible journals). The documents are given in two versions: a flat segmented version and the XML version. The structure mapping task aims at recovering the XML structure using only the text segments as input. The segments are given in the exact order. The second corpora is made of 1390 movie descriptions available in three versions: two different HTML versions and one mediated XML version. This corresponds to a scenario where two different websites have to be mapped onto a predefined mediated schema. The transformation includes node suppression and some node movings.

4.2 Features, Loss Function and Evaluation Measures

The second word of the input node is made of lower case letters and we are creating a node with label <code>LINE</code> .
The first word of the input node is made of one upper case letter followed by lower case letters and we are creating a node with label <code>ISBN</code> .
The first word of the input node is made of one lower case letter followed by digits letters and the node we are creating follows a <code>SECTION</code> node.
The first word is a symbol and we are creating a node which follows a <code>TITLE</code> .
The input node contains between 5 and 10 words and we are creating a node at height 2.
The input node parent's tag is <code>H2</code> and we are creating a node which follows a <code>PARAGRAPH</code> .
The input node has tag <code>H3</code> and we are skipping it.
The first word of the input node is a symbol and we are skipping this node.
...

Table 1. Some examples of features of $\phi(s, a)$ which jointly describe the current state s and the action a . These features are usually binary features (in $0, 1$) but real valued features are also possible. The features are generated in a data-driven way: a feature is considered only once it is observed once in the learning data.

The model uses a sparse vectorial representation $\phi(s, a)$ of the state s and action a (see section 3). Table 1 gives some examples of the features we have used. These features jointly describe aspects of the current state and aspects of the current actions (*e.g.* related to the `Tags` and `Positions` parameters. Depending

on the corpora, we often have more than one million distinct features. Each corpus is split in two parts: 50% for training and 50% for testing.

The loss-function $\Delta(\hat{d}, d^*)$ is based on a F1 Score that reflects the proportion of common subtrees between \hat{d} and d^* . The score is computed in the following way:

1. Build the set of all subtrees of the two compared tree. There is one subtree per node of the document
2. Compute recall and precision on the subtrees. Two subtrees are identical iff they have the same label, the same text (for leaf nodes), and the same children trees (for internal nodes).
3. Compute the F1 score: $F1 = \frac{2 * Recall * Precision}{Recall + Precision}$.
4. $\Delta(\hat{d}, d^*) = 1 - F1(\hat{d}, d^*)$

This measure corresponds to the usual F1 score used in the field of Natural Language parsing but it is slightly different since the model allows us to throw content nodes of the input document. In order to highlight the score distribution upon the documents, we also provide the percentage of documents from the test corpus with a F1 score greater than x%, for x between 0% and 100%.

4.3 Results

Our experiments are still running and we have already obtained a large set of nice results over different corpora. Final results on the XML Document Mining corpora will be available and presented at the INEX Workshop in December in Dagstuhl.

5 Related Work

Several approaches to automating document transformation have been explored ranging from syntactic methods based on grammar transformations or tree transducers to statistical techniques. A majority of them only consider the structural document information and do not exploit content nodes. Even this structural information is used in a limited way and most methods exploit only a few structural relationships. Many of them heavily rely on task specific heuristics. Current approaches to document transformation are usually limited to one transformation task or to one type of data. Besides, most proposed techniques do not scale to large collections.

In the database community automatic or semi-automatic data integration — known as *schema matching* — has been a major concern for many years. A recent taxonomy and review of these approaches can be found in [6]. [7] describes one of the most complete approach which can handle ontologies, SQL and XML data. The matching task is formulated as a supervised multi-label classification problem. While many ideas of the database community can be helpful, their corpora are completely different from the textual corpora used in

the IR community: all documents — even XML ones — keep an attribute-value structure like for relational database and are thus much smaller and more regular than for textual documents; textual data hardly appears in those corpora. With database corpora, finding the label of a piece of information is enough to build the corresponding tree because each element usually appears once in the tree structure. Document structure mapping, also shares similarities with the information extraction task, which aims at automatically extracting instances of specified classes and/or relations from raw text and more recently from HTML pages. Recent works in this field [8] have also highlighted the need to consider structure information and relations between extracted fields.

The structure mapping model proposed here is related to other Machine Learning models of the literature. Different authors ([9], [10]) have proposed to use natural language formalisms like probabilistic context free grammars (PCFG) to describe the internal structure of documents. Early experiments [1] showed that the complexity of tree building algorithms is so high that they cannot be used on large corpora like INEX. The work closest to ours is [11]. They address the HTML to XML document conversion problem. They make use of PCFGs for parsing text segment sequences and of a maximum entropy classifier for assigning tags to segments.

6 Conclusion

We have described a general model for mapping heterogeneous document representations onto a target structured format. This model learns the transformation from examples of input and target document pairs. It is based on a new formulation of the structure mapping problem based on Deterministic Markov Decision Processes. This formulation allows us to deal with a large variety of tasks ranging from the automatic construction of a target structure from flat documents to the mapping of XML collections onto a target schema. The model operates fast and scales well with large collections. We have shown its efficiency on two real world large scale tasks.

References

1. Denoyer, L., Wisniewski, G., Gallinari, P.: Document structure matching for heterogeneous corpora. In: SIGIR 2004. Workshop, Sheffield (2004)
2. Sutton, R., Barto, A.: Reinforcement learning: an introduction. MIT Press (1998)
3. Howard, R.A.: Dynamic Programming and Markov Processes. Technology Press-Wiley, Cambridge, Massachusetts (1960)
4. Collins, M., Roark, B.: Incremental parsing with the perceptron algorithm. In: Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume, Barcelona, Spain (2004) 111–118
5. J. Si, A. G. Barto, P.W.B., II, D.W.: Handbook of Learning and Approximate Dynamic Programming. Wiley&Sons, INC., Publications (2004)
6. Doan, A., Halevy, A.Y.: Semantic integration research in the database community: A brief survey. AI Magazine, Special Issue on Semantic Integration (2005)

7. Doan, A., Domingos, P., Halevy, A.: Learning to match the schemas of data sources: A multistrategy approach. *Maching Learning* **50**(3) (2003) 279–301
8. Califf, M.E., Mooney, R.J.: Bottom-up relational learning of pattern matching rules for information extraction. *J. Mach. Learn. Res.* **4** (2003) 177–210
9. Young-Lai, M., Tompa, F.W.: Stochastic grammatical inference of text database structure. *Mach. Learn.* **40**(2) (2000) 111–137
10. Chidlovskii, B., Fuselier, J.: Supervised learning for the legacy document conversion. In: *DocEng '04*, ACM Press (2004) 220–228
11. Chidlovskii, B., Fuselier, J.: A probabilistic learning method for xml annotation of documents. In: *IJCAI*. (2005)

Clustering XML Documents by Structural Similarity with PCXSS

Tien Tran, Richi Nayak and Kerry Raymond

*Faculty of Information Technology
Queensland University of Technology
Brisbane, Australia
r.nayak@qut.edu.au*

Abstract. *This paper reports on the results and experiments performed on the INEX 2006 Document Mining Challenge Corporuses with the PCXSS clustering method. The PCXSS method is an incremental clustering method that computes the similarity between a new XML document and existing clusters, instead of computing similarity between each pair of documents. We conducted the clustering task on the INEX and Wikipedia data sets.*

1. Introduction

With the emergence of XML standard, XML documents are widely accepted by many industries such as business, education, entertainment and government [1]. With the continuous growth of XML data, many issues concerning with the management of large XML data sources have also arisen. For efficient data management and retrieval, a possible solution is to group XML documents based on their structure and content. The clustering of XML documents facilitates in a number of applications such as improved information retrieval, document classification analysis, structure summary, improved query processing [2, 3] and so on.

The clustering process categorizes the XML data based on a similarity measure without the prior knowledge on the taxonomy [4]. Clustering techniques have frequently been used to group similar database objects and text data. However, clustering of XML documents is more challenging because a XML document has a hierarchical structure and there exist relationships between element objects at various levels.

We propose to use the PCXSS algorithm that is developed to deal with XML schemas to cluster the INEX 2006 Document Mining Challenge Corporuses [5]. Our philosophy is based on the fact that the XML is mainly used to represent instances in semi-structural format. The clustering algorithm should group the documents that share a similar structure. For example, the documents from the publication domain would have different structure from the documents from the movie domain. The structure of the documents play prominent role in grouping similar documents [6]. The

actual instances may not play an important role. The inclusion of instances (the contents within the tag) rather incurs an additional computing cost.

The PCXSS (*P*rogressively *C*lustering *X*ML by *S*tructural *S*imilarity) algorithm employs a global criterion function *CPSim* (common path coefficient) that measures the similarity between an XML data and existing clusters of XML data, instead of computing the pair-wise similarity between two data objects. The PCXSS, originally developed for the purpose of clustering of heterogeneous XML schemas, has been modified and applied to cluster the INEX 2006 XML documents by considering only the structure of XML documents.

2. The PCXSS method: overview

Figure 1 illustrates a high level view of the PCXSS method. The pre-processing phase decomposes every XML schema into structured path information called node paths (each path contains the node properties from the root node to the leaf node). The first stage of the clustering phase i.e. ‘Structure Matching’ measures the structural similarity between node paths of a data object and clusters. This stage determines the similarity between two objects according to the nodes they share common in their paths. The output of the structure matching stage is common path coefficients (*CPSim*) between the data object and all existing clusters. The second stage of clustering phase groups the data object into an existing cluster with which it has the maximum *CPSim* or assigns it to a new cluster.

A number of modifications have been made to the PCXSS method in order to experiment with INEX 2006 corpus. Firstly, the pre-processing phase extracts the structure of every XML documents into X_Paths where only the name of the element is considered in this case. Other information such as data type and constraints are ignored. Secondly, the structure matching of the clustering phase measures only the structural similarity between X_Paths of a document and clusters, where semantic of a tag name is not considered. We have shown elsewhere that semantics of a tag name (such as person vs. people) in XML documents do not make any significant contribution while determining similarity between two XML documents/schemas [6].

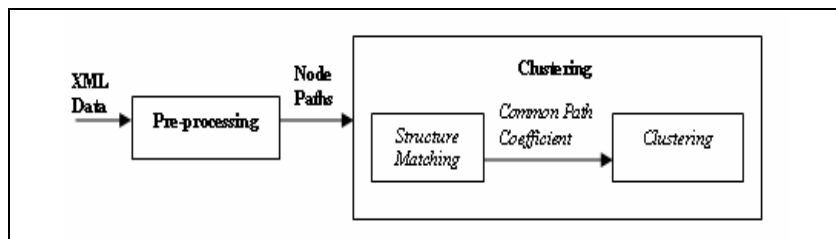


Figure 1. The PCXSS methodology

3. PCXSS Phase 1: Pre-processing

The entire documents in the INEX collection or in the Wikipedia collection conform to only one DTD schema. As a result, we do not require to perform pre-processing of tag

names while inferring the structure of the documents. Only a simple pre-processing step has been applied on the XML documents. The XML documents are first parsed and modelled as the labelled tree (Figure 2). The attribute of an element is modelled exactly the same way as its child elements. The trees are then decomposed into X_Paths to represent the structure of these XML documents.

An X_Path is formally defined as an ordered sequence of tag names from a root to a leaf node which includes hierarchical structure. An XML document consists of many X_Path sequences and the order of X_Paths is ignored because each X_Path is considered as an individual item in an XML document structure. Moreover, duplicated X_Paths in a document structure are eliminated. Duplicated paths are redundant information for structure's presentation as the occurrence of elements is not important for clustering in most cases. After the pre-processing of XML documents, it is presented as a collection of distinct X_Paths.

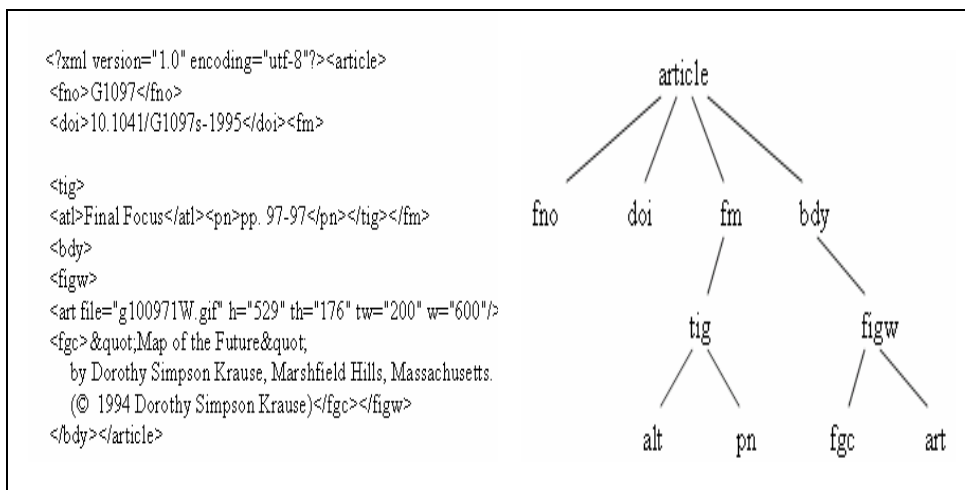


Figure 2. An XML Document (article) & its tree representation

4. PCXSS Phase 2: Clustering

The clustering phase consists of two stages: structure matching and clustering. At structure matching stage, it measures the similarity between a XML document and existing clusters. This stage generates a similarity value called *CPSim* (Common Path Similarity) between an XML document and a cluster. *CPSim* is then used in the clustering stage to group the XML document into an existing cluster with which it has the maximum *CPSim*, or assigns it to a new cluster if the clustering number has not yet exceeded and *CPSim* does not exceed the clustering threshold.

4.1 Structure Matching Stage

Each node in a node path of a document is matched with the node in a node path of the clusters, and then aggregated to form the node path (or structure) similarity.

4.1.1 Node matching. The node matching measures the similarity between the nodes in node paths by considering name similarity ($Nsim$), data type similarity ($Tsim$) and constraints similarity ($Csim$). A weight is assigned by the user to each of the similarities computed. For example, the similarity value for the element's name is assigned with the highest weight since the name is one of the crucial information in determining the node similarity. Below is the formula to compute node similarity ($NodeSim$).

$$NodeSim = w_1 * Nsim(name_1, name_2) + w_2 * Tsim(type_1, type_2) + w_3 * Csim(min Occur, max Occur)$$

The above formula of $NodeSim$ is applied when performing the clustering of XML schemas. For the clustering of XML documents, the $NodeSim$ is calculated by measuring the similarity between the elements in X_Paths by considering only the name similarity ($Nsim$), where a weight of 1 is assigned. The data type similarity ($Tsim$) and constraints similarity ($Csim$) are ignored. The INEX 2006 documents follow the same schema, neither semantic nor syntactic similarity computation is needed on the element name matching. Consequently, the $NodeSim$ value between element names is equal to 1 if they have an identical name else it is assigned with a 0.

4.1.2 Structure similarity. The frequency of common nodes appearing in two XML structures is not sufficient to measure the similarity of XML data. XML is different from other web documents such as HTML or text because it contains the hierarchical structure and relationship between elements. The order of where the element resides in the structure is important in determining the structural similarity between the XML trees and existing clusters.

The structural similarity between two XML schemas is measured by first finding the common nodes between two paths and then finding the common paths between two trees. The structure matching process in PCXSS is advanced by starting at leaf node between two paths to detect more similar elements within structures.

Common nodes finding. The degree of similarity between two node paths, defined as path similarity coefficient ($Psim$), is measured by considering the common nodes coefficient (CNC) between two paths. The CNC is the sum of $NodeSim$ of the nodes between two paths P_1 and P_2 as shown in Figure 6. $Psim$ of paths, P_1 and P_2 is the maximum similarity of the two CNC functions (P_1 to P_2 and P_2 to P_1) with respect to the maximum number of node in both paths, P_1 and P_2 , defined as:

$$Psim(P_1, P_2) = \frac{Max(CNC(P_1, P_2), CNC(P_2, P_1))}{Max(|P_1|, |P_2|)}$$

```

Function:  $CNC (P_1, P_2)$ 
Sim:= 0; for each  $n_i \in P_1$ 
  while j not end of  $P_2$  length
    if(NodeSim( $n_i, n_j$ )) > threshold (defined by user)
      Sim += NodeSim( $n_i, n_j$ )
      j--
      break from 'While' loop
    else
      j--
    end if
  end while
end for
return Sim

```

Figure 6. The CNC algorithm

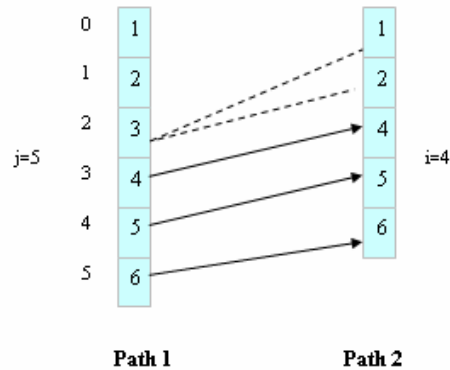


Figure 7. Example of CNC matching

Figure 7 shows an example of traversing through the *CNC* function. The Path1 1/2/3/4/5/6 contains 6 element names showed as numbers for convenience. The Path 2 1/2/4/5/6 contains 5 elements. The following steps are iterated when calculating the *CNC* function:

1. Start at the leaf element of both paths ($j=5, i=4$). If the NodeSim coefficient of the leaf elements exceeds a threshold (a match) then increase Sim (Figure 5) with the NodeSim value and go to step 2 else go to step 3.
2. Move both paths to the next level ($j--, i--$) and start element matching at this level. If the NodeSim coefficient of these elements exceeds a threshold (a match) then increase Sim with the NodeSim value and repeat step 2 else go to step 3.
3. Move only path 1 to the next level ($j--$) then start element matching in the original level of path 2 (i) to the new element of path 1.

It is important to note that $CNC(P_1, P_2)$ is not equal to $CNC(P_2, P_1)$. If the leaf element from P_1 can not be found in P_2 then no further matching requires. In some cases, one path may be a sub path of the other. If P_2 is a sub path of P_1 , and if the leaf element can not be found in P_2 then the $CNC(P_1, P_2)$ returns 0 however $CNC(P_2, P_1)$ will return a value according to the matching. Thus, both $CNC(P_1, P_2)$ and $CNC(P_2, P_1)$ are computed and the maximum of the two is used to measure the degree of similarity between the two paths.

The $Psim$ value is monitored by a path similarity threshold. The threshold determines whether the two node paths are similar. If the $Psim$ of two node paths exceeds the path similarity threshold then it is used to determine the structural similarity between the trees and existing clusters.

Common paths finding. PCXSS measures common paths (1) between two trees and (2) between a tree and a cluster.

Tree to Tree Matching: The tree to tree matching is the matching between a new tree and a cluster that contains only one tree. This is defined as:

$$CPSim(Tree_1, Tree_2) = \frac{\sum_{i=1}^{|TPath_1|} \max(\int_{j=1}^{|TPath_2|} Psim(P_i, P_j))}{Max(|TPath_1|, |TPath_2|)}$$

where $CPSim$ is the common path similarity between two XML trees. The $CPSim$ of trees, $Tree_1$ and $Tree_2$ is the sum of the best path similar coefficient ($Psim$) of paths, P_i and P_j with respect to the maximum number of paths, $|TPath_1|$ and $|TPath_2|$ of trees, $Tree_1$ and $Tree_2$, respectively. The clustering process in PCXSS works on the assumption that only one path from $Tree_1$ matches with one path in $Tree_2$. Thus, it only selects the maximum $Psim$ between each pair of paths of $Tree_1$ and $Tree_2$.

Tree to Cluster Matching: The tree to cluster matching is the matching between a new tree and the common paths in a cluster. The common paths are the similar paths that are shared among the trees within the cluster (normally a cluster must contain at least 2 or more trees in the cluster to have the common paths or else the tree to tree matching is required). Initially, the common paths are derived in the tree to tree matching. Then every time a new tree is assigned to the cluster, the similar paths are added to the cluster if paths are not already in the cluster. The tree to cluster matching is defined as:

$$CPSim(Tree, Cluster) = \frac{\sum_{i=1}^{|TPath|} \max(\int_{j=1}^{|CPath|} Psim(P_i, P_j))}{Max(|TPath|)}$$

Similar to the tree to tree matching, $CPSim$ between a tree and a cluster is the sum of the best $Psim$ of paths, P_i and P_j w. r. t. the number of paths, $|TPath|$ in the *Tree*.

4.2 Clustering Stage

PCXSS is an incremental clustering method. It first starts off with no cluster. When a new tree comes in, it is assigned to a new cluster. When the next tree comes in, *CPSim* is computed between the tree and the existing cluster. If *CPSim* exceeds the clustering threshold and the cluster has the largest *CPSim* with the tree then the tree is assigned to that cluster else it is assigned to a new cluster. The node paths of the tree that are used to compute the *CPSim* are then added to the cluster. The node paths in the cluster are referred to as common paths. The common paths in the cluster are then used to measure the *CPSim* between the cluster and new trees. Since the common paths (instead of all the node paths of the trees held within a cluster) are used to compute *CPSim* with new trees, the computation time reduces significantly. In addition, the cluster contains only the distinct common paths (duplicate paths are removed from the cluster). Figure 8 shows the algorithm for the clustering process in more detail.

-
1. assign the first tree T_1 to a new cluster C_1
 2. **while** tree file has more
 3. read the next tree (i.e. a set of node paths, denoted by T_i);
 4. **while** cluster C has more
 5. **if** C_j contains only one tree, T_j
 6. compute $\text{sim} = \text{CPSim}(T_i, T_j)$;
 7. **else** compute $\text{sim} = \text{CPSim}(T_i, C_j)$ **end if**
 8. **end while**
 9. **if** $\text{Max}(\text{sim}) \geq \text{clustering threshold}$
 10. assign T_i to C_j ;
 11. add node paths to C_j ;
 12. **else** assign T_i to new cluster **end if**
 13. **end while**
-

Figure 8. PCXSS clustering process

5. Experiment and Discussion

Test data. The data used in the experiments are the INEX corpus and Wikipedia corpus from the INEX XML Mining Challenge 2006. Table 1 shows the properties of the experimental corpus.

Test Data	No. of Classes	No. of XML documents	Size (MB)
INEX	18	6054	259
Wikipedia	60	75047	530

Table 1. Test Data Sets

Evaluation methods. For INEX XML Mining Challenge 2006, the clustering solutions are measured using f1-measures: micro-average f1 and macro-average f1. These measures are used to evaluate multi-labeled classification (more than 2 labels). To understand how micro-average f1 and macro-average f1 are measured, it is necessary to revisit the precision, recall and f1-measure which are widely used in

information retrieval. For example, for binary classification, the precision (p), recall (r) and f1-measure are defined as below:

$$p = A / A + B \quad r = A / A + C \quad f1 = 2pr / p + r$$

Where A stands for the number of positive samples which are predicted as positive, B stands for the number of negative samples which are predicted as positive, and C stands for the number of positive samples which are predicted as negative. In a multi-label classification, summing up A,B,C over all binary classification respectively and then f1 is calculated based on them is called micro-average f1 measure. The macro-average f1 is determined when the f1 value is averaged over all binary classification. Micro and macro f1 measures are applied directly on multi-label classification solutions for evaluation. Refer to paper [7], for more information on f1 measure for multi-label classification. However, to measure the clustering solutions, the clustering solutions are first converted to classification solutions before calculating the micro and macro f1 measures.

Experiments and Results. We submitted the 3 results for INEX test data and 1 result for Wikipedia test data to the INEX XML Mining track 2006. The varied submissions were made due to results obtained by setting different thresholds during experiments. The results of the clustering solutions performed by PCXSS are shown in Table 2.

Clustering Threshold	Categories Discovery	Micro F1	Macro F1
0.5 (INEX)	7	0.072944	0.039460
0.7 (INEX)	6	0.088004	0.044307
0.8 (INEX)	7	0.088824	0.044641
0.3 (Wikipedia)	20	0.120460	0.037375

Table 2. Results from INEX XML Mining track 2006

The F1 measure of the clustering solutions obtained with PCXSS on INEX and Wikipedia test data are low. We examined the results and our experimental setups to find out why the clustering solutions have low performance. Firstly, we used the different thresholds to see whether does the threshold value makes any difference in finding the clusters. The results do not seem to improve much.

Secondly, we eliminate the attributes of the element to see whether it can improve the clustering solutions. The results in Table 3 show that the removals of the attributes of the elements can improve the clustering results a little using the same thresholds. However, the results are not yet satisfactory. The reason of the improvement may be that the attributes contained by the Wikipedia and INEX corpuses do not play an importance in understanding the structure of the XML document itself.

Clustering Threshold	Categories Discovery	Micro F1	Macro F1
0.5 (INEX)	7	0.149186	0.090254
0.7 (INEX)	10	0.150553	0.096187

0.8 (INEX)	10	0.150553	0.096187
------------	----	----------	----------

Table 3. Clustering solution without the attributes

The clustering solution using clustering threshold of 0.8 in table 2 is further analysed. This clustering solution has discovered 7 out of 18 true categories. Table 4 below shows the mapping between the 18 clusters that have been generated by PCXSS to its true category. It shows that the documents in category 3 are widely spread out over the 18 clusters that have been discovered by PCXSS. This may happen due to many reasons. Firstly the XML documents from same category (in this case 3) are not grouped together into one cluster by PCXSS due to the difference in structure and size. The PCXSS algorithm mainly derives the solution based on structure similarity. Moreover, the instances play a significant role in measuring the similarity between INEX documents when documents conform to only one document. We have ignored the instances in our experiments.

18 Clusters Discover by PCXSS	True Category
11	11
10	3
13	17
12	13
15	3
14	3
17	5
16	3
18	14
1	3
3	13
2	3
5	3
4	3
7	3
6	12
9	3
8	5

Table 3. Mapping of 18 Clusters Discover by PCXSS to its True Category

To achieve some success, we tried another modification to the clustering algorithm itself. The principle is to increase the time performance while maintaining the accuracy. Since the accuracy obtained is not very high, we decided to measure the similarity between a XML document with the first tree in the cluster without using common paths. We only consider the first tree that formed the cluster instead of comparing with all the common paths (of all trees) that are included in the cluster. The results of the INEX corpus are shown in Table 5.

The clustering solutions achieve little better results than the Table 3. It shows that the clustering on common paths on these kinds of data may not be sufficient enough without the instance. Moreover, there was significant time saving to reach to

the clustering solutions that compared a new tree with only one tree in the cluster. This area will further be analysed in future work.

Clustering Threshold	Categories Discovery	Micro F1	Macro F1
0.8 (INEX)	9	0.179525	0.115392
0.9 (INEX)	9	0.174740	0.118604
0.3 (INEX)	6	0.103753	0.051152
0.4 (INEX)	7	0.126618	0.086362
0.4 (Wikipedia)	18	0.121828	0.050716
0.7 (Wikipedia)	10	0.125178	0.033793
0.6 (Wikipedia)	13	0.126537	0.034368

Table 3. Results from the modification of the clustering algorithm in PCXSS

Based on these experiments, it can be ascertained that the measuring the structure similarity in the documents derived from the same schema do not show any advantage. The usual methods of matrix computations considering only the instances of documents such as vector space, neural networks may have been appropriate here.

The structure overlapping in the documents of the corpus due to the belonging in the same schema and the difference in the sizes and structures within the documents from the same category also play the negative role in clustering.

6. Conclusions and future work

This paper presented the experience of applying the PCXSS clustering method considering only the structure of the XML document to cluster the data of the INEX 2006 document mining challenge. Our aim was to explore whether the structure of the XML documents overplay the instances (contents within tags) of the documents for the clustering task. The experiments show that the structure matching employed by PCXSS alone can not be applied well on the INEX documents especially when the XML documents conform to only one schema.

The development of the PCXSS clustering algorithm originally meant to cluster the heterogeneous schemas. To use the PCXSS on XML documents may need a number of extensions such as the learning of instance and data type for a more efficient clustering solution.

7. References

1. Bray, T., et al., *Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation*. 2004.

2. Boukottaya, A. and C. Vanoirbeek. *Schema matching for transforming structured documents*. in *2005 ACM symposium on Document engineering*. November 02-04, 2005. Bristol, United Kingdom.
3. Nayak, R., R. Witt, and A. Tonev. *Data Mining and XML documents*. in *The 2002 International Workshop on the Web and Database (WebDB 2002)*. June 24-27 2002.
4. Han, J. and M. Kamber, *Data Mining: Concepts and Techniques*. 2001, San Diego, USA: Morgan Kaufmann.
5. Nayak, R. and T. Tran, *A Progressive Clustering Algorithm to Group the XML Data by Structural and Semantic Similarity*. To be published in "International Journal of Pattern Recognition and Artificial Intelligence" (Data of Acceptance: 9th Oct), 2006.
6. Nayak, R. *Investigating Semantic Measures in XML Clustering*. in *The 2006 IEEE/ACM International Conference on Web Intelligence*. Dec, 2006. Hong Kong.
7. Luo, X. and N. Zincir-Heywood. *Evaluation of two systems on multi-class multi-label document classification*. in *ISMIS05*. 2005. New York, USA.

Classifying XML Documents Based on Structure/Content Similarity

Guangming Xing, Zhonghang Xia

Department of Computer Science, Western Kentucky University, Bowling Green, KY
42104, guangming.xing@wku.edu

Abstract. In this paper, we present a framework for classifying XML documents based on structure/content similarity between XML documents. Firstly, an algorithm is proposed for computing the edit distance between an ordered labeled tree and a regular hedge grammar. The new edit distance gives a more precise measure for structural similarity than existing distance metrics in the literature. Secondly, we study schema extraction from XML documents, and an effective solution based on minimum length description (MLD) principle is given. Our schema extraction method allows trade off between schema simplicity and preciseness based on the user's specification. Thirdly, classification XML documents based on the edit distance is discussed. The efficacy and efficiency of our methodology have been tested using the data sets from XML Mining Challenge.

1 Motivation and Literature Review

The widely use of XML in different business applications results in large volume of heterogeneous data: XML documents conforming to different schemata. An XML document is defined by markup tags from a *Document Type Definition (DTD)*, forming a tree structure. Classifying XML documents based on the tree structure and the content is an important problem and is crucial for XML document storage and retrieval [11].

Various methods [10, 4] have been proposed and implemented for XML document classification, and most of them use tree edit distance as a measure of similarity. Tree edit distance [9, 6] is defined as the cost of the sequence of edit operations to transform one tree to another. It offers a very precise measure for document similarity between two documents. However, tree edit distance is not a good measure for structural similarity, as edit distance between two documents can be large (one large document and one small document) while they have very similar structure (conform to the same schema). To overcome this problem, various methods have been proposed. For example, Jagadish [10] proposed a method using *graft* and *prune* to improve the efficiency of computing edit distance and accuracy of classification. More recently, Dalamagas [4] studied XML document classification/clustering using tree summaries and top-down tree edit distance. Both methods offers very high classification/clustering accuracy when the set of documents conforms to the DTDs whose length of the repeat patterns is 1.

However, the performance of these two methods get significantly degraded when the underlying DTDs have more complicated patterns. Although the tree summary method significantly reduces the time complexity for computing the tree edit distance, the structures of the trees may not be preserved by the structural summaries. For example, consider the example in Fig. 1: the two trees on the left side have different structures, but they share the same structural summary based on the methods in [4].

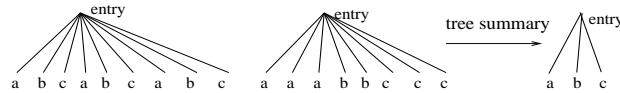


Fig. 1. Trees of Different Structure with the Same Structural Summary

As the structure of an XML document is defined by a schema, it is natural to study the distance between XML documents and schemata and use it as a similarity measure for document classification.

The remainder of this paper is organized as follows. Tree representation of an XML document, normalized regular hedge grammar (NRHG), tree edit operations, and the algorithm to compute the edit distance between a tree and a NRHG is presented in Section 2. The algorithm to find a schema that can generate a set of XML documents is covered in Section 3. Section 4 covers the use of the edit distance between an XML document and a schema in classifying XML documents. The implementation and experimental studies are presented and discussed in section 5, and the conclusion remarks are given in section 6.

2 XML Documents, Schemata and Edit Distances

An XML document can be represented as a node labeled ordered tree. *Ordered* means the order among the siblings is significant, while *labeled* means that each node in the tree is labeled with a symbol from a predefined alphabet.

Document Type Definition (DTD) has been widely used to specify the schemata of XML documents. An XML document conforms to a DTD if it can be generated by the DTD. A DTD can also be viewed as a tree, with the edges labeled with the cardinality of the elements. But a DTD may be recursive, some nodes may lead to a infinite path (it is a DAG instead of a tree in this case). Therefore, instead of working on a DTD directly, we convert it to a normalized regular hedge grammar (NRHG) [2], which can be defined as follows:

Definition 1. A NRHG is a 5-tuple (Σ, V_T, V_F, P, s) , where:

1. Σ is a finite set of terminals,
2. V_T is a finite set of tree variables,
3. V_F is a finite set of forest variables,

4. P is a set of production rules, each of which takes one of the four forms:
 - (a) $v_t \rightarrow x$, where v_t is a tree variable in V_T , and x is a terminal in Σ .
 - (b) $v_t \rightarrow a\langle v_f \rangle$, where v_t is a tree variable in V_T , a is a terminal in Σ and v_f is a forest variable in V_F .
 - (c) $v_f \rightarrow v_t$, where v_f is a forest variable and v_t is a tree variable.
 - (d) $v_f \rightarrow v_t v'_f$, where v_f and v'_f are forest variables and v_t is a tree variable.
5. $s \in V_T$ is the starting symbol, which defines the tree pattern that can be generated by this grammar.

In the above definition, the terminals are used to label the nodes (both leaf and internal) in a tree; the tree variables are grammar symbols to generate trees; and the forest variables are used to generate forests (string of tree variables). Rule (a) is used to generate a tree with a single node, rule (b) is used to put a new node as a new root of the forest that is generated by forest variable, rule (c) is the base case to generate a tree for a forest, and rule (d) is used to concatenate a tree with a forest to form a new forest.

An ordered labeled tree is said to conform to a NRHG if it can be generated by the grammar. When a tree doesn't conform to a NRHG, it can be transformed by a sequence of edit operations such that the result tree conforms to the NRHG. In this paper, we use the same types editing operations for ordered labeled forests as described in [6]: (1) insert as a leaf, (2) delete a leaf, and (3) replace. A cost is assigned to each of these operations. The *edit distance between a tree and a NRHG* is the minimum cost of a sequence of edit operations transforming the tree to conform to the grammar.

2.1 Matching Algorithms

In this section, we present the recursion to calculate the distance between an ordered labeled tree and a NRHG.

Notations:

To identify the nodes in a tree, the nodes are numbered based on post-order traversal. Given a tree T , and an integer i :

- $t[i]$ represents the node of T whose post-order is i ;
- $t[i]$ refers to the label of the node $t[i]$ when there is no confusion;
- $T[i]$ represents the sub-tree rooted at node $t[i]$;
- $F[i]$ represents the sub-forest obtained by deleting $t[i]$ from the tree $T[i]$;
- $p(i)$ refers to the order of the parent node of $t[i]$;
- $n(i)$ refers to the order of the right sibling of $t[i]$;
- $F_s[i]$ denotes the suffix-forest obtained by deleting the left sibling(s) of $t[i]$ from $F[p(i)]$.

$\delta(T_t, T_s)$: is the minimum cost to transform T_s to T_t ;

$\delta(F_t, F_s)$: is the minimum cost to transform the source forest F_s to the target forest F_t ;

For $v_t \in V_T$ in a NRHG, and a tree t , define:

$$C[v_t, T[i]] = \min\{\delta(t, T[i]) : v_t \rightarrow^* t\}.$$

Similarly, for $v_f \in V_F$ in a NRHG, and a forest f , define:

$$C[v_f, F[i]] = \min\{\delta(f, F[i]) : v_f \rightarrow^* f\}.$$

$C[v_t, T[i]]$ is the minimum cost to transform $T[i]$ such that it can be generated by v_t , and $C[v_f, F_s[i]]$ is the minimum cost to transform $F_s[i]$ such that it can be generated by v_f . $C[v_t, T[i]]$ and $C[v_f, F_s[i]]$ can be computed using the following recursions.

Theorem 1. For each $v_t \in V_T$, and each sub-tree $T[i]$:

$$C[v_t, T[i]] = \min \begin{cases} v_t \rightarrow x & \delta(x, T[i]) & (1) \\ v_t \rightarrow a\langle v_f \rangle & \delta(\lambda, T[i]) + C[v_t, \lambda] & (2) \\ v_t \rightarrow a\langle v_f \rangle & C[v_f, F[i]] + \delta(a, t[i]) & (3) \end{cases}$$

and for each $v_f \in V_F$ and sub-forest $F_s[i] = T[i]F_s[n(i)]$:

$$C[v_f, F_s[i]] = \min \begin{cases} v_f \rightarrow v_t & C[v_t, T[i]] + \delta(\lambda, F_s[n(i)]) & (4) \\ v_f \rightarrow v_t & \delta(\lambda, T[i]) + C[v_f, F_s[n(i)]] & (5) \\ v_f \rightarrow v_t v'_f & C[v_t, T[i]] + C[v'_f, F_s[n(i)]] & (6) \\ v_f \rightarrow v_t v'_f & \delta(\lambda, T[i]) + C[v_f, F_s[n(i)]] & (7) \\ v_f \rightarrow v_t v'_f & C[v_t, \lambda] + C[v'_f, F_s[i]] & (8) \\ v_f \rightarrow v'_f & C[v'_f, F_s[i]] & (9) \end{cases}$$

Due to the space limit, the correctness of the above theorem is omitted from this paper.

The above algorithm can be implemented using straight forward dynamic programming except that $C[v_f, F_s[i]]$ may depend on itself based on rule (7) and (8). Just as argued in [8], the value $C[v_f, F_s[i]]$ may potentially depend on itself. This precludes direct use of dynamic programming. We may use the same modification given in [8] to circumvent this problem.

Firstly, the other three cases that lead to smaller cases of the problem can be computed by the following formula:

$$known[v_f, F_s[i]] = \min \begin{cases} v_f \rightarrow v_t & C[v_t, T[i]] + \delta(\lambda, F_s[n(i)]) & (4) \\ & \delta(\lambda, T[i]) + C[v_f, F_s[n(i)]] & (5) \\ v_f \rightarrow v_t v'_f & C[v_t, T[i]] + C[v'_f, F_s[n(i)]] & (6) \end{cases}$$

Secondly, we can use the following procedure to compute the edit distance between each grammar variable and sub-tree.

- 1: **procedure** *ComputeMatrix*(G, F)
- 2: Input: NRHG G and F that is post-order traversed
- 3: Output: $C[n..F[1.. | T |]]$ matrix
4. // $C_t[|V_T|][n]$: cost matrix holds $C[v_t, T[i]]$
5. // $C_f[|V_F|][n]$: cost matrix holds $C[v_f, F_s[i]]$
6. **for** $i = 1$ **to** $|V_T|$ **do**
7. **for** $j = 1$ **to** n **do**
8. $C_t[i, j] = \infty$
9. **for** $i = 1$ **to** $|V_F|$ **do**
10. **for** $j = 1$ **to** n **do**

```

11.    $C_t[i, j] = \infty$ 
12. for1  $s = 0$  to  $n$  do
13.   for2 all tree  $T[i]$  of size  $s$  do
14.     for3 all  $v_t \in V_T$  do
15.       calculate  $C[v_t, T[i]]$ 
16.   for2 all forest  $F_s[i]$  of size  $s$  do
17.     for3 all  $v_f \in V_F$  do
18.        $C[v_f, F_s[i]] = \text{known}[v_f, F_s[i]]$ 
19.      $H \leftarrow$  heap of  $C[v_f, F_s[i]]$ 
20.     while not  $H.\text{empty}()$ 
21.        $C[v_f, F_s[i]] \leftarrow H.\text{extract\_min}$ 
22.       for each  $v_f \rightarrow v_t v_f$ 
23.          $C[v'_f, F_s[i]] \leftarrow \min \{C[v'_f, F_s[i]], \delta(v_t, \lambda) + C[v'_f, F_s[i]]\}$ 
24.          $H.\text{decrease}(C[v'_f, F_s[i]])$ 
25.     endFor
26.   endWhile

```

For a tree with n nodes and a grammar with p rules, there are $O(n \times p)$ $C[v_t, T[i]]$ to compute, and it takes constant time to compute each $C[v_t, T[i]]$. Similarly, there are $O(n \times p)$ $C[v_f, F_s[i]]$ to compute. For each $F_s[i]$, it takes $p \log p$ time to compute $C[v_f, F_s[i]]$ for all the forest variables. So the above procedure can be completed in $O(n \times p \log p)$ time. In most applications, p tends to be much smaller than the number of nodes in a document. Theoretical analysis shows that our method is more efficient than computing the edit distance between two trees, which will be verified by experimental studies in Section 5.

3 Schema Extraction from XML Documents

Most classification/clustering methods for structured documents rely on pairwise distances. In order to use the edit distance defined above for document classification/clustering, the first task is to extract the underlying schema from the XML documents. The problem can be formulated as: Given a collection of XML documents $\{d_1, d_2, \dots, d_n\}$, find a schema s , such that d_1, d_2, \dots, d_n are document instances that can be generated by schema s .

As the definition of an element in a schema is independent of the definitions of other elements, and only restricts the sequence of subelements (the attributes are omitted in this paper) nested within the element. Therefore, the schema extraction can be simplified as inferring a regular expression R (right linear grammar or nondeterministic finite automata) from a collection of input sequences I with the following restrictions:

- R is concise, i.e., the inferred expression is simple and small in size;
- R is general enough to accept sequences that are similar to those in I ;
- R is precise enough to exclude sequences not similar to those in I .

Inferring regular expressions from a set of strings has been studied in [13, 12]. One novel contribution in Xtract is the introduction of Minimum Length Description (MLD) to rank candidate expressions. In general, the MLD principle states that the best grammar (schema) to infer from a set of data is the one that minimizes the sum of:

1. The length of the grammar L_g , and
2. The length of the data L_d when encoded with the grammar.

The overall goal is to minimize $L = L_g + L_d$.

In our system, we use a similar approach as introduced in Xtract: Candidate regular expressions are extracted based on the analysis of the repeat patterns appearing in the input sequences. The candidate regular expressions are then ranked using MLD principle. We have made the following improvements over Xtract:

1. The frequencies of the children sequences are considered in our system and the system may intentionally pick some victims that are not covered by the inferred regular expressions. The victims are those sequences that appears very infrequently. This feature helps to minimize the negative effects of noise data in classification/clustering.
2. In our system, the relative weight between the definition and description can be dynamically adjusted, which can be used to tune the performance of our system. The overall goal in our system is to minimize $L = \lambda L_g + L_d$. The λ can be used to adjust the preciseness and general of the result.
3. Instead of using a regular expression to determine the cost of encoding, we use the cost of nondeterministic finite automata (NFA) simulation as the cost of encoding. This eliminates the necessity for enumerate multiple sequence partitions to compute the minimum cost for encoding.

It is difficult to represent the encoding of a string with a regular expression. So instead of working on regular expressions, we consider NFA constructed by Thompson's [5] method.

For example, given $\{abcab, abdab, abcabab, abcababab, abdababab\}$ as the set of input sequences, we may have $ab(c|d)(ab)^*$ or $(ab|c|d)^*$ as candidate regular expressions after analyzing the repeat patterns. The corresponding Thompson NFAs can be constructed as illustrated in Fig. 2.

The NFA can be represented by encoding the states and its transitions. So

$$L_g = (S + T) \log S,$$

where S is the number of states and T is the number of transitions.

To compute L_d , we use the cost of NFA simulation as the cost of string encoding. Intuitively, the number of states in each state vector denotes the number of choices for each step, which should be encoded. For example, the state vectors in NFA simulation for NFA_1 and NFA_2 on string $abcabab$ can be illustrated by Table 1.

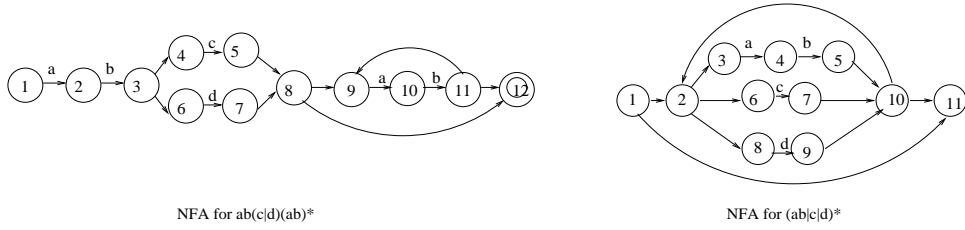


Fig. 2. NFAs for $ab(c|d)(ab)^*$ and $(ab|c|d)^*$

Step	1	2	3	4	5	6	7	8	9	10	11	12	cost	1	2	3	4	5	6	7	8	9	10	11	cost
1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	1	6
2	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1
3	0	0	1	1	0	1	0	0	0	0	0	0	3	0	1	1	0	1	0	0	1	0	1	1	7
4	0	0	0	0	1	0	0	1	1	0	0	1	4	0	1	1	0	0	1	0	1	1	0	1	7
5	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0	1	0	1	1	3	0	1	1	0	1	1	0	1	0	1	1	7
7	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	1	0	1	1	3	0	1	1	0	1	1	0	1	0	1	1	7

Table 1. Transition Vectors for NFA_1 and NFA_2

It is obvious that $ab(c|d)(ab)^*$ is a better choice as the description cost L_d using NFA_1 is much smaller than that of the second one.

The complexity of the above algorithm is $O(cn^2)$, where n is the length of the string for inference, and c is the numbers of the strings. Although the algorithm is quadratic w.r.t. the length of input string, it is highly efficient when the length of the string is short. With heuristic tree size reduction in our implementation, the running time is linear w.r.t. the size of the document for most applications.

4 Measuring the Content Similarity

To measure the similarity of the text contents of the XML documents, we use the standard techniques for text categorization. The vector space model (VSM) is a standard representation approach in document classification. In VSM, a document is represented by the words (terms) it contains. The full set of documents, denoted by D , is referred to as the corpus and the full set of terms, denoted by T , occurring the corpus as the dictionary. There are three phases in the VSM: document indexing, term weighting, and similarity evaluation. Document indexing usually counts frequencies of terms and removes some high frequency terms, such as 'a', 'the', etc. Term weighting procedure measures the contribution of each term to the document and calculates the weight w_j . A document d_i is represented as a vector $d_i = (w_1, w_2, \dots, w_{|T|})$, where $0 \leq w_j \leq 1$ is the weight of the j th term contributing to document d_i , and $|T|$ is the number of terms in

T . Although many words have been removed from T in document classification, the dimensionality of the term space is still very large. Based on the observation that some terms are semantically equivalent, Latent Semantic Indexing (LSI) has been successfully introduced in dimensionality reduction by mapping query and documents into a 'latent' semantic space. In LSI, semantically related terms are mapped onto the same dimensions, and non-related terms onto different dimensions. More specifically, by using Singular Value Decomposition (SVD), a term-by-document matrix $A_{|T| \times |D|}$ is decomposed as

$$A_{|T| \times |D|} = U_{|T| \times r} \Lambda V_{r \times |D|}^T$$

where r is the rank of A and Λ is a diagonal matrix.

In similarity evaluation when a query q is given, we treat it as a document and project it into the LSI space with the transformation as follows.

$$\hat{q} = \Lambda^{-1} U_{r \times |T|}^T q_{|T|}.$$

After the transformation, the similarity between query q and document d_i can be evaluated in the LSI space.

5 Document Classification

In this section, we show how to classify XML documents based on their structures using the edit distance between XML documents and schemas. Classifying documents based on their structures plays an important role for XML data storage and information retrieval [11].

The most important quantitative measure for structure oriented classification of XML document is document distance. Most methods in literature are based on distances between documents, especially tree edit distances. However, tree edit distances between two XML documents may not be a good measure for structural similarity. For two documents of different sizes (different number of nodes), the distance between them may be large even if they conform to the same schema. Based on this observation, it would be ideal to use a measure how good an XML document conforms to a schema. The edit distances between XML documents and schemas will be ideal for this purpose.

The design of the classification system can be illustrated by Figure 3:

According to our approach, there are three steps to get a classifier. The first step is representative selection and schema extraction. The schema extraction has been covered in detail in Section 3.

The second step is to compute the distance between the documents and the schemas (one from each class). Suppose there are n classes of objects in the training set, one unified schema is generated for each class. For each document, a distance vector $\langle d_1, d_2, \dots, d_n \rangle$, which represents the distance between each document and the "center" points of the class, is computed and fed into a learning machine to train a classifier.

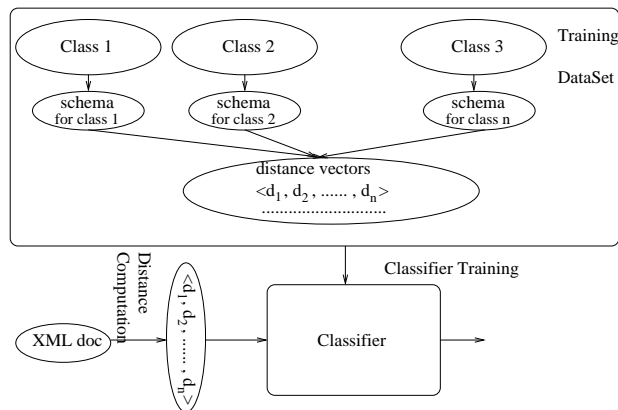


Fig. 3. System Architecture

The third step is the classifier training. Various software packages have been implemented for data classification. Once each document is represented by a distance vector, feeding the distance vector and training the classifier is straightforward for most software systems.

To classify a document, the distance vector consists of the distances between the document and the schema from each class is computed. The classifier gives the label of the document based on the distance vector.

We also tested the classification system on the benchmark data from XML Mining Challenge [11] in order to:

1. Show that the distance between an XML document and a schema is an effective similarity measure for XML documents.
2. Assess the feasibility of automatic document classification based on this new distance.

An SVM classifier is trained using Weka [14] software package.

6 Implementation and Experimental Results

We have fully implemented the algorithms described in the above sections, and developed a prototype system for document clustering. The system is a java-based software system with the following functionalities:

- Extract a schema from a collection of XML documents. The schema can be represented in DTD or NRHG.
- Generate content summaries for a collection of XML documents.
- Compute pairwise distance between XML documents and schemata.
- Compute pairwise distance between XML documents using structural summaries.

- Train an SVM classifier and classification using SVM.

Based on the implementation, we have tested the classification system using the Inex data set provided by XML Mining Challenge [11]:

1. Show that the distance between an XML document and a schema is an effective similarity measure for XML documents.
2. Assess the feasibility of XML document classification based on this new distance metric.

The classification result for MovieDB data set based on the structure is presented in Table 2.

Cluster No	w/o summaries			w summaries			tree grammar		
	a	b	c	a	b	c	a	b	c
1	8	12	12	14	4	6	20	3	0
2	7	14	13	15	7	5	20	0	3
3	10	9	10	14	6	6	20	0	0
P & R	$P = 41.6\%$			$P = 71.6\%$			$P = 95\%$		

Table 2. Classification Result: MovieDB Data Based on Structure

Notice that in our algorithm, the values of P reach excellent level (better than 95%) for the MovieDB data when only the structure information is used for classification. The structural summary method can produce very good results when the length of the repeat pattern is 1, but the accuracy becomes significantly degraded (to 71.6%) when the repeat patterns are more complicated.

The classification result for the Inex data set based on the structure and the content is presented in Table 3.

Remarks:

The evaluation results indicate the following:

- Regular hedge grammar is a better way to characterize the structural properties of XML documents than structural summary.
- The schema extraction method using MLD principle and NFA simulation cost is effective in extracting schema rules from a collection of documents.

7 Conclusions

In this paper, we have studied the problem of computing the edit distance between an XML document and a schema. Three edit operations were considered for manipulating an XML tree: insert a node as a leaf, delete a leaf node and replace, and each operation is of constant cost. We gave a novel solution to this problem by studying how an ordered labeled tree could be transformed such that it conforms to a NRHG with minimum cost (edit distance).

Group	1	10	11	12	13	14	15	16	17	18	2	3	4	5	6	7	8	9	Precision
1	85	0	1	2	1	0	0	0	0	1	0	1	0	0	2	1	0	2	0.88
10	6	113	2	16	0	1	0	1	0	1	39	35	4	5	8	4	0	5	0.47
11	3	10	81	6	2	1	2	0	0	0	4	52	16	11	34	18	3	10	0.32
12	13	24	7	272	1	0	0	0	0	1	24	85	15	4	17	4	3	16	0.55
13	1	0	1	0	284	46	1	6	8	7	1	0	0	0	1	2	0	0	0.79
14	0	0	0	0	82	242	2	7	3	18	0	0	0	0	0	0	0	0	0.68
15	0	0	0	0	1	5	64	3	8	0	1	0	0	0	0	0	0	0	0.78
16	0	0	1	0	32	16	10	200	34	21	1	0	0	0	0	0	0	1	0.63
17	0	0	0	0	13	4	7	10	372	9	0	0	0	0	0	0	0	1	0.89
18	2	0	2	3	91	73	26	60	48	235	5	8	2	2	3	1	3	0	0.41
2	1	23	1	8	0	1	1	0	1	0	134	15	9	3	5	2	0	2	0.65
3	1	15	7	37	1	0	0	0	0	1	14	475	22	6	13	9	1	13	0.77
4	17	26	26	27	5	3	4	2	6	8	64	39	158	19	22	12	3	21	0.34
5	3	3	1	9	2	2	1	2	0	0	11	23	1	164	2	1	1	21	0.66
6	12	7	37	35	0	2	1	0	2	2	10	74	37	10	210	23	3	22	0.43
7	3	3	15	9	0	0	0	0	0	0	0	11	4	2	12	181	4	7	0.72
8	3	1	2	15	0	0	0	0	0	0	19	59	11	13	6	3	112	8	0.44
9	6	10	8	21	1	0	1	0	0	1	18	86	7	34	16	5	0	155	0.42
Recall	.54	.48	.42	.59	.55	.61	.53	.69	.77	.77	.39	.49	.55	.60	.60	.68	.84	.54	

Table 3. Classification Result: Inex Data Based on Structure and Content

Based on the definition of the edit distance between an XML document and a schema, we presented an approach for classification of XML documents using *structural distance*. Although it is more complicated than the methods presented in [10] and [4], it can classify documents having more complicated structure with much higher accuracy. Both classifying based on the structure, and a combination of structure and content are studied in our project. Experimental studies have shown the effectiveness of our methods.

References

1. Nobutaka Suzuki, *Finding an Optimum Edit Script between an XML Document and a DTD*, Proceedings of ACM Symposium on Applied Computing, pp. 647 - 653, March, 2005, Santa Fe, NM.
2. G. Xing, *Fast Approximate Matching Between XML Documents and Schemata*, APWeb 2006 (X. Zhou et al. Eds), LNCS 3841, pp. 425-436, Springer-Verlag, 2006.
3. Rodney Canfield, Guangming Xing, *Approximate XML Document Matching (Poster)*, Proceedings of ACM Symposium on Applied Computing, March, 2005, Santa Fe, NM.
4. Theodore Dalamagas, Tao Cheng, Klaas-Jan Winkel, Timos K. Sellis *A methodology for clustering XML documents by structure*, Information Systems, 31(3): 187-228 (2006).
5. K. Thompson, *Regular Expression Search Algorithm*, Communications of ACM, vol 11-6, pp 419-422, 1968.

6. D. Shasha, K. Zhang, *Approximate Tree Pattern Matching*, Chapter 14 Pattern Matching Algorithms (eds. Apostolico, A. and Galil, Z.), Oxford University Press, June 1997.
7. M. Murata *Hedge Automata: A Formal Model for XML Schemata* http://www.xml.gr.jp/relax/hedge_nice.html
8. G. Myers *Approximately Matching Context Free Languages*, Information Processing Letters, 54, 2, pp. 85-92, 1995.
9. W. Chen, *New Algorithm for Ordered Tree-to-Tree Correction Problem*, J. of Algorithms, 40:135-158, 2001.
10. A. Nierman, H. V. Jagadish, *Evaluating structural similarity in XML documents*, WebDB 2002, Madison, Wisconsin, June 2002.
11. XML Document Mining Challenge, <http://xmlmining.lip6.fr/>
12. Boris Chidlovskii, *Schema Extraction from XML Data: A Grammatical Inference Approach*, KRDB'01 Workshop, Rome, Italy, September 15, 2001.
13. Minos N. Garofalakis, Aristides Gionis, Rajeev Rastogi, S. Seshadri, Kyuseok Shim, *Xtract: A System for Extracting Document Type Descriptors from XML Documents*, SIGMOD Conference 2000, pp. 165-176, May 16-18, 2000, Dallas, Texas, USA.
14. WEKA Project, <http://www.cs.waikato.ac.nz/ml/weka/>
15. Fabrizio Sebastiani, *Machine learning in automated text categorization*, ACM Computing Surveys, vol 34-1, pp 1-47, 2002.
16. G. Karypis, *CLUTO A clustering toolkit* Technical Report 02017, University of Minnesota, Department of Computer Science, Minneapolis, MN 55455, Aug. 2002.

XML Document Mining using Graph Neural Network

S.L. Yong¹, M. Hagenbuchner¹, A.C. Tsoi², F. Scarselli⁴, M. Gori⁴

¹ University of Wollongong, Wollongong, Australia

² Monash University, Melbourne, Australia

³ University of Siena, Siena, Italy

Abstract. This paper addresses the application of a relatively new machine learning method known as the *Graph Neural Network* to the task of classifying semi-structured documents, like XML documents. A relatively large set of XML formatted documents is being used to investigate the suitability of Graph Neural Networks for such a task. It will be shown that the Graph Neural Network approach is capable and it outperforms any other competitor's approach at an international competition on XML document classification.

1 Introduction

Neural Networks are popular machine learning methods that can be trained on a set of examples. Multi-layer Perceptron networks in particular, and networks based on such architectures, are well studied methods and are the most popularly applied in practice. Such networks are trained in a supervised fashion which means: for every (input) sample, a desired (output) target is given. Networks are generally trained iteratively by adjusting internal network parameters such that for a given input pattern the desired output pattern is achieved. The greatest advantage of such a method is in its ability to generalize over previously unseen data. This means that a neural network, once trained, can produce a suitable response to an input that was not part of the set of training samples. In addition, neural networks have good noise immunity, in the sense that they can provide suitable response even though the input samples are contaminated with noise. As a consequence, neural networks are popularly applied to tasks involving noisy samples. Let us take the task of image recognition as an example. No two photographs of the same object are ever identical. This is due to the analogue nature of traditional cameras (or probabilistic nature of digital cameras), and due to the fact that environmental conditions such as lighting conditions and object aging, are constantly changing. A human observer would have no difficulty in assessing object depicted in photographs that can differ vastly in size, pose or quality. Many computer methods are an exact science which require the hard-coding of information or rules to produce object recognition abilities. Such systems are likely to fail if anything unaccounted should occur. For example, if hard coded data or rules do not account for the possibility for an object to fly, then such applications may be unable to identify an apple as it falls from a tree. Neural networks on the other hand are a generic method that can learn from possibly noisy samples and produce a desired output regardless. As such, a neural network would be able to recognize an apple as an apple regardless to its surroundings or physical condition.

Traditionally, neural networks are trained on constant-length vectorial inputs. One extension, called Recurrent Neural Networks, allows the method to be used in characterizing continuous signals. Such networks are trained on constant sized, shifting sub-sections of a signal through a feedback loop which provides information from the processing of a previous sub-section as an additional input to the network. These ideas were extended further through the introduction of Recursive Neural Networks which are capable of processing directed acyclic graphs by individually processing the (labelled) nodes of a graph rather than the graph as a whole. This is achieved through the notion of *states* in the network, which is analogous to the same concept in physics, characterizes the activation of a node, and by providing the states of neighboring nodes as an additional network inputs. The recursive nature of the training algorithm ensures that the states of any node in the graph is propagated through the entire graph, and thus, the graph as a whole is encoded by such a method. This method of processing graph structured data is different to those used in traditional methods in which any graph structured data, e.g. acyclic tree, is first “squashed” to become a vector first, before feeding it to e.g. a multilayer perceptron for the usual processing. By retaining the graph structured data as long as required, it is argued that the information encoded in the topological nature of the graph is retained [6].

A recent development produced the *Graph Neural Network* (GNN), it is a new kind of neural networks that can process much more general types of graphs such as undirected or cyclic graphs, and graphs that feature labelled links and labelled nodes. This has been a breakthrough development since it has become possible for the first time to process general types of graphs without requiring pre-processing of graph structured information.

Graph structured information can be found everywhere in the real world. Any object is a structural composition of basic elements where the elements can be scaled down as far as a molecular level – which by itself is structured object. Even the most basic type of information can be encoded as a structure by simply representing such information by a graph consisting of a single node. In practice, most information can be represented appropriately as a set of vectors. However, there are numerous instances for which a vectorial representation is insufficient. For example, the World Wide Web is a large graph with web pages serving as nodes, and hyperlinks as edges. It would not be useful to represent the Web in vectorial form as most of the information is encoded in the topological nature of the Web.

The GNN extends the idea emerged from the Recursive Neural Network by assuming each node is represented by a recurrent neural network [8] so as to allow the processing of cyclic graphs. The recurrent neural network will model the recursive nature of the information encoded in a cyclic graph or an undirected graph in the form of a *state*. The outputs from the hidden layer are then fed to the succeeding node (along the link) as inputs. The output is obtained from the output node of the entire document. In this case, this will be an ordinal number denoting the classification of the document. The training at each stage will involve the adaptation of the weights of each node from the local inputs to the node. With some generic conditions on the behaviour of the GNN, it is possible to show that such networks have universal approximator property, and that the states in each node will converge. The detailed theoretical description of the GNN

Table 1. INEX XML document classes in the training dataset. The associated numbers are the class IDs.

	Computer	Graphics	Hardware	Artificial Intelligence	Internet	Parallel
Transaction	tc(13),ts(18)	tg(15)		tp(17),tk(16)		td(14)
Non Transaction	an(1),co(3), cs(4),it(8),so(12)	cg(2)	dt(5),mi(9)	ex(6),mu(10)	ic(7)	pd(11)

method is quite complex and is not within the scope of this paper. The reader is advised to consult the following references: [4, 5, 7].

This paper applies the GNN to the task of classifying XML formatted documents to investigate its suitability for such a task. The task is of particular interest due to the increasing popularity of XML formatted information. Vast repositories of electronic documents are being represented using this structured meta language and hence, the mining of information from such an environment is becoming increasingly important.

It will be shown that the GNN method requires little or no pre-processing of such semi-structured data, that it can perform well, and that it is suited for mining large databases due to its generalization abilities. It will furthermore be demonstrated that the performances obtained by processing either the document structure only or both the structure and content the results are superior to those of any other competitors' approaches obtained to date.

This paper is structured as follows: an overview of the learning task is given in Section 2. A detailed description of the experimental framework, and findings are presented in Section 3. Some conclusions are drawn in Section 4.

2 The Learning Problem

For the INEX XML Mining task, a training dataset is provided. The dataset contains published papers in computer science, in the areas of hardware, graphics, artificial intelligence, etc. It contains papers from transactional and non-transactional journals. The training dataset consists of a total of 6,053 XML documents which are divided into 18 different classes. These 18 classes are shown in Table 1.

Information on which XML document belongs to which class is provided for the training dataset. The distribution of the training dataset is shown in Figure 1. It can be seen that the largest class is "co" (3) with a total of 963 documents and the smallest class is "tg" (15) with a total of 105 documents. In this training dataset, it is important to note that there are two major difficulties:

1. The dataset is unbalanced⁴ for the number of documents in each class; and
2. The dataset is unbalanced between "Transaction" and "Non-Transaction" articles.

There are two classification tasks specified in this competition:

⁴ By "unbalanced" we mean that there are uneven distributions of data.

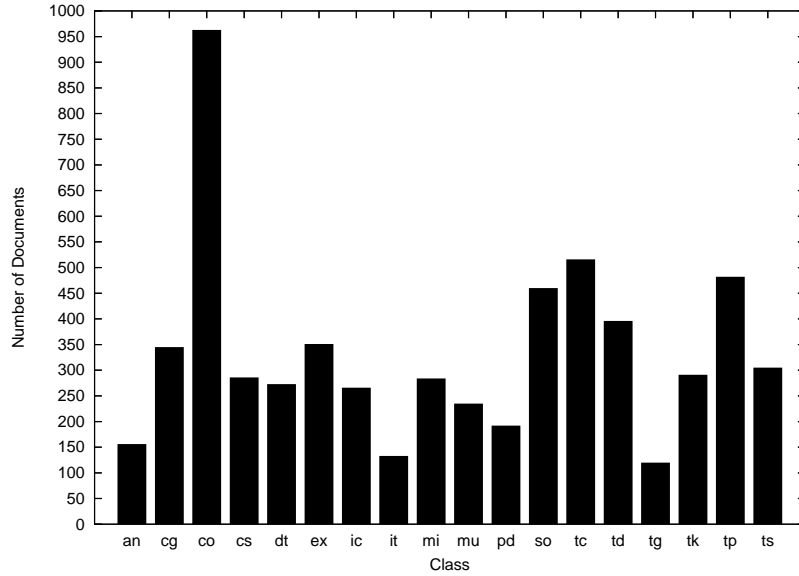


Fig. 1. Distribution of INEX XML Documents in Their Respective Class

Structure only: Classification of documents by considering the XML structure only. Any textual content (other than the XML tags) is ignored.

Structure and content: Classification of XML documents using both structure and content.

For the classification using structure only task, it is noted that there are a total of 165 different XML tags which may or may not occur in all the XML documents. A test dataset was made available towards the end of the XML clustering competition. The test data were unlabeled. Performances on the test dataset were computed by using a script as provided by the organizers of the clustering competition⁵. This situation is analogous to the real world situation where machine learning methods are only trained on the training dataset and deployed on the real data which has not been used to train the algorithms.

For both tasks, the performance is measured using Macro and Micro- F_1 . F-measure[2] is the weighted harmonic mean of precision and recall defined as:

$$F = \begin{cases} 0 & \text{if } \alpha R + (1 - \alpha)P = 0 \\ \frac{PR}{\alpha R + (1 - \alpha)P} & \text{else} \end{cases} \quad (1)$$

where P is the precision, R is the recall, α is a balancing factor, and $F, R, P \in [0; 1]$. In the standard F_1 (or simply refer to as F-measure), α is set to $\frac{1}{2}$ where P and R is weighted equally. Macro- F_1 is the averaged F_1 value over all classes while Micro- F_1 is the weighted average F_1 value. The task is to maximise F .

⁵ The script can be obtained from <http://xmlmining.lip6.fr/Results>

3 Experiments

3.1 Initial experiments

Test data were not available until shortly before the conclusion of the XML clustering competition. As a consequence, the initial approaches addressed in this Section evaluate network performances on the training data. Performance evaluations on test data will be given in Section 3.2. Thus, initially we resorted to splitting the available data (the original training data set) into three sub-sets ⁶:

Training Set: 80% of the original training data set is selected to be used for training purposes.

Validation Set: 10% of the original training data set is selected to serve as a validation data set. Validation data are used to test the network during training but are not used to train the network. The network is trained to perform optimally on the validation dataset.

Test Set: The remaining 10% are used as test data.

Experiments were conducted by applying the multilayer perceptron (MLP) which processes data without considering the topological features, and by applying the GNN to the graph structured representation of the same dataset. Unless stated otherwise, the neural models trained were non-linear models using the linear output function $x = f(x)$ in the output layer. All other neurons used the sigmoidal *hyperbolic tangent* transfer function. Two hidden layers with 10 neurons in each layer were used. In addition, for the GNN, we used 5 state neurons between the transition network and output network.

Processing without topological features: As a baseline performance measure, a standard multilayer perceptron (MLP) is used. This is to validate to whether the inclusion of structural information into the machine learning method produces a noticeable advantage. When using the MLP, the input data needs to “squash” the structured input into a corresponding vector. This is done as illustrated in Figure 2. Thus, the training set consists of 6,053 vectors of dimension 165, each of which is associated with an 18 dimensional binary target vector. Iterating the MLP training algorithm for 1000 iterations required 30 minutes of CPU time on a 3GHz Intel based CPU. The performance of the trained MLP on the test set is as shown in Table 2. It can be seen that the performance measured P and R can vary greatly across the various classes and tend to produce better results for the larger classes.

To counter possible issues that may have arisen from the unbalanced distribution of pattern classes, we then trained the MLP by balancing the training set. This was

⁶ There are other ways of dividing up the training dataset for training purpose, e.g. N-fold cross validation method. In this case, the training dataset is randomly divided into the training dataset, and N validation datasets. The algorithm is trained using the training dataset, evaluated on randomly selected (N-1) validation sets and then tested on the remaining validation set. The result is the average over all the experiments, and the model which provides the best result will be selected. We decided not to use the N-fold cross validation method as it will take too much time due to the complexity of the tasks. This will become clear later in this section.

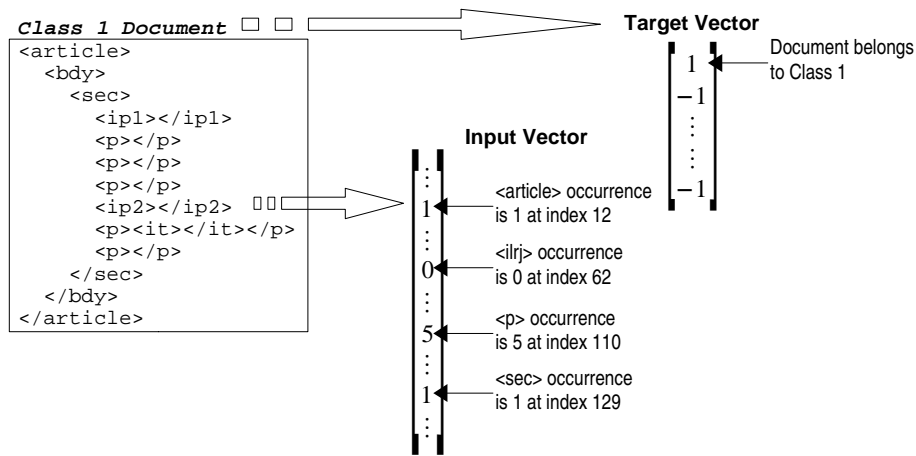


Fig. 2. Conversion of an XML document to a MLP input and target.

Table 2. MLP training results.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
<i>R</i>	0.06	0.11	0.10	0.13	0.22	0.28	0.02	0.04	0.14	0.30	0.17	0.14	0.62	0.41	0.33	0.43	0.16	0.56
<i>P</i>	0.03	0.07	0.03	0.08	0.12	0.11	0.01	0.01	0.09	0.08	0.12	0.07	0.34	0.29	0.31	0.39	0.09	0.27
	Macro F1: 0.17, Micro F1: 0.12																	

Table 3. Balanced MLP training results.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
<i>R</i>	0.08	0.12	0.12	0.15	0.23	0.28	0.04	0.05	0.15	0.35	0.19	0.16	0.62	0.41	0.35	0.43	0.18	0.56
<i>P</i>	0.05	0.09	0.05	0.09	0.12	0.11	0.02	0.02	0.01	0.09	0.12	0.09	0.34	0.29	0.31	0.39	0.10	0.28
	Macro F1: 0.18, Micro F1: 0.12																	

achieved by increasing the samples from the small classes by creating copies of randomly chosen data from the same class. Copies were added until each of the classes was of the same size as the largest class. The result of training the MLP on this dataset is shown in Table 3. It can be seen that the performance increased slightly. The F1 values serve as a benchmark on which to compare alternative approaches.

Processing data by including topological features: Graph Neural Network is a neural network architecture which can process graph structures without pre-processing. However for real world tasks, in general, training GNN without pre-processing is not feasible due to the limitation of computer hardware memory and processing speed.

Without pre-processing, the resulting graphs have a maximum size of 14, 047 nodes. When considering the general schema of the GNN learning process as depicted in Figure 3, it is seen that the present implementation of GNN requires each graph to be

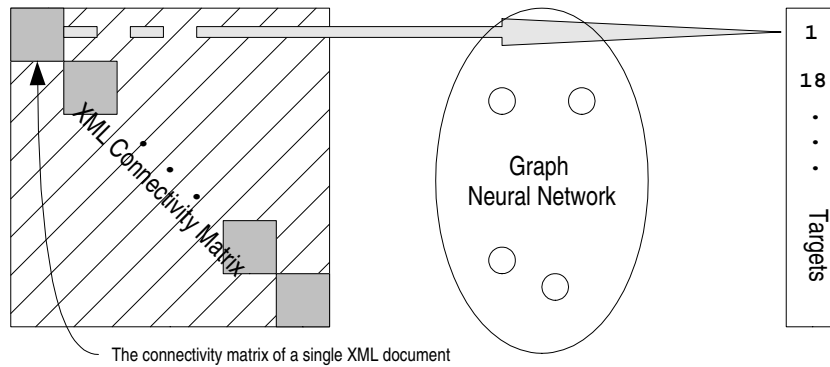


Fig. 3. The Graph Neural Network learning process.

mapped to a fixed size matrix ⁷. In addition, since GNN represents each node by a recurrent neural network, the memory requirements also depend on the state dimension and on the number of hidden nodes in the transition net, which give an important contribution to the memory occupation. Thus, it is found that without pre-processing, for 6,053 XML documents, it will require storage for a total of $14,047 \times 6,053 = 85,026,491$ nodes. To process these graphs directly is not practical, some pre-processing is necessary to reduce the size of the data.

To find a good pre-processing technique, it is important to first understand how GNN is deployed for the XML learning task. In the XML learning task, an XML file is translated into a graph as shown in by an example in Figure 4. In Figure 4 is can be seen that for each XML tag that is encapsulated by another tag this produces one child node in the graph. Since in XML it is not possible to have overlapping tags such as `<it><p></it></p>`, the resulting graph is a tree where links can be directed so as to indicate the parent/child relationship between the nodes, or be undirected if no such relationship is to be assumed.

GNN estimates the target state of a particular node depending on the neighbour nodes iteratively. Referring to Figure 4, for example, the state of the node `<sec>` depends on the nodes: `<bdy>`, `<ip1>`, `<p>` and `<ip2>`. However, if we observe more carefully in Figure 4, the shaded node `<p>` differs from other `<p>` nodes since it has an extra node `<it>` as its neighbour. Due to the extra neighbour, the shaded `<p>` node should have a state different from the unshaded `<p>` nodes. In Figure 4, we note further that there are four `<p>` nodes in the same configuration. These would group together to become one `<p>` node. The shaded `<p>` node is a separate one from the unshaded `<p>` nodes. Thus, in order to decrease the dimension of the training set we consolidate repeated sub-structures. The result is that the graph depicted in Figure 4 is reduced to a graph as shown in Figure 5.

With the pre-processing method, the graph size is reduced to a maximum of 78 nodes per XML document. Each of the node is then given a unique numeric label ac-

⁷ This is a limitation with the current software implementation. No such limitation is imposed by the underlying theory of GNN.

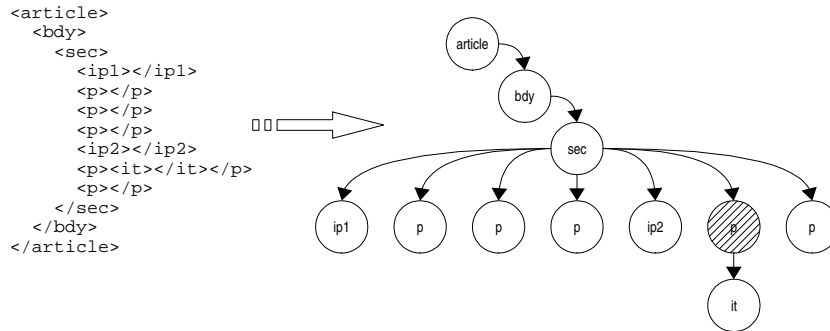


Fig. 4. Conversion of an XML document to a corresponding graph structure.

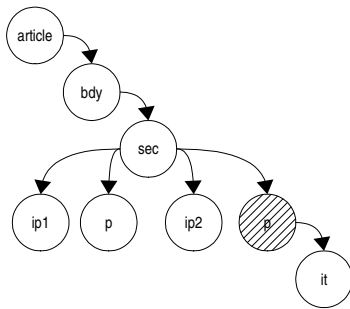


Fig. 5. The reduced XML Graph.

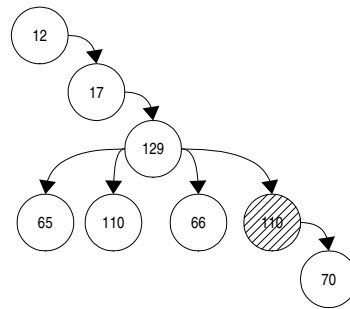


Fig. 6. XML graph with node labels.

according to the XML tag (as shown in Figure 6) to differentiate the nodes. Ideally these node labels should be multi-dimensional (i.e. 165 dimension in the INEX XML since there are 165 different tags) to maximize the Euclidian distance. However, a 165 dimension label is also not practical in this experiment, instead only a 1-dimensional node label is used.

Graph Neural Network: Classification Using Structure Only As a first approach, we trained one GNN for each of the 18 document classes by creating a training set which has a positive target value for patterns that belong to the particular class, and a negative target value for all other patterns. The result is 18 GNNs where each of the GNN has been trained to recognize one class only. In order to produce an overall classification of the test data, we also experimented on a number of ways as follows:

1. The GNN that produced the largest (positive) value at the root node determines the class to which the document should be assigned;
2. Training a MLP to process the output of all 18 GNNs on the root node. Here, the MLP receives an 18-dimensional input (the output of the GNNs) and is trained to produce as output the class membership of the input pattern. Once trained, the GNN-MLP dual system produces the classification of a test data.

Table 4. The results for unbalanced training of GNN’s.

Approach	Micro F1	Macro F1
Highest Output of Root Node	0.13	0.09
MLP Output of Root Node	0.15	0.12
Highest Output of all Nodes	0.04	0.02
MLP Output of all Nodes	0.11	0.05

3. For each of the 18 GNNs, combine the output created for all (78) document nodes then take the highest output to determine the class of a document; this is akin to the idea of a winner take all approach to processing the classification outcome; and
4. Combine the scores of all (78) document nodes and process the output through an MLP similarly to approach 2.

The results are as shown in Table 4. Upon closer investigation of these results, it is found that most of the documents are classified as class “co”. Referring to Figure 1, we observe that the class “co” is the largest class and its size is about 166% of the second largest class “tp” and 894% of the smallest class “tg”. This raises the suspicion that the highly unbalanced nature of the training dataset causes the low performance of the learning task.

Even with such a low performance, the results shown in Table 4 gives some useful insight about using a GNN model:

1. Performance improves with using an MLP output. This can be explained by the fact that the outputs of the GNN may have different magnitudes as classes with more documents tend to have a larger output magnitude. These magnitudes can be normalized to a common scale by using a MLP.
2. Classification based on root node performs better. This observation can be explained as follows: the GNN architecture estimates the state of a node by integrating the contributions of its neighborhood nodes. As the GNN learns the structure of a graph iteratively, eventually the root node will have a consolidated state which allows it to separate the graph as a whole from other graphs. Using the outputs of other nodes to classify a graph can add noise. This is so because there are many similar substructures in the graphs even if they belong to different classes.

With more understanding on this learning task using GNN, the experiments are repeated with a balanced training dataset. This is achieved by weighting the training data with respect to the frequency of occurrence in the training set. Then the weights are normalized such that the total sum of all weights is equal to 1. For example, class “an” has 160 documents and class “mi” has 320 documents, the weights of each document in class “an” would be double those of the documents in class “mi”. Balancing the dataset produced much better results as shown in Table 5. The results confirm that an unbalanced distribution of training patterns can have a significant negative impact on the performance of a machine learning method.

The INEX XML Classification using structure only results it is observed that GNN is able to perform better than the baseline MLP method as reported in Section 3.1. This

Table 5. The results for GNN Classification based on root node with a balanced training dataset.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
<i>R</i>	0.44	0.68	0.28	0.40	0.46	0.51	0.47	0.55	0.77	0.43	0.39	0.33	0.48	0.45	0.50	0.42	0.59	0.68	
<i>P</i>	0.79	0.32	0.55	0.66	0.24	0.57	0.34	0.62	0.62	0.33	0.37	0.21	0.45	0.70	0.57	0.68	0.54	0.80	
	Macro F1: 0.48,				Micro F1: 0.34														

is an indication that the inclusion of structural information can help to improve the overall performance of a machine learning method.

It is important to understand that using GNN, the number of network parameters are generally greater than for a standard MLP. For the baseline MLP training, a network with 2 hidden-layer each containing 50 neurons is deployed. For the GNN, 2 neurons are used for each of the 2 hidden-layer of the *Transition Network*. The transition network featured furthermore 2 *states nodes* which also produce the output, and 2 neurons are used in the 2 hidden-layer of the *Output Network*. Due to architectural differences between MLP and GNN, a fair comparison between the two methods is difficult. However, in practise, the experiments in GNN typically finished within 12 hours on a 3GHz Intel machine with 2GB RAM. Thus, requiring considerably more time for training when compared to the MLP method.

During the training of GNN, it was noticed that the overfitting of GNN does not appear to occur, and that convergence to a stable Mean Square Error value occurred at about 1,000 to 1,500 training epochs. Since GNN does not appear to overfit in this learning problem, there appears no particular need for a validation dataset. Thus, in Section 3.2, the experiments are done using all of the data in the original training dataset.

3.2 Advanced experiments

Having identified a suitable approach to the training task, in the following we report experimental results which are based on utilizing 100% of the original training set for training purposes, and include test results based on the test dataset as provided by INEX 2006. In the following, all experiments are based on balancing the training data.

Better Understanding of INEX XML Document Structures: The results in Section 3.1 show that using structure only for classifying the INEX XML documents is not really practical. When the Test Dataset was released, some more experiments are performed to test the performance of GNN on the classification of either a document is in the class of “Transaction” or in the class of “Non-Transaction”. It is found that the results are of no significant differences (see Table 6). This means that in the INEX XML dataset, there is not much structural differences between “Transaction” and “Non-Transaction” documents.

GNN, Classification Using Structure and Content: A method needed to be found to compute a numerical representation of the text contained in a document. As a first approach, we extracted the textual content from each of the documents in the training

Table 6. The results for GNN Classification on “Transaction” and “Non-Transaction” classes of documents.

	Transaction	Non-Transaction	Precision
Transaction	2341	1623	0.59
Non-Transaction	657	1433	0.69
Recall	0.78	0.47	

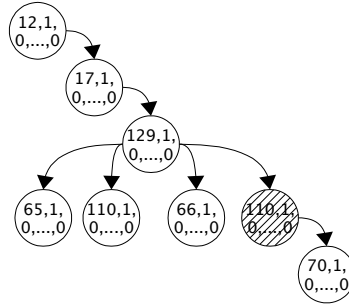


Fig. 7. Graph of XML Document with Multi-Dimensional Node Labels.

set, applied the Porter Stemming Algorithm[3] to obtain a numerical vector representing the text, then trained a naïve Bayes classifier[1] on these data. The naïve Bayes classifier produced an 18-dimensional binary vector indicating the class membership of the document (which is computed based on document’s textual content only). Thus, to generate the labels for each node in a graph, we applied the following procedure for each of the documents in the dataset:

- Step 1:** remove the XML tags;
- Step 2:** remove the stop words;
- Step 3:** perform Porter Stemming Algorithm[3] on the content of the documents;
- Step 4:** naïve Bayes classifier is trained on the training set;
- Step 5:** the XML documents are classified into 1 of the 18 classes and a 18 dimensional binary node labels is concatenated to the original node label (which represent XML tag) to form a 19 dimensional node label. The concatenation affects all nodes of the same graph.

For example, the root node of the graph shown in Fig. 6 would now have a node label of $\langle 12 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \rangle$ (assuming that the naïve Bayes classifier output is class 1) instead of just $\langle 12 \rangle$ as was shown in Fig. 7.

Naïve Bayes classifiers are fast to train, and hence, are a suitable method for pre-processing and labeling purposes. This paper considers two flavours of the Bayes classifier: the simple Bayes classifier, and the Bayes classifier using maximum entropy. The latter is computationally more demanding but can produce enhanced results.

The Bayes classifiers classify the documents based on content only. The result of the simple Bayes classifier on the test set is illustrated in Table 7. It can be seen that the classifier produces a performance which is somewhat better when compared to the results obtained on structure only approaches.

Table 7. Naïve Bayes on document content of test data only.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
<i>R</i>	0.29	0.42	0.72	0.21	0.88	0.27	0.41	0.07	0.49	0.39	0.06	0.76	0.88	0.55	0.26	0.54	0.87	0.27
<i>P</i>	0.96	0.87	0.38	0.88	0.77	0.94	0.89	0.90	0.79	0.81	1.00	0.63	0.36	0.58	0.91	0.82	0.67	0.88
	Macro F1: 0.50, Micro F1: 0.53																	

Table 8. Test results for GNN Classification using Structure and Content.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
<i>R</i>	0.94	0.75	0.69	0.54	0.70	0.76	0.64	0.17	0.53	0.48	0.34	0.81	0.82	0.82	0.83	0.90	0.96	0.81
<i>P</i>	0.80	0.76	0.49	0.89	0.83	0.82	0.68	0.92	0.70	0.81	0.79	0.69	0.83	0.77	0.88	0.87	0.89	0.78
	Macro F1: 0.72, Micro F1: 0.71																	

Table 9. Naïve Bayes using maximum entropy on document content of test data only

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
<i>R</i>	0.94	0.75	0.69	0.54	0.70	0.76	0.64	0.17	0.55	0.48	0.34	0.81	0.82	0.82	0.83	0.90	0.96	0.81
<i>P</i>	0.80	0.77	0.49	0.89	0.83	0.82	0.68	0.92	0.70	0.81	0.79	0.69	0.83	0.77	0.88	0.87	0.89	0.78
	Macro F1: 0.72, Micro F1: 0.72																	

The result of a GNN trained on the data labeled by this classifier is illustrated in Table 8. It can be seen that the performance of classification using this method is significantly increased. This may indicate that the incorporation of structural information into the learning task does indeed provide an input which allows for significantly improved results. The result on the test dataset shows that this method achieved Micro F1 of 0.721440 and Macro F1 of 0.713969, which is the best result obtained at the INEX XML classification competition⁸. The best result submitted by any competitor was: Micro F1=0.59, Macro F1 0.58.

A confirmation of the results and observations made so far is found through the application of the Bayes classifier using maximum entropy. The performance of this advanced classifier is illustrated in Table 9. It is seen that the advanced classifier performs virtually at the same level as the GNN when trained on data that were labeled by the simple classifier. We then labeled the nodes in the training set by the response of the advanced classifier, and re-trained the GNN accordingly. The result is shown in Table 10. It is seen that the incorporation of structural information has again helped to significantly improve the classification of the GNN.

4 Conclusions

This paper demonstrated for the first time that a supervised machine learning method capable of processing structured data is a very suitable approach for classifying possibly

⁸ Only one other party submitted results for the INEX 2006 classification competition despite of 41 registered participants. This confirmed our impression that the training task was very challenging.

Table 10. GNN trained on labels produced by the Naïve Bayes Maximum Entropy classifier.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
<i>R</i>	0.90	0.79	0.83	0.67	0.87	0.74	0.59	0.56	0.60	0.48	0.40	0.82	0.82	0.77	0.64	0.82	0.99	0.74
<i>P</i>	0.98	0.76	0.55	0.84	0.86	0.84	0.80	0.94	0.84	0.80	0.88	0.75	0.81	0.76	0.95	0.90	0.89	0.87
	Macro F1: 0.76, Micro F1: 0.76																	

large sets of XML formatted documents. While the training phase may seem time consuming, the application of a trained network to test data is very fast. The classification of all test pattern completed in a matter of minutes.

It was furthermore shown that the combination of both, structure and content, can help to improve the classification performance significantly. This seems to indicate that the XML structure may not be a feature that allows for an effective differentiation of the 18 pattern classes.

The encoding of document content into the training and test dataset can be improved. Instead of utilizing the entire textual content of a document and the attachment of identical labels to all nodes in a graph, it should be better to consider only the text that is encapsulated by the individual XML tags. This would allow for a distinct labeling of the nodes in a graph and should advocate an improved separation of the pattern classes. This approach is not covered in this paper and is left as a future task.

Acknowledgments

The work presented in this paper received financial support from the Australian Research Council in form of a Linkage International Grant and a Discovery Project grant.

References

1. L. Crnkovic-Dodig and P. Elkan. Classifier showdown. <http://blog.peltarion.com/2006/07/10/classifier-showdown/>.
2. C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.
3. M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.
4. F. Scarselli, S. Yong, M. Gori, M. Hagenbuchner, A. Tsoi, and M. Maggini. Graph neural networks for ranking web pages. In *Web Intelligence Conference*, 2005.
5. F. Scarselli, S. Yong, M. Hagenbuchner, and A. Tsoi. Adaptive page ranking with neural networks. In *14th International World Wide Web conference*, Alternate track papers and posters, pages 936–937, Chiba city, Japan, May 2005.
6. A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, Vol. 8(No. 3):714–735, 1997.
7. A. Tsoi, F. Scarselli, M. Gori, M. Hagenbuchner, and S. Yong. A neural network approach to web graph processing. In Y. Zhang, K. Tanaka, J. X. Yu, S. Wang, and M. Li, editors, *The Seventh Asia Pacific Web Conference*, Lecture Notes in Computer Science, pages pp 27 – 38, Shanghai, China, March 29 2005. <http://apweb05.csm.vu.edu.au/>, Springer Verlag. Keynote speech of the conference.
8. R. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. In *Neural Computation*, volume 1(2), pages 270–280, 1989.

The Wikipedia XML Corpus

Ludovic Denoyer, Patrick Gallinari
14th of April, 2006

Laboratoire d'Informatique de Paris 6
8 rue du capitaine Scott
75015 Paris

<http://www-connex.lip6.fr/denoyer/wikipediaXML>
{ludovic.denoyer, patrick.gallinari}@lip6.fr

1 Introduction

Wikipedia¹ is a well know free content, multilingual encyclopedia written collaboratively by contributors around the world. Anybody can edit an article using a wiki markup language that offers a simplified alternative to HTML. This encyclopedia is composed of millions of articles in different languages.

Content-oriented XML retrieval is an area of Information Retrieval (IR) research that is receiving an increasing interest. There already exists a very active community in the IR/ XML domain which started to work on XML search engines and XML textual data. This community is mainly organized since 2002 around the INEX initiative (INitiative for the Evaluation of XML Retrieval) which is funded by the DELOS network of excellence on Digital Libraries.

In this article, we describe a set of XML collections based on Wikipedia. These collections can be used in a large variety of XML IR/Machine Learning tasks like ad-hoc retrieval, categorization, clustering or structure mapping. These corpora are currently used for both, INEX 2006² and the XML Document Mining Challenge³. The article provides a description of the corpus.

The collections are downloadable on the website:

– <http://www-connex.lip6.fr/~denoyer/wikipediaXML>

2 Description of the corpus

The corpus is composed of 8 main collections corresponding to 8 different languages⁴ : English, French, German, Dutch, Spanish, Chinese, Arabian and Japanese. Each collection is a set of XML documents built using Wikipedia and encoded in UTF-8. In addition to these 8 collections, we also provide different *additional collections* for other IR/Machine Learning tasks like categorization and clustering, NLP, machine translation, multimedia IR, entity search, etc.

¹ <http://www.wikipedia.org>

² <http://inex.is.informatik.uni-duisburg.de/2006>

³ <http://xmlmining.lip6.fr>

⁴ Some additional languages will be added during the next months.

2.1 Main Collections

The main collections are a set of XML files in 8 different languages. The table 1 gives a detailed description of each collection.

Collection name	Language	Number of documents	Size of the collection (MegaBytes)
main-english	English	659,388	≈ 4,600
20060130_french	French	110,838	≈ 730
20060123_german	German	305,099	≈ 2,079
20060227_dutch	Dutch	125,004	≈ 607
20060130_spanish	Spanish	79,236	≈ 504
20060303_chinese	Chinese	56,661	≈ 360
20060326_arabian	Arabian	11,637	≈ 53
20060303_japanese	Japanese	187,492	≈ 1,425

Table 1. General statistics about the *Main Collections*

Each collection contains a set of documents where each filename is a number corresponding to the id of the file (for example : *15243.xml*). Each id is unique and each file corresponds to an article of Wikipedia. We only kept articles and removed all the wikipedia pages corresponding to "Talks", "Template", etc.. Each file is an UTF-8 document which is created from the wikitext of the original article. Figure 1 gives an example of an English article extracted from the corpus.

Tag labels We introduced different tags in order to represent the different parts of a document. We distinguish two types of tags:

- The general tags (*article, section, paragraph, ...*) that do not depend on the language of the collection. These tags correspond to the structural information contained in the wikitext format (for example : `== Main part ==` is transformed into `<title>Main part< /title>`)
- The template tags (*template_infobox, ...*) represent the information contained into the wikipedia templates. Wikipedia templates are used to represent a repetitive type of information. For example, each country described into wikipedia starts with a table containing its population, language, size, ... In order to uniformize this type of information, wikipedia uses templates. These templates are translated into XML using tags starting by *template_...* (for example : *template_country*). The template tags depend on the language of the collection because the templates are not the same depending on the language of the wikipedia collection used.

The DTD is downloadable on the Web site.

Statistics about the collections These statistics are given in table 2

Language	Mean size of document (bytes)	Mean Document Depth	Number of Nodes/Document
English	7,261	6.72	161.35
French	6,902	7.07	175.54
German	7,106	6.76	151.99
Dutch	5,092	6.41	122.8
Spanish	6,669	6.65	165.74
Chinese	6,664	6.91	179.23
Arabian	4,826	5.85	182.1
Japanese	7,973	7.1	94.96

Table 2. Statistics about the structure of the documents from the *Main Collections*

2.2 Categories

The documents of the wikipedia XML collections are organized in a hierarchy of categories defined by the authors of the articles. For each main collection, we propose a set of files describing:

- the hierarchy of categories (file : categories_hcategories.csv)
- the categories of each articles (file : categories_categories.csv)
- the categories names (file : categories_name.csv)

Table 3 gives statistics about the categories.

Language	Number of categories in the hierarchy	Mean number of categories for each document
English	113,483	2.2849
French	28,600	1.9570
German	27,981	2.5840
Dutch	13,847	1.6628
Spanish	12,462	1.6180
Chinese	27,147	2.0797
Japanese	26,730	2.0039

Table 3. Statistics about the categories of the *Main Collections*

3 Additional collections

We also propose additional collections. These collections can be used in a large variety of Information Retrieval and Machine Learning tasks. Some other collections will be added in the future and are not described here.

3.1 Categorization/Clustering Collections

English Multi-Label Categorization Collection For the English collection, we also provide a *Multi-Label Categorization Collection* with :

- A list of articles from the *Main English Collection*
- A set of categories (without hierarchy, based on the *Portals* of wikipedia): a document belongs to one or more categories

This corpus can be used to compare categorization algorithms and it is described in table 4.

Number of Documents	415,310
Number of categories	72
Mean number of categories for each document	2.2
Mean number of documents for each category	12,5
Number of documents in the larger category	137,9
Number of Documents in the smaller category	108

Table 4. Statistics about the *English Multi-Label Categorization Collection*

English Single-Label Categorization Collection - XML Document Mining Collection We provide a specific collection where each document belongs to **exactly** one category. It is composed of the documents of the preceding collection belonging to a single category. This collection can be used for categorization and clustering of documents (see table 5). This collection is aimed at categorization/clustering benchmark.

Number of categories	60
Number of documents	150,094
Number of train documents	75,047
Number of test documents	75,047
Mean number of categories for each document	1
Structure of the corpus	The directory <i>documents</i> contains all the corresponding articles. The directory <i>relfiles</i> contains one file per category giving the id of the documents that belongs to this category ⁵ .

Table 5. Statistics about the *XML Document Mining Challenge Collection (Single-Label Categorization Collection)*

3.2 Multimedia English Collection

This collection corresponds to the Main English Collection with the pictures of the different articles. This collection can be used for Multimedia Information Retrieval. Table 6 gives statistics about this collection.

Number of documents	659,388
Number of pictures	more than 300,000
Approximate size of the corpus	≈ 60Gb

Table 6. Statistics about the *Multimedia English Corpus*

3.3 Entity corpus

We provide an *Entity Corpus* where each article of the *Main English Corpus* has been tagged using a set of possible entity types extracted using the different categories of wikipedia. For example : *Silverster Stallone* has been tagged as `<actors>Silverster Stallone< /actors>`. Table 7 gives statistics about this collection.

Number of Documents	XML Entity collection
Number of documents	659,388
Size of the corpus	≈ 6 Gb

Table 7. Entity Collections

4 Conclusion

This technical report describes XML collections based on Wikipedia and developed for Structured Information Retrieval, Structured Machine Learning and Natural Language processing. Other collections will be added in the future.

5 Acknowledgment

The wikipediaXML corpus is distributed under the GPL Documentation license. It is completely free and can be used for non-profit educational and research purposes. All publications based on the wikipediaXML corpus should cite this technical report.



INEX 2006 Guidelines for Topic Development

Birger Larsen, Andrew Trotman, *et al*[†]

1 Aims

The aim of the INEX initiative is to provide the means, in the form of a large test collection and appropriate measures, for the evaluation of content-oriented XML element retrieval. Within the INEX initiative it is the task of the participating organizations to provide the topics and relevance assessments that will contribute to the test collection. Each participating organization, therefore, plays a vital role in this collaborative effort.

2 Introduction

Test collections, as traditionally used in information retrieval (IR), consist of three parts: a set of documents, a set of information needs called topics, and a set of relevance assessments listing (for each topic) the set of relevant documents.

A test collection for XML retrieval differs from traditional IR test collections in many respects. Although it still consists of the same three parts, the nature of these parts is fundamentally different. In IR test collections, documents are considered units of unstructured text, queries are generally treated as collections of terms and / or phrases, and relevance assessments provide judgments whether a document as a whole is relevant to a query or not. XML documents, on the other hand, organize their content into smaller, nested structural elements. Each of these elements in the document's hierarchy, along with the document itself (the root), is a retrievable unit. In addition, with the use of XML query languages users of an XML retrieval system can express their information need as a combination of content and structural conditions: they can restrict their search to specific structural elements within the collection. Consequently the relevance assessments for an XML collection must also consider the structural nature of a document and provide assessments at different levels of the document hierarchy.

This guide deals only with topics. Each group participating in INEX will have to submit **6 CO+S topics by 21st April 2006**. This guide provides detailed guidelines for creating these topics.

3 Topic Creation Criteria

Creating a set of topics for a test collection requires a balance between competing interests. The performance of retrieval systems varies largely for different topics. This variation is usually greater than the performance variation of different retrieval methods on the same topic. Thus, to judge whether one retrieval strategy is (in general) more effective than another, the retrieval performance must be averaged over a large and diverse set of topics. In addition, to be a useful diagnostic tool, the average performance of the retrieval systems on the topics can be neither too good nor too bad as little can be learned about retrieval strategies if systems retrieve no, or only relevant, documents.

When creating topics, a number of factors should be taken into consideration. Topics should:

- be authored by an expert in (or someone familiar with) the subject areas covered by the collection,
- reflect real needs of operational systems,
- represent the type of service an operational system might provide,
- be diverse,
- differ in their coverage, e.g. broad or narrow topic queries,
- be assessed by the topic author.

[†] Based on prior guidelines additionally authored by Börkur Sigurbjörnsson, Shlomo Geva, Mounia Lalmas, and Saadia Malik

4 Topic Format

In previous years, different topic types have been used for the two main *ad hoc* retrieval tasks at INEX (i.e., a distinction was made between Content Only (CO) and Content And Structure (CAS) topics). In addition, different parts of the topics were designed also to be used in other tracks (e.g., the topic <description> has been tuned to the needs of the Natural language Processing (NLP) track). Following initial trials at INEX 2005 and recommendations from the 2005 Dagstuhl workshop these topic types are all merged into one type for INEX 2006: **Content Only + Structure (CO+S)**. In the 2006 topics all the information needed by the different *ad hoc* tasks and tracks are expressed in the individual topic parts, and only one topic type is therefore needed. The 2006 CO+S topics consist of the following parts, which are explained in detail below:

<title>	in which Content Only (CO) queries are given
<castitle>	in which Content And Structure (CAS) queries are given
<description>	from which NLP queries are derived
<narrative>	in which the definitive definition of relevance and irrelevance are given
<ontopic_keywords>	in which terms that are expected in relevant elements are listed
<offtopic_keywords>	in which terms that are expected in non-relevant elements are listed

4.1 General considerations

A clear and precise description of the information need is required in order to unambiguously determine whether or not a given element fulfills the given need. In a test collection this description is known as the **narrative**. It is the only true and accurate interpretation of a user's needs. Precise recording of the narrative is important for scientific repeatability – there must exist, somewhere, a definitive description of what is and is not relevant to the user. To aid this, the <narrative> should explain not only what information is being sought, but also the context and motivation of the information need, i.e., *why* the information is being sought and what work-task it might help to solve.

Many different queries could be drawn from the <narrative>, and some are better than others. For example, some might contain phrases; some might contain ambiguous words; while some might even contain domain specific terms or structural constraints. Regardless of the query, the search engine results are not necessarily relevant. Even though a result might contain search terms from the query, it might not match the explanation given in the <narrative>. Equally, some relevant documents might not be found, but they remain relevant because they are described as so by the <narrative>.

The different CO+S topic parts relate to different scenarios that lead to different types of queries.

The topic <title> simulates a user who does not know (or does not want to use) the actual structure of the XML documents in a query. The query expressed in the topic <title> is therefore a Content Only (CO) query. This profile is likely to fit most users searching XML digital libraries.

Upon discovering their <title> query returned many irrelevant hits, a user might decide to add structural hints (to rewrite as a CAS query). This is similar to a user adding + and – to a web query when too many irrelevant pages are found. At INEX, these added structural constraints (+S) are specified using the formal syntax called NEXI [1] (see the INEX website for the specification) and recorded in the topic <castitle>.

Example

Suppose a user wants to find pictures of the Apple II computer. They enter the CO query:

```
Apple II figure
```

but discover that most results are figures of products for the Apple II. They decide to add structural hints:

```
//figure[about(., Apple II)]
```

restricting the results to `figure` elements only, known to contain the captions of figures.

4.1.1 The *Ad Hoc* Task

The CO+S task is investigating relevance ranking algorithms for *ad hoc* element retrieval. This year the task is specifically investigating the usefulness of structural hints. At INEX 2006 it will be possible to compare the performance (on the same topic) of using structural hints to that of not using structural hints.

4.1.2 The NLP Task

As an alternative to entering queries into search engines, a user might ask a librarian to find the information to satisfy their need. Such a user would give a verbal description to the librarian using a natural language. The NLP track at INEX is examining the ability of a search engine to satisfy the information need given this natural language description (recorded in the topic **<description>**).

Just as there are many CO queries derivable from the **<narrative>**, there are many ways to express the need in natural language. However it is expressed, it is important that it matches the **<narrative>** while at the same time it is not the **<narrative>**.

The purpose of the NLP experiments at INEX 2006 is to compare the performance of CO, CO+S and NLP techniques. To do this it is important that the same terms are used in each version of the query.

Example

Suppose a user wants to find pictures of the Apple II computer and chose the CAS query:

```
//figure[about(., apple II)]
```

the NLP version would contain the same words and structural constraints:

```
Show me a figure of the Apple II
```

which is a brief matter of fact description of the information need.

4.1.3 On-Topic and Off-Topic Terms

The 2006 CO+S topics contain on-topic and off-topic keywords. These are keywords that are either likely to be found in relevant **or** irrelevant elements retrieved by the user's query. They are recorded in the **<ontopic_keywords>** and **<offtopic_keywords>** topic parts. These keywords are needed for special tests (at INEX 2006) into the possibility of doing automatic assessment.

Example

On-topic keywords for the user's information need for pictures of the Apple II computer might be:

```
macintosh ; "personal computer" ; photos ; images ; posters
```

while off-topic keywords might be:

```
fruit ; "New York" ; Beatles ; granny
```

4.2 Topic parts

Topics are made up of several parts, these parts explain the *same information need*, but for different purposes. An example of a full topic combining all these is given in the Appendix.

<narrative> A detailed explanation of the information need and the description of what makes an element relevant or not. The **<narrative>** should explain not only what information is being sought, but also the context and motivation of the information need, i.e., *why* the information is being sought and what work-task it might help to solve. Assessments will be made on compliance to the narrative alone; it is therefore important that this description is clear and precise.

<title> A short explanation of the information need. It serves as a summary of the content of the user's information need. The exact format of the topic title is discussed in more detail below.

<castitle> A short explanation of the information need, specifying any structural requirements. The exact format of the castitle is discussed in more detail below. The castitle is optional but the majority of topics should include one.

<description> A brief description of the information need written in natural language – to be used in the NLP track. The description must be precise, concise, and as informative as the <title> and <castitle> combined. The topic description is discussed in more detail below.

Note that the <description> must be interchangeable with the <title> and <castitle>. Any ambiguity or disagreement is resolved by reference to the <narrative>, the only accurate definition of the information need.

<ontopic_keywords> Terms and phrases that are likely to appear in most relevant documents. For example, if the user is searching for information about element retrieval and the query has the title INEX then on-topic terms might be: `element, xml`.

<offtopic_keywords> Terms that are likely to appear in documents retrieved by the query, while at the same time are not likely to appear in relevant documents. Some queries are genuinely ambiguous and such terms can be used to disambiguate the correct from incorrect interpretation of the query. For example, if the user is searching for information about element retrieval and the query has the title INEX then off-topic terms might be: `neutral, biopharmaceutical, nanotechnology`

4.2.1 Topic <title>

To ensure topics are syntactically correct, a parser has been implemented in Flex and Bison (the GNU tools compatible with LEX and YACC) and is available for download or online use (see <http://metis.otago.ac.nz/abin/nexi.cgi>)

The topic title is a short representation of the information need. Each term is either a word or a phrase. Phrases are encapsulated in double quotes. Furthermore the terms can have either the prefix + or –, where + is used to emphasize an important concept, and – is used to denote an unwanted concept.

Example

A user wants to retrieve information about computer science degrees that are not master degrees:

```
"computer science" +degree -master
```

the + and – signs are used as hints to the search engine and do not have strict semantics. As an example the following text might be judged relevant to the information need, even though it contains the word master.

```
The university offers a program leading to a PhD degree in computer science. Applicants must have a master degree...
```

Example

A user wants to retrieve information about IR from semi-structured documents:

```
"information retrieval" +semi-structured documents
```

As in the previous example the following text might be judged relevant, even though it neither contains the word semi-structured, nor the phrase “information retrieval”.

```
The main goal of INEX is to promote the evaluation of content-oriented XML retrieval by providing a large test collection of XML documents, uniform scoring procedures, and a forum for organizations to compare their results...
```

Although the semantics of phrases and the + / – tokens is not strict, they may be of use to the retrieval engine.

4.2.2 Topic <castitle>

As structural constraints are not an inherent part of all information needs the <castitle> is optional. However, we aim at having topics for INEX 2006 where the **majority of topics do include a castitle**. This is needed in order to facilitate the evaluation of structural hints, which is a central concern at INEX.

Only a high level description is included here, for a more formal specification of the topic description language (NEXI) see the INEX web-site or in the proceedings of INEX 2004 [1].

To make sure that topics are syntactically correct, parsers have been implemented in Flex and Bison (the GNU tools compatible with LEX and YACC) and are available for download. An online version of the parser is also available: <http://metis.otago.ac.nz/abin/nexi.cgi>

Castitles are XPath (<http://www.w3c.org/TR/xpath>) expressions of the form:

A[B]

or

A[B]C[D]

where *A* and *C* are navigational XPath expressions using only the descendant axis. *B* and *D* are predicates using *about* functions for text (explained below); the arithmetic operators *<*, *<=*, *>*, and *>=* for numbers; and the connectives *and* and *or*. The *about* function has (nearly) the same syntax as the XPath function *contains*. Usage is restricted to the form:

```
about(.path, query)
```

where *path* is empty or contains only tag-names and descendant axis; and *query* is an IR query having the same syntax as the CO titles (i.e. query terms). The *about* function denotes that the content of the element located by the path is about the information need expressed in the query. As with the title, the castitle is only a hint to the search engine and does not have definite semantics.

Example

A user wants to know about Tolkien's languages and assumes an article on Tolkien will have a section discussing these languages:

```
//article[about(., Tolkien)]//section[about(., language)]
```

But the user might be happy with retrieving whole articles. In the formalism expressed above,

```
A = //article
B = about(., Tolkien)
C = //section
D = about(., language)
```

A CAS query contains two kinds of structural hints: where to look (support elements; in this case *//article* and *//section*), and which elements to return (target elements; in this case *//article//section*). In prior INEX workshops the target element hint has been interpreted either strictly or loosely (vaguely). Where to look has always been interpreted loosely. This created considerable debate over how to interpret where to look. There is the database view: *all* structural constraints must be followed strictly (by exact match). Then there is the information retrieval view: an element is relevant if it satisfies the information need, irrespective of the structural constraints.

The main purpose of the INEX initiative is to build a test collection for the evaluation of content oriented XML retrieval. The most valuable part of the collection is the human made relevance assessments. Thus, each structured query **must** have at least one *about* function in the rightmost predicate.

4.2.3 Topic <description>

The <description> should be precise and concise, but it must contain the same terms and the same structural requirements that appear in the <title> and the <castitle>, albeit expressed in natural language.

Example

A user wants to retrieve information about computer science degrees that are not master degrees and has chosen the title query:

```
"computer science" +degrees -master
```

for the <title>. From this they might choose a <description> of either:

```
retrieve information about degrees in computing science, but not masters degrees
```

or

```
I want descriptions of computer science degrees that are not master degrees
```

as they are equivalent, but:

```
get information about computing degrees, but not about master or PhD computing degrees
```

cannot be chosen as it expresses a different information need - there is an additional requirement that information about PhD degrees is not sought.

It is important to compare results that are based on natural language queries (the <description>) with queries that are based on the more formal languages (the <title>, and <castitle>). The description must, therefore, be as informative as the <title> and <castitle>.

5 Procedure for Topic Development

Each participating group will have to submit **6 CO+S topics** by the **21st April 2006**. Submission is done by filling in the Candidate Topic Submission Form on the INEX web site:
[http://inex.is.informatik.uni-duisburg.de/2006/ under Tasks/Tracks → Adhoc → Topics](http://inex.is.informatik.uni-duisburg.de/2006/under%20Tasks/Tracks%20-%20Adhoc%20-%20Topics).

The topic creation process is divided into several steps. When developing a topic, use a print out of the online Candidate Topic Form to record all information about the topic you are creating.

Step 1: Initial Topic Statement

Create a one or two sentence description of the information you are seeking. This should be a simple description of the information need without regard to retrieval system capabilities or document collection peculiarities. This should be recorded in the Initial Topic Statement field. Record also the context and motivation of the information need, i.e. *why* the information is being sought. Add to this a description of the *work-task*, that is, with what task it is to help (e.g. writing an essay on a given topic).

Step 2: Exploration Phase

In this step the initial topic statement is used to explore the collection. Obtain an estimate of the number of relevant elements then evaluate whether this topic can be judged consistently. You may use any retrieval engine for this task, including your own or the TopX system (<http://infao5501.ag5.mpi-sb.mpg.de:8080/topx/>), provided through the INEX website.

While exploring the collection make a list of the on-topic and off-topic terms that might be used to distinguish between relevant and irrelevant results retrieved by the search engine.

Step 2a: Assess Top 25 Results

Judge the top 25 retrieval results. To assess the relevance of a retrieved element use the following working definition: *mark it relevant if it would be useful if you were writing a report on the subject of the topic, or if it contributes toward satisfying your information need*. Each result should be judged on its own merits. That is, information is still relevant even if it is the thirtieth time you have seen the same information. It is important that your judgment of relevance is consistent throughout this task. Using

the Candidate Topic Submission Form record the number of found relevant elements and the path representing each relevant element. Then if there are:

- fewer than 2 or more than 20 relevant within the top 25, abandon the topic and use a new one,
- more than 2 and fewer than 20 relevant within the top 25, perform a feedback search (see below).

Step 2b: Feedback Search

After assessing the top 25 elements, you should have an idea of which terms (if any) could be added to the query to make the query as expressive as possible for the kind of elements you wish to retrieve. You should also have an idea of which terms could be used to disambiguate relevant from irrelevant results.

Use the expanded query, to retrieve a new list of candidates. Judge the top 100 results (some are already judged), and record the number of relevant results in Candidate Topic Form. Record the expanded query in the title field of the Candidate Topic Submission Form. Now record the on-topic and off-topic terms on the Candidate Topic Submission Form.

Step 3: Write the <narrative>

Having judged the top 100 results you should have a clear idea of what makes a component relevant or not. It is important to record this in minute detail as the <narrative> of the topic. The <narrative> is the definitive instruction used to determine relevance during the assessment phase (after runs have been submitted). Record not only what information is being sought, but also what makes it relevant or irrelevant. Also record the context and motivation of the information need. Include the work-task, which is: the form the information will take after having been found (e.g. written report). Make sure your description is exhaustive as there will be several months between topic development and topic assessment.

Step 4 CO+S: Optionally write the <castitle>

Optionally re-write the title by adding structural constraints and target elements. Record this as the <castitle> on the Candidate Topic Submission Form. Also record why you think the structural hints might help in the <narrative>. Please note that we aim at having castitles in most topics.

Step 5: Write the <description>

Write the <description>, the natural language interpretation of the query. Ensure the information need as expressed in the <title>, and <castitle> is also expressed in the <description>. Make sure the <description> does not express any additional information needs.

Step 6: Add ontopic and offtopic keywords

Based on the documents you have seen, add a set of ontopic keywords (terms that are likely to be found in relevant documents and elements), and offtopic keywords (terms that are likely to be found in irrelevant documents and elements). The more terms the better.

Step 7: Refining Topic Statements

Finalize the topic <title>, <castitle>, <description>, and <narrative>. It is important that these parts all express the same information need; it should be possible to use each part of a topic in a stand-alone fashion (e.g. title for retrieval, description for NLP, etc.). In case of dispute, the <narrative> is the definitive definition of the information need – all assessments are made relative to the <narrative> and the <narrative> alone.

Step 8: Topic Submission

Once you are finished, fill out and submit the on-line Candidate Topic Submission Form on the INEX website <http://inex.is.informatik.uni-duisburg.de/2006/> under Tasks/Tracks → Adhoc → Topics. After submitting a topic you will be asked to fill out an online questionnaire (this should take no longer than 5-10 minutes). It is important that this is done as part of the topic submission as the questions relate to the individual topic just submitted and the submission process. This is part of an effort to collect more context for the INEX topics as discussed at the Dagstuhl workshop.

Please make sure you submit all candidate topics no later than the 21st April 2006.

6 Topic Selection

From the received candidate topics, the INEX organizers will decide which topics to include in the final set. This is done to ensure inclusion of a broad set of topics. The data obtained from the collection exploration phase *is* used as part of the topic selection process. The final set of topics will be distributed for use in retrieval and evaluation.

7 Acknowledgments

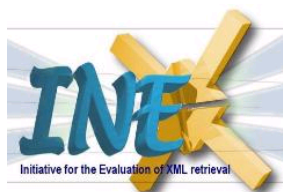
The topic format proposed in this document is based on the outcome of working groups set up during previous INEX workshops along with the online discussions they created. We are very grateful for this contribution. This document is a modified version of the topic development guides from previous INEX workshops additionally authored by Börkur Sigurbjörnsson, Shlomo Geva, Mounia Lalmas, and Saadia Malik.

References

- [1] Trotman, A., & Sigurbjörnsson, B. (2004). Narrowed Extended XPath I (NEXI). In *Proceedings of the INEX 2004 Workshop*, (pp. 16-40).

Appendix 1: Example CO+S Topic

```
<inex_topic query_type="CO+S">
<title>Tolkien languages "lord of the rings"</title>
<castitle>//article[about(., Tolkien) or about(., "lord of the rings")]//sec[about(.,
Tolkien languages)]</castitle>
<description>Find information about Tolkien languages from the Lord of the
Rings.</description>
<narrative>The "Lord of the Rings" movie trilogy fascinate me. I have learned from
other fans that the languages spoken by e.g., elves and dwarfs in the screen version
are not just the usual effects. Apparently, these languages were invented by Tolkien
himself and are central to his work with the original books.
For my own personal interest, I would like to learn more background about Tolkien's
artificial languages, and how they have affected the world portrayed in the Lord of
the Rings universe. Later I may want to add a section on the influence languages to my
Lord of the Rings fan web page. As Tolkien's languages seem to be a rather specialized
topic, I expect to find relevant information as elements in larger documents that deal
with Tolkien or Lord of the Rings, e.g., as sections in documents about Tolkien or the
Lord of the Rings (although I would be pleasantly surprised to see whole documents on
the topic of Tolkien's languages).
To be relevant an element should discuss Tolkien's artificial languages and their
influence on the Lord of the Rings books or movies. Information on the languages alone
without explicit discussion of their impact on the books or movies is not relevant;
nor is general information on Tolkien or the Lord of the Rings.</narrative>
<ontopic_keywords>"High Elvish" ; Quenya ; Sindarin</ontopic_keywords>
<offtopic_keywords>inspired, film</offtopic_keywords>
</inex_topic>
```

INEX 2006 Retrieval Task and Result Submission Specification

Charlie Clarke, Jaap Kamps, Mounia Lalmas

Saturday, May 20, 2006

1 Retrieval Task

The retrieval task to be performed by the participating groups of INEX 2006 is defined as the ad-hoc retrieval of XML elements. In information retrieval (IR) literature, ad-hoc retrieval is described as a simulation of how a library might be used, and it involves the searching of a static set of documents using a new set of topics. While the principle is the same, the difference for INEX is that the library consists of XML documents, the queries may contain both content and structural conditions and, in response to a query, arbitrary XML elements may be retrieved from the library.

The general aim of an IR system is to find *relevant information* for a given topic of request. In the case of XML retrieval there is, for each article containing relevant information, a choice from a whole hierarchy of different elements to return. Hence, within XML retrieval, we regard as *relevant elements* those XML elements that both

- contain relevant information (the element exhaustively discusses the topic), but
- do not contain non-relevant information (the element is specific for the topic).

That is, if an XML element contains another element but they have the same amount of relevant text, the shorter element is strictly more specific and a preferred result.

Within the ad-hoc XML retrieval task we define the following four sub-tasks:

1. THOROUGH TASK asks systems to estimate the relevance of elements in the collection.
2. FOCUSED TASK asks systems to return a ranked list of elements to the user.
3. RELEVANT IN CONTEXT TASK asks systems to return relevant elements clustered per article to the user.
4. BEST IN CONTEXT TASK asks systems to return articles with one best entry point to the user.

1.1 THOROUGH TASK

1.1.1 Motivation for the Task

The core system's task underlying most XML retrieval strategies is the ability to estimate the relevance of potentially retrievable elements in the collection. Hence, the INEX 2006 THOROUGH TASK simply asks systems to return elements ranked by their relevance to the topic of request. Since the retrieved elements are meant for further processing (either by a dedicated interface, or by other tools) there are no display-related assumptions nor user-related assumptions underlying the task.

What we hope to learn from this task is: How well are systems capable of estimating the relevance of XML elements? How well are systems capable of locating all the relevant elements in the collection? How do structural constraints in the query help retrieval?

1.1.2 Results to Return

The aim of the THOROUGH TASK is to find all relevant elements ranked in relevance order. It will be therefore the case that, due to the nature of relevance in XML retrieval (e.g. if a child element is relevant, so will be its parent, although to a greater or lesser extent), an XML retrieval system that has estimated an element to be relevant may decide to return all its ancestor elements. This means that runs for this task may contain a large number of overlapping elements. It is however a challenge to rank these elements appropriately.

Summarizing: THOROUGH TASK returns elements ranked in relevance order (where specificity is rewarded). Overlap is permitted.

1.2 FOCUSED TASK

1.2.1 Motivation for the Task

A continuation of the Focused retrieval strategy from INEX 2005, the scenario underlying the FOCUSED TASK is to return to the user a ranked-list of elements for her topic of request. The INEX 2006 FOCUSED TASK asks systems to find the most focused elements that satisfy a (focused) information need, without returning “overlapping” elements. That is, for a given topic, no element in the result set may contain text already contained in another element. Or, in terms of the XML tree, no element in the result set should be a child or descendant of another element. The task makes a number of assumptions:

Display the results are presented as a ranked-list of elements to the user.

Users view the result list top-down, one-by-one. Users do not want overlapping elements in the result-list, and prefer smaller elements over larger ones (if equally relevant). User is mostly concerned with what happens at the early ranks.

What we hope to learn from this task is: How does the user-oriented FOCUSED TASK differ from system-oriented THOROUGH TASK? Can the FOCUSED TASK be reduced to a straightforward filter on the THOROUGH TASK? What techniques are effective at the early ranks? How do structural constraints in the query help retrieval?

1.2.2 Results to Return

The aim of the FOCUSED TASK is to return a ranked-list of elements, where no element may be overlapping with any other element. Hence the decision to return a particular element to the user will outlaw all its ancestors, as well as all its descendants to be returned. Since all these ancestors and possibly also the descendants are relevant (be it to a lesser or greater extent) it is however a challenge to chose the elements appropriately. *Please note that submitted runs containing overlapping elements will be disqualified.*

Summarizing: FOCUSED TASK returns elements ranked in relevance order (where specificity is rewarded). Overlap is **not** permitted in the submitted run.

1.3 RELEVANT IN CONTEXT TASK

1.3.1 Motivation for the Task

The scenario underlying the RELEVANT IN CONTEXT TASK is to return the relevant information (captured by a set of elements) within the context of the full article. As

a result, an article devoted to the topic of request, will contain a lot of relevant information across many elements. The INEX 2006 RELEVANT IN CONTEXT TASK asks systems to find a set of elements that corresponds well to (all) relevant information in each article. The task make a number of assumptions:

Display results will be grouped per article, in their original document order, providing access through further navigational means.

Users consider the article as the most natural unit, and prefer an overview of relevance in their context.

What we hope to learn from this task is: How does the user-oriented RELEVANT IN CONTEXT TASK differ from THOROUGH TASK? What techniques are effective at locating relevance within articles? How do structural constraints in the query help retrieval?

The RELEVANT IN CONTEXT TASK is based on the INEX 2005 Fetch-And-Browse retrieval strategy.

1.3.2 Results to Return

The aim of the RELEVANT IN CONTEXT TASK is to first identify relevant articles (the fetching phase), and then to identify the relevant elements within the fetched articles (the browsing phase). In the fetching phase, articles should be ranked according to their topical relevance. In the browsing phase, we have a set of elements that cover the relevant information in the article. The `//article[1]` element itself need not be returned, but is implied by any result from a given article. Since the content of an element is fully contained in its parent element and ascendants, the set may **not** contain overlapping elements. *Please note that submitted runs containing results from interleaved articles will be disqualified, as will submitted runs containing overlapping elements.*

Summarizing: RELEVANT IN CONTEXT TASK returns a ranked list of articles. For each article, it returns an unranked **set** of elements, covering the relevant material in the article. Overlap is not permitted.

1.4 BEST IN CONTEXT TASK

1.4.1 Motivation for the Task

The scenario underlying the BEST IN CONTEXT TASK is to find the best-entry-point for starting to read articles with relevance. As a result, even an article completely devoted to the topic of request, will only have one best starting point to read. The INEX 2006 BEST IN CONTEXT TASK asks systems to find the XML elements that corresponds to these best-entry-points. The task make a number of assumptions:

Display single result per article.

Users consider the article as the most natural unit, and prefer to be guided to the best point to start to read the most relevant content.

What we hope to learn from this task is: How does the BEST IN CONTEXT TASK differ from the RELEVANT IN CONTEXT TASK? How do best-entry points relate to the relevance of elements (and THOROUGH TASK and FOCUSED TASK)? How do structural constraints in the query help retrieval?

The BEST IN CONTEXT TASK is also based on the INEX 2005 Fetch-And-Browse retrieval strategy.

1.4.2 Results to Return

The aim of the BEST IN CONTEXT TASK is to first identify relevant articles (the fetching phase), and then to identify the element corresponding to the best entry points for the fetched articles (the browsing phase). In the fetching phase, articles should be ranked according to their topical relevance. In the browsing phase, we have a single element whose opening tag corresponds to the best entry point for starting to read the relevant information in the article. Note that there is no implied end-point: if (the start of) a paragraph is returned, it's not indicating that the reader should stop at the end of the paragraph. The `//article[1]` element itself may be returned in case it is the best entry point, otherwise it will implied by any result from a given article. *Please note that submitted runs containing multiple results per article will be disqualified.*

Summarizing: BEST IN CONTEXT TASK returns a ranked list of articles. For each article, it returns a **single** element, representing the best entry point for the article with respect to the topic of request.

1.5 Structured Queries

Queries with content-only conditions (CO queries) are requests that ignore the document structure and contain only content related conditions, e.g. only specify what an element should be about without specifying what that component is. The need for this type of query for the evaluation of XML retrieval stems from the fact that users may not care about the structure of the result components or may not be familiar with the exact structure of the XML documents. CAS queries are more expressive topic statements that contain explicit references to the XML structure, and explicitly specify the contexts of the user's interest (e.g. target elements) and/or the context of certain search concepts (e.g. containment conditions). More precisely, a CAS query contains two kinds of structural constraints: where to look (i.e. the support elements), and what to return (i.e. the target elements). The structural constraints are considered as structural hints, and similar to CO queries the elements will be assessed using the `<narrative>` part of the topics. Runs using CO queries and runs using CAS queries will be merged to create the assessment pool (this will in fact improve the pool quality).

At INEX 2006, there is no separate CAS task, but the vast majority of topics have both a keyword CO query and a structured CAS query.¹ As noted above, for all the tasks, we want to find out if, when and how the structural constraints in the query have an impact on retrieval effectiveness. Although both types of queries may be used for each task, mixing runs with both query types, the best performing CAS query runs (restricted to topics containing a CAS query) will be reported. The use of CO/CAS query fields is recorded in submission format.

2 Result Submission

Fact sheet:

¹Of course, any CO query can be directly rephrased as a CAS query `//*[about(.,"CO query")]` using the tag wildcard `*` that matches any element.

- For all four tasks, we allow up to 3 CO submissions, and up to 3 CAS submissions. That is, a participant can never submit more than 24 runs in total.
- All participants are required to submit a title-only run (free choice between the CO title and the CAS title) for the THOROUGH TASK.
- There is a common format for all submission files (details below), which allows up to 1,500 elements per topic.
- There are additional requirements on the submissions for three out of the four tasks:
 - FOCUSED TASK: for the same topic, results may not be overlapping.
 - RELEVANT IN CONTEXT TASK: articles may not be interleaved, and results may not be overlapping.
 - BEST IN CONTEXT TASK: only **one single** result per article is allowed.

Runs that violate these requirements in any way, will be disqualified.

2.1 INEX 2006 Topics

There is only one set of topics to be used for all ad-hoc retrieval tasks at INEX 2006. The format of the topics is defined in the following DTD:

```
<!ELEMENT inex_topic
  (title,castitle?,description,narrative,ontopic_keywords)>
<!ATTLIST inex_topic
  id      CDATA #REQUIRED
  ct_no   CDATA #REQUIRED
>
<!ELEMENT title          (#PCDATA)>
<!ELEMENT castitle      (#PCDATA)>
<!ELEMENT parent        (#PCDATA)>
<!ELEMENT description   (#PCDATA)>
<!ELEMENT narrative     (#PCDATA)>
<!ELEMENT ontopic_keywords (#PCDATA)>
```

The submission format will record the precise topic fields that are used in a run. Participants are allowed to use all fields, but only runs using either the `<title>`, `<castitle>`, or `<description>` fields, or a combination of these, will be regarded as truly *automatic*, since the additional fields will not be available in operational settings.

The `<title>` part of the INEX 2006 topics should be used as queries for the CO submissions. The `<castitle>` part of the INEX 2006 topics should be used as queries for the CAS submissions. In the number of runs allowed to be submitted, runs using more fields than the `<title>` (or `<castitle>`) will still be regarded as an CO (or CAS) submission.

Since the comparative analysis of CO and CAS queries is a main research question at INEX 2006, we encourage participant to submit runs using only the `<title>` field (CO query) or only the `<castitle>` field (CAS query). We do not outlaw the use of the other topic fields, to allow participants to conduct their own experiments involving them, and since such deviating runs may in fact improve the quality of the assessment pool.

2.2 Runs

There is an obligatory run, which is a submission to the THOROUGH TASK, using only the short topic statement from either the `<title>` or the `<castitle>` field of the topics.

For each of the four tasks, we allow up to 3 CO submissions, and up to 3 CAS submissions. The results of one run must be contained in one submission file (i.e. up to 24 files can be submitted in total). A submission may contain up to 1,500 retrieval results for each of the INEX topics included within that task.

There are however a number of additional task-specific requirements.

For the THOROUGH TASK there are no further restrictions.

For the FOCUSED TASK, it is not allowed to retrieve elements than contain text already retrieved by another element. That is, within the same article, the element `//article[1]//section[1]` is disjoint from `//article[1]//section[2]`, but overlapping with all ancestors (e.g., `//article[1]`) and all descendants (e.g., `//article[1]//section[1]//p[1]`).

For the RELEVANT IN CONTEXT TASK, articles may not be interleaved. That is, if an element of article a is retrieved, and then an element of a different article b, then it is not allowed to retrieve further elements from article a. Additionally, it is not allowed to retrieve elements than contain text already retrieved by another element (similar to the FOCUSED TASK). Note also that for this task the `//article[1]` element is implied by any element of the article, and need not be returned.

For the BEST IN CONTEXT TASK, only a single element per article is allowed. The `//article[1]` element may be returned in case it is regarded as the best place to start reading, otherwise it is implied by any other element from this article. Note that for this task, the `//article[1]` element may be returned in case it is regarded as the best element to start reading the relevant information in the article, otherwise it will be implied by any element of the article.

2.2.1 Submission format

For relevance assessments and the evaluation of the results we require submission files to be in the format described in this section. The submission format for all tasks is defined in the following DTD:

```
<!ELEMENT inex-submission (topic-fields, description, collections, topic+)>
<!ATTLIST inex-submission
  participant-id CDATA #REQUIRED
  run-id         CDATA #REQUIRED
  task          (Thorough | Focused | AllInContext | BestInContext ) #REQUIRED
  query        (automatic | manual) #REQUIRED
>
<!ELEMENT topic-fields EMPTY>
<!ATTLIST topic-fields
  title          (yes|no) #REQUIRED
  castitle       (yes|no) #REQUIRED
  description    (yes|no) #REQUIRED
  narrative      (yes|no) #REQUIRED
  ontopic_keywords (yes|no) #REQUIRED
>
<!ELEMENT description (#PCDATA)>
<!ELEMENT topic (result*)>
<!ATTLIST topic topic-id CDATA #REQUIRED >
<!ELEMENT collections (collection+)>
<!ELEMENT collection (#PCDATA)>
<!ELEMENT result (in?,file, path, rank?, rsv?)>
<!ELEMENT in (#PCDATA)>
<!ELEMENT file(#PCDATA)>
```

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

Each submission must contain the participant ID of the submitting institute (available at the INEX web-site <http://inex.is.informatik.uni-duisburg.de/2006/ShowParticipants.html>), a run ID (which must be unique for the submissions sent from one organization – also please use meaningful names as much as possible), the identification of the task (e.g. Thorough, Focused, etc), and the identification of whether the query was constructed automatically or manually from the topic. Furthermore, the used topic fields must be indicated in the `<topic-fields>` tag. Furthermore each submitted run must contain a description of the retrieval approach applied to generate the search results. A submission contains a number of topics, each identified by its topic ID (as provided with the topics).

For compatibility with the heterogeneous collection track, the `<collections>` tag is mandatory. There should be with `<collections>` at least one `<collection>` tag, which is by default set to "wikipedia" for the ad hoc track. The `<in>` tag is optional for the ad hoc track (`<in>` states from which collection each result comes from).

For each topic a maximum of 1500 result elements may be included per task. A result element is described by a file name and an element path, and it may include rank and/or retrieval status value (rsv) information. For the ad hoc retrieval task, `<collection>` is set to "wikipedia". Here is a sample submission file for the THOROUGH TASK:

```
<inex-submission participant-id="12" run-id="VSM_Aggr_06"
  task="Thorough" query="automatic">
  <topic-fields title="no" castitle="yes" description="no"
    narrative="no" ontopic_keywords="no"/>
  <description>Using VSM to compute RSV at leaf level combined with
    aggregation at retrieval time, assuming independence and using
    augmentation weight=0.6.</description>
  <collections>
    <collection>wikipedia</collection>
  </collections>
  <topic topic-id="01">
    <result>
      <file>9996</file>
      <path>/article[1]</path>
      <rsv>0.67</rsv>
    </result>
    <result>
      <file>9996</file>
      <path>/article[1]/name[1]</path>
      <rsv>0.1</rsv>
    </result>
    [ ... ]
  </topic>
  <topic topic-id="02">
    [ ... ]
  </topic>
  [ ... ]
</inex-submission>
```

Rank and RSV The rank and rsv elements are provided for submissions based on a retrieval approach producing ranked output. The ranking of the result elements can be described in terms of:

- Rank values, which are consecutive natural numbers, starting with 1. 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. The file names in the Wikipedia collection uniquely define an article, so there is no need for including the directory in which the file resides (in contrast with the earlier IEEE collection). The extension .xml must be left out. Example:

9996

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 ::= *ElementName Index*

AttributeNode ::= '@' *AttributeName*

Index ::= '[' *integer* ']'

Example:

/article[1]/body[1]/section[2]/p[1]

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 "body" element, within which we select the second "section" element, and finally within that element we select the first "p" element. Important: XPath counts elements starting with 1 and takes into account the element type, e.g. if a section had a title and two paragraphs then their paths would be given as: ../title[1], ../p[1] and ../p[2].

A result element may then be identified unambiguously using the combination of its file name and element path. Example:

```
<result>
  <in>wikipedia</in>
  <file>9996</file>
  <path>/article[1]/body[1]/section[2]/p[1]</path>
</result>
```

2.3 Result Submission Procedure

To submit a run, please use the following link: <http://inex.is.informatik.uni-duisburg.de/2006/> Then go to Tasks/Tracks, Adhoc, Submissions. The online submission tool will be available soon.



INEX 2006 Relevance Assessment Guide

1. Introduction

During the retrieval runs, participating organisations evaluated the 125 INEX 2006 topics (CO+S) against the Wikipedia 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 2005 test collection. This requires first a definition of relevance (Section 2), followed by details of how to assess (Section 3). Finally, we describe the on-line relevance assessment system that should be used to record your assessments (Section 4).

2. Relevance in INEX

Relevance in INEX is defined according to the notion of **specificity**, which describes the extent to which the document component focuses on the topic of request. This definition was adopted after a number of studies that showed that in terms of retrieval effectiveness, the same conclusions could be in most cases generated from using the specificity dimension of relevance compared to using more complex definitions. Up to INEX 2005, relevance was defined according to two dimensions, specificity and exhaustivity. The latter describes the extent to which the document component discusses the topic of request. This year, only the specificity dimension is used. Its measuring is based on the highlighting procedure used in INEX 2005. The main advantage of this highlighting approach is the specificity of any (partially highlighted) elements can be calculated automatically as some function of the contained relevant and irrelevant content (e.g. in the simplest case as the ratio of relevant content to all content, measured in number of words or characters).

3. How to assess

The assessment process is to be done as follows. Assessors highlight text fragments that contain only relevant information. It is important that only purely relevant information fragments get highlighted. To decide which text to highlight, you should skim-read the whole article and identify any relevant information as you go along. The on-line system can assist you in this task by highlighting keywords (that are chosen using the interface) and pool elements (elements retrieved by participating systems) within the article (see Section 5). If you highlight any part of a document, the document is considered relevant and you should then select a so-called "best entry point" (BEP) of the document.

During the relevance assessment of a given topic, all parts of the topic specification should be consulted in the following order of priority: narrative, topic description, and topic title. The narrative should be treated **as the most authoritative description of the user's information need**, and hence it serves as the main point of reference against which relevance should be assessed. In case there is conflicting information between the narrative and other parts of a topic, the information contained in the narrative is decisive. *Note that it is not because that a term listed within the topic is not present in an element that the element is not relevant.* It may be that a component contains some or maybe all the terms, but is irrelevant to the topic of the request. Also, there may be components that contain none of the terms yet are relevant to the topic.

For the CO+S, the topic titles (may) contain structural constraints in the form of XPath expressions. These structural conditions should be ignored during your assessment. This means that you should assess the elements returned for a CO+S topic as whether they satisfy your information need (as specified by the topic) **with respect to the content criterion only**.

¹ The terms document component and XML element are used interchangeably.

You should judge each text fragment on its own merit! That is, a text fragment is still relevant even if it is the twentieth you have seen 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.

4. Using the on-line assessment system (X-Rai)

There is an on-line relevance assessment system (XML Retrieval Assessment Interface) provided at:

<https://inex.lip6.fr/2006/adhoc>

which allows you to view the pooled result set of the topics assigned to you for assessment, to browse the Wikipedia document collection and to record your assessments. Use your INEX username and password to access this system.

The assessment tool works with opera and recent "gecko" browsers: we highly recommend you to use Opera (version 8 or up only; version 9 is recommended) available at <http://www.opera.com>. Other compatible browsers are:

- < **Mozilla** (version 1.7 or up) at <http://www.mozilla.org/products/firefox/>.
- < **Firefox** (version 1 and up) at <http://www.mozilla.org/products/mozilla1.x/>.

Note that **JavaScript must be enabled** for the assessment tool to work and that **the assessment tool is not compatible with Internet Explorer**. Any bug report should be submitted using the project homepage (<https://developer.berlios.de/projects/x-rai/>) using the link in the "Links" menu of the interface (Figure 1).

4.1. Home page

After logging in, you will be presented with the Home page (see Figure 1) 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.

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 1),
 2. A list of menu items, which can be accessed by holding the mouse over the menu label (e.g. "Links" in Figure 1),
 3. The location within X-Rai, where each location step is a hyperlink (in Figure 1, 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 2a). Click on one of these icons to display (or hide):



Information about X-Rai.



Toggle the help

- < The main window.
- < An optional status bar (see Figure 4), displayed only when assessing a pool, i.e. in pool, sub-collection or article view (see relevant sections below) appears at the bottom of the window and shows the number of unknown assessments you have to judge before completing assessing the document (in Figure 4, there is only one unknown assessment).
- < In the status bar, three arrows (←, ↑ and →) may be used to navigate quickly between the elements to be assessed. You may also use the shortcut keys of 1 (left), 2 (up) and 3 (right). The up arrow enables you to move to a level up in the hierarchy, e.g. from an article or a collection part to its innermost enclosing part of the collection (you move in the opposite direction by selecting a sub-collection or an article). The left arrow can be used to go to the previous element to be assessed, while the right arrow to go to the next element to be assessed.

The on-line assessment system provides three main views (Sections 4.2 to 4.4):

1. Pool view,
2. Sub-collection view, and
3. Article view



Figure 2: Home page and menu bar

In the “**Links**” menu

- < **INEX 2006**: link to the official INEX web site.
- < **X-Rai project**: link to the development web site of X-Rai where you can submit bug reports or/and feature requests.
- < **Guide**: the latest version of this assessment guide.

4.2. Pool view

Clicking on a topic ID will display the Pool main page for that topic (see Figure 2a).

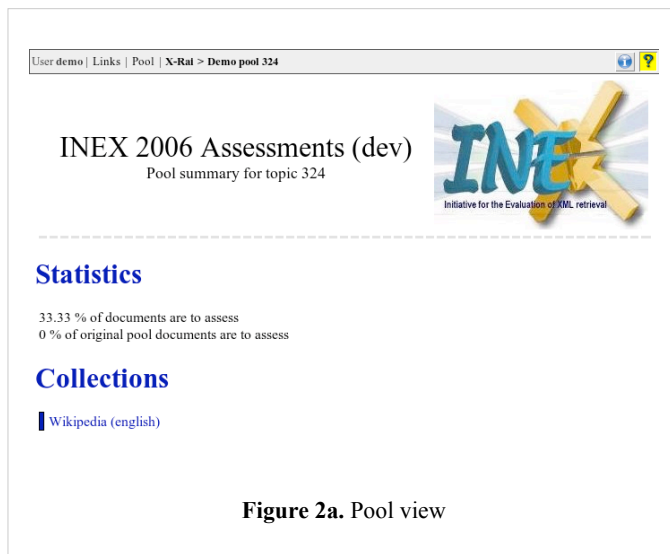


Figure 2a. Pool view

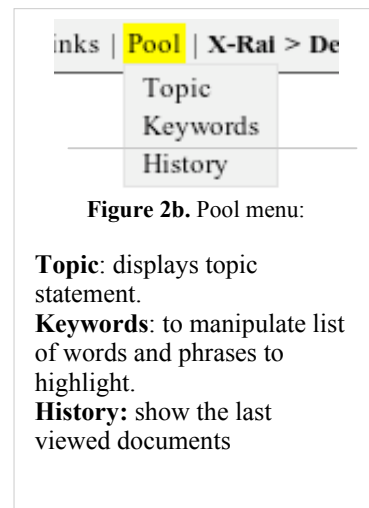


Figure 2b. Pool menu:

- Topic**: displays topic statement.
- Keywords**: to manipulate list of words and phrases to highlight.
- History**: show the last viewed documents

Here, a new menu item, “**Pool**”, appears on the menu bar at the top of the window.

Within the “Pool” menu (Figure 2b), 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.

The “**History**” item allows you to access the list of last viewed documents, which can be useful if you want to go back to a wrongly assessed document.

Under the title “**Collections**” is the list of collections to be assessed. In INEX 2006 (ad hoc task) there is only one such collection, the English Wikipedia 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, no change will occur).

Clicking the hyperlink of “Wikipedia (english)” will take you into the sub-collection view.

4.3. Sub-collection view

The sub-collection views allow you to browse the different sub-collections within the Wikipedia collection. Sub-collections within Wikipedia are based on the alphabetical order, as depicted Figure 3. The first link of the page let you browse the Wikipedia sub-collection starting from "" to "Ali Baba...". This part will then be in turn divided into other sub-collections within the "" to "Ali Baba" range. Eventually, the last sub-collection view will contain a list of Wikipedia documents. Note that this view will show all articles within the collection, and not only those that need to be assessed.

For each possible sub-collection, there is an indication on the number of documents to be assessed in it (if this number is greater than 0), both for documents that were initially in the pool and for documents you choose to assess.

The left or right arrows on the status bar move the focus to the previous or next sub-collection, where there is at least one document to assess. You can also directly click on a link to a sub-collection.

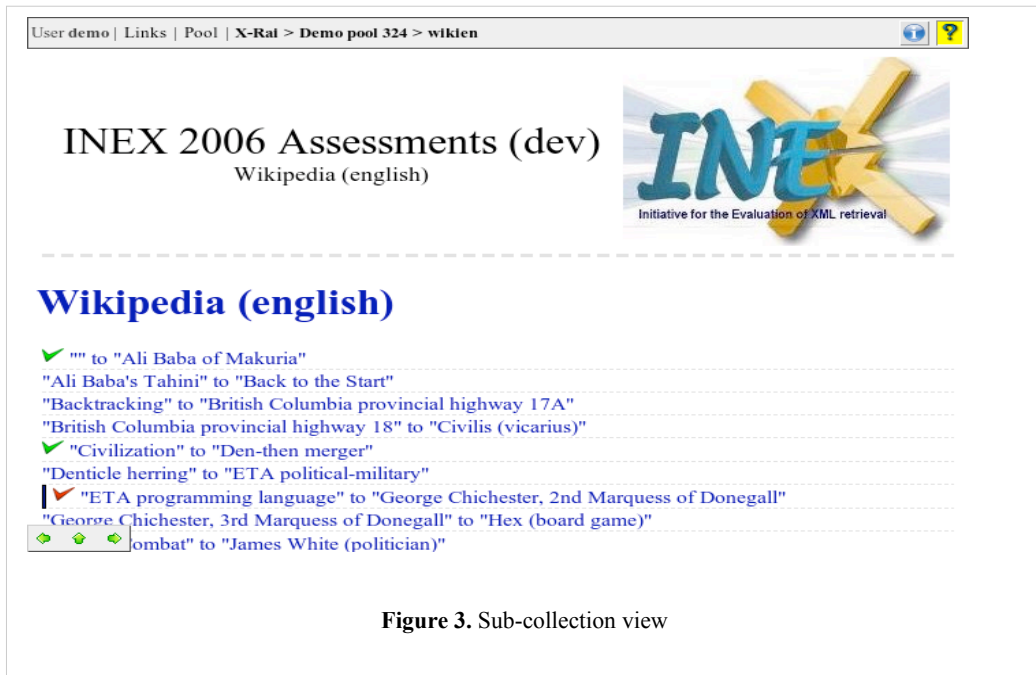


Figure 3. Sub-collection view

4.4. Article view

It is in this article view that elements can be assessed. The article view (see Figure 4) displays all the XML elements of an article together with their content. There are two types of objects within an article view: XML elements and passages. The latter are defined by the assessor while highlighting whereas the former are predefined by the XML file. A highlighted passage in the interface has a yellow background. Note that you should take care of not selecting colours for keyword highlighting too close to the colour X-Rai uses to mark highlighted passages.

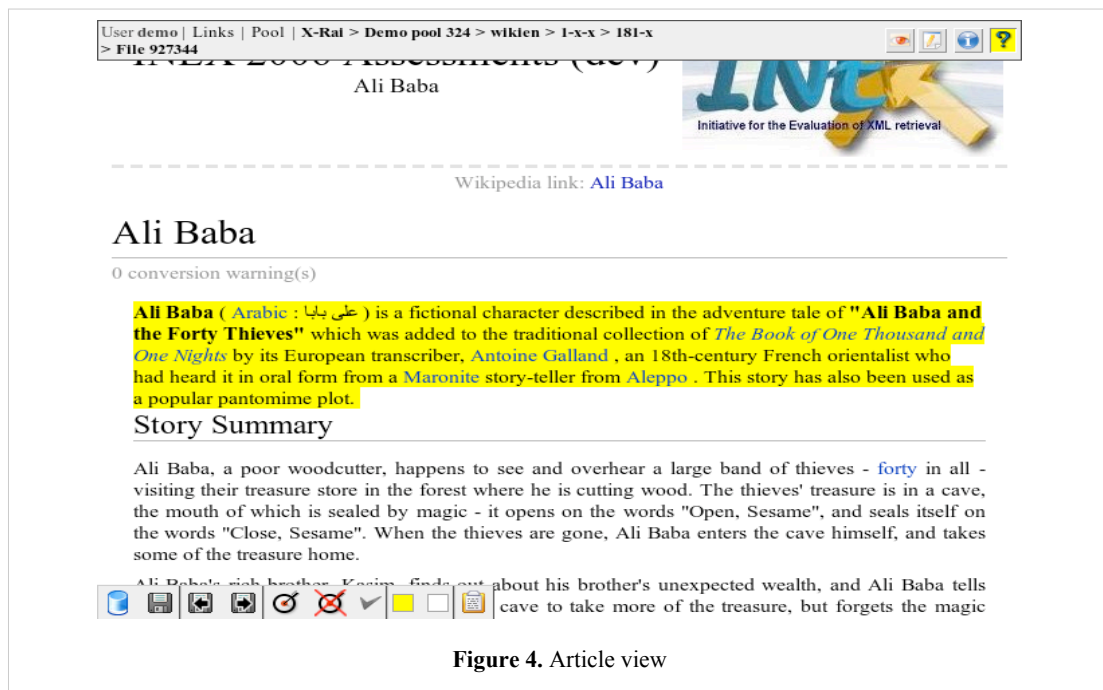


Figure 4. Article view

Highlighting

During the highlight phase, you should identify only relevant (i.e. totally specific) passages by highlighting them. Passages can span over XML element boundaries. The passage limits are predefined by a pre-processing of XML files and correspond “more or less” to sentence boundaries. A consequence of this is that you should highlight the smallest passage that encloses the only relevant information if the predefined boundaries do not correspond exactly to the totally specific fragment.

To highlight a passage, select it with the mouse as you would do in any word processor or text editor, and click on the square with the yellow background (or press “h”).

If you make an error, you can unhighlight it by selecting the non relevant passage and clicking on the square with the white background (or press “u”).



Figure 5. Status bar (article view only)

The disk icon (here disabled): saving your assessments

The disk icon with the left (respectively right) arrow: save (if necessary) and goes to the previous (respectively next) document to assess.

The up arrow allows you to go up to the sub-collection view.

The eye (if applicable): shows or hides the pool elements

The target is used to set the BEP. The stroked target is used to remove the current BEP (if it is already defined for the document).


The mark reflects the status of the document: completely assessed and validated (green), completely assessed but not validated (red), and not completely assessed and not validated (grey). You can validate a document (i.e., mark it as finished) only if the mark is red.

The yellow/white square permits to (un)highlight the selected passage.


The clipboard shows the boundaries of the currently selected passage (as a couple of XPath expressions). This can be useful e.g. to submit

Best Entry Point

Focussed structured document retrieval employs the concept of best entry point (BEP), which is intended to provide optimal starting-point from which users can browse to relevant document components. In INEX, you are requested to indicate one and only one BEP for every document that that has relevant content (that has highlighted passages). No BEP should be defined if the document is not relevant (i.e. does not contain any highlighted passage).

To set the BEP within a document (i.e. to be in the BEP mode), click on the  button (or press b) and then click on the position that you want to set as the BEP of that document. It is not possible to set the BEP at an arbitrarily position within the document. The same constraints to those used for highlighting apply for the BEP. In order to help you to know where the BEP will be located, when the mouse pointer is over a Wikipedia text and that you clicked on the "target button", the BEP symbol should appear at the position it would be set if you have clicked. Also note that there are one and one only BEP per relevant document.

Note that although you can set the BEP at any moment, we recommend that you first highlight and then set the BEP.

To remove any previously set BEP, simply click on  (or press shift+b).















4.6. Saving your assessments

The assessment tool this year does not automatically save the assessments, but you NEED TO SAVE YOUR RELEVANCE ASSESSMENTS by clicking on the disk icon:



The icon is disabled (grey shade) when all assessments are saved.

Be warned that Opera does not provide a way to prevent from exiting a page without saving assessments. PLEASE ONLY USE THE INTERFACE TO NAVIGATE INTO THE SITE as this is the only way to prevent you from leaving a page with non-saved assessment(s).

<i>Icon</i>	<i>Shortcut</i>	<i>Action description</i>
All views within a pool		
	1	Highlight the previous (sub)collection or document to assess.
	2	Go to the container (sub-collection for an article, etc.)
	3	Highlight the next (sub)collection or document to assess
Article view		
	control+s	Save the current assessment
	p	Hide the pool elements
	p	Show the pool elements
	b	Set the BEP
	shift+b	Remove the BEP
shift + 	9	Go to the previous article to assess.
shift + 	0	Go to the next element to assess.
Article view - assessing		
	h	Highlight the currently selected passage.
	u	Unhighlight the currently selected passage
	f	Mark the article as finished
	f	Mark the article as not finished



INEX'06 Het Track Retrieval Task and Result Submission Specification

Ingo Frommholz and Ray Larson
ingo.frommholz@uni-due.de
ray@sims.berkeley.edu
September 26, 2006

Retrieval Tasks

The primary INEX test collection is based on a single DTD. In practical environments, such a restriction will hold in rare cases only. Instead, most XML collections will consist of documents from different sources, and thus with different DTDs or Schemas. In addition, distributed systems (federations or peer-to-peer systems), where each node manages a different type of collection, will need to be searched and the results combined. If there is a semantic diversity between the collections, not every collection is suitable to satisfy the user's information need. On the other hand, querying each collection is expensive w.r.t. communication costs, so a preselection of appropriate collections should be performed. So a heterogeneous collection poses a number of challenges for XML retrieval. The following task definitions try to cope with some of them:

Adhoc CO Task (CO)

Here, content-oriented queries are applied. The systems return a ranked list of documents from all collections.

CAS Task 1 (CAS1)

The system should return only elements specified in `<castitle>`.

CAS Task2 (CAS2)

The system should basically return the elements specified in `<castitle>`, but also similar elements. As an example, `<doctitle>` in ZDNet and `<title>` in other collections are most probably equivalent. The `<description>` in ZDNet, which is the description grabbed from RSS feeds, is similar, but not equivalent, to the `<about>` tag elsewhere. A possible scenario would be a system which likes to present the user only the title and a representative summary of the content so that she could decide if a document is relevant or not without higher cognitive overload (the need for reading the whole article). Furthermore, short comments could be presented here as well.

Resource Selection (RS)

The goal here is to select relevant resources for a given topic. The system should return a ranked list of collections. The scenario is that a system should identify relevant collections beforehand and query them, instead of querying all resources (which might be expensive when it comes to communication or access costs).

Result Submission

For each topic up to 12 runs may be submitted, 3 for each task. The results of one run must be contained in one submission file (e.g. up to 3 files can be submitted for each task). A submission may contain up to 1500 retrieval results for each of the topics.

Submission format

For relevance assessments and the evaluation of the results we require submission files to be in the format described in this section. The submission format for all tasks is defined in the following DTD:


```

<!ELEMENT inex-het-submission (topic-fields, description,
    collections, topic+)>
<!ATTLIST inex-het-submission
    participant-id CDATA #REQUIRED
    run-id CDATA #REQUIRED
    task (CO | CAS1 | CAS2 | RS) #REQUIRED
    query (automatic | manual) #REQUIRED
>
<!ELEMENT topic-fields EMPTY>
<!ATTLIST topic-fields
    title (yes|no) #REQUIRED
    castitle (yes|no) #REQUIRED
    description (yes|no) #REQUIRED
    narrative (yes|no) #REQUIRED
    ontopic_keywords (yes|no) #REQUIRED
>
<!ELEMENT description (#PCDATA)>
<!ELEMENT topic (result*)>
<!ATTLIST topic topic-id CDATA #REQUIRED >
<!ELEMENT collections (collection*)>
<!ELEMENT collection (collectionid, rank?, rsv? )>
<!ELEMENT collectionid (#PCDATA)>
<!ELEMENT result (collectionid, file, path, rank?, rsv?)>
<!ELEMENT file (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT rank (#PCDATA)>
<!ELEMENT rsv (#PCDATA)>

```

The `<collectionid>` element is used to specify the collection a result is in. Valid collection names are:

berkeley	- Berkeley
bibdbpub	- Uni Duisburg BibDB
compuscience	- CompuScience
dblp	- DBLP
hcibib	- HCIBib
qmulcsdbpub	- QMUL BibDB
ieee	- IEEE
zdnart	- ZDNet articles
zdncom	- ZDNet comments
wikipedia	- Wikipedia
lp	- Lonely Planet
idea_eml	- IDEAlliance Extreme Markup Languages
idea_xml1	- IDEAlliance XML 2002 & 2003
idea_xml2	- IDEAlliance XML 2004 & 2005
idea_xmle	- IDEAlliance XML Europe 2003 & 2004
idea_xtech	- IDEAlliance XTech 20058y□

The `<collections>` tag might be empty, except for the RS task (see below).

Submissions for the RS task

Submissions for the resource selection task contain an empty `<topic>` element with just the topic ID, e.g.

```
...
<topic topic-id="id123"/>
<collections>
  <collection>
    <collectionid>hcibib</collectionid>
    <rsv>0.78</rsv>
  </collection>
  <collection>
    <collectionid>ieee</collectionid>
    <rsv>0.71</rsv>
  </collection>
</collections>
...
```

Please use the collection names above for the `<collectionid>` tag.

INEX 2006 Multimedia Track Guidelines

Thijs Westerveld Roelof van Zwol et al.*

August 25, 2006

1 Introduction

Structured document retrieval allows for the retrieval of document fragments, i.e., XML elements, containing relevant information. The main INEX ad-hoc task focuses on text-based XML element retrieval. Although text is dominantly present in most XML document collections, other types of media can also be found in those collections. Existing research on multimedia information retrieval has already shown that it is far from trivial to determine the combined relevance of a document that contains several multimedia objects. The objective of the INEX 2006 multimedia track is to exploit the XML structure that provides a logical level at which multimedia objects are connected, to improve the retrieval performance of an XML-driven multimedia information retrieval system. This document serves as the guidelines for the track. We discuss the track resources, tasks, topic development guidelines, run submission details and assessment procedure.

2 Track resources

The resources used for the multimedia track are based on wikipedia data. For each task, as described in Section 3, all resources can be used. The following six resources are available. Detailed description of each of them and information on how to obtain them are provided at the INEX MM track website at <http://inex.is.informatik.uni-duisburg.de/2006/mmtrack.html>.

Wikipedia Ad Hoc XML collection: This is the same collection that is used for the INEX 2006 Ad Hoc track. The assumption is that a user will be able to see images from the multimedia corpus in-place in the XML fragments when assessing a fragment.

Wikipedia image collection: A subset of images referred to in the Wikipedia Ad Hoc XML collection is chosen to form the Wikipedia image collection. Note that (due to possible copyright issues) not all images referred to in

*Based on prior MM and Ad Hoc guidelines [6] and [1].

the Ad Hoc collection are included in the multimedia corpus. Only the multimedia corpus images are assumed to be available for the user.

Wikipedia image XML collection: This XML collection is specially prepared for the Multimedia track. It consists of XML documents containing image meta-data. See Figure 1 for an example document. The corresponding image is given in Figure 2. Each document contains exactly one image with (often) a short description. This corresponds with the information that is also available on wikipedia, consider for instance: <http://en.wikipedia.org/wiki/Image:AnneFrankHouseAmsterdam.jpg>.

```
<?xml version="1.0"?>
<article>
  <name id="1116948">AnneFrankHouseAmsterdam.jpg</name>
  <image xmlns:xlink="http://www.w3.org/1999/xlink"
    xlink:type="simple" xlink:actuate="onLoad"
    xlink:show="embed"
    xlink:href="../../../Pictures/AnneFrankHouseAmsterdam.jpg">
AnneFrankHouseAmsterdam.jpg</image>
  <text>Anne Frank House - The Achterhuis - Amsterdam.
    Photo taken by
  <wikilink type="internal" parameters="2">
    <wikiparameter number="0">
      <value>User:Rossrs</value>
    </wikiparameter>
    <wikiparameter number="1" last="1">
      <value>Rossrs</value>
    </wikiparameter>
  </wikilink>mid 2002
  <wikitemplate parameters="1">
    <wikiparameter number="0" last="1">
      <value>PD-self</value>
    </wikiparameter>
  </wikitemplate>
  <p />
  <wikilink type="internal" parameters="1">
    <wikiparameter number="0" last="1">
      <value>es:Image:AnneFrankHouseAmsterdam.jpg</value>
    </wikiparameter>
  </wikilink>
  <p />
  <wikilink type="internal" parameters="1">
    <wikiparameter number="0" last="1">
      <value>Category:Building and structure images</value>
    </wikiparameter>
  </wikilink></text>
```

Figure 1: XML document containing meta-data for image: AnneFrankHouseAmsterdam.jpg

Image classification scores: For each image the classification scores for 101 different concepts are derived by UVA [3].



Figure 2: Example image: AnneFrankHouseAmsterdam.jpg

CBIR system: An on-line service to get a ranked list of similar images given a query image (from the collection) is provided by RMIT [2].

Image features: A set of 120D feature vectors, one for each image, is available that has been used to derive the image classification scores. These feature vectors can be used to build a custom CBIR-system, without having to pre-process/access the image collection [4].

Details of these additional sources can be found on the MM track webpages.

3 Task description

The task for the multimedia track is to retrieve relevant information, based on an information need with a (structured) multimedia character. A structured document retrieval approach in that case should be able to combine the relevance of different media types into a single ranking that is presented to the user. The INEX multimedia track differs from other approaches in multimedia information retrieval, like TRECVID and IMAGECLEF, in the sense that it focuses on using the structure of the document to extract, relate and combine the relevance of different multimedia fragments.

For INEX 2006 Multimedia track we define two main tasks:

MM Fragments: Similar to last year's approach [5], the objective of the *MM-fragments* task is to find relevant XML fragments given an multimedia

information need. This task is in essence comparable to the retrieval of XML elements, as defined in the thorough task of the Ad Hoc track. The thorough task returns elements ranked in relevance order (where specificity is rewarded) and overlap is permitted. The main differences with the INEX Ad Hoc track are that all topics in this track ask for multimedia fragments (i.e., fragments containing more than text only) and that the topics may contain visual constraints (see Section 4). The core collection for this task is the Wikipedia Ad Hoc XML collection.

MM Images: Find relevant images given an information need. Here the type of the target element is defined (an image), so basically this is image retrieval, rather than XML element retrieval. Still, the structure of (supporting) documents could be exploited to get to the relevant images. The core collection for this task is the Wikipedia image collection.

All track resources can be used for both tasks.

4 Topic development

4.1 Topic Creation criteria

Creating a set of topics for a test collection requires a balance between competing interests. The performance of retrieval systems varies largely for different topics. This variation is usually greater than the performance variation of different retrieval methods on the same topic. Thus, to judge whether one retrieval strategy is (in general) more effective than another, the retrieval performance must be averaged over a large and diverse set of topics. In addition, the average performance of the retrieval systems on the topics can be neither too good nor too bad as little can be learned about retrieval strategies if systems retrieve no, or only relevant, documents.

When creating topics, a number of factors should be taken into consideration. Topics should:

- be authored by an expert in (or someone familiar with) the subject areas covered by the collection,
- reflect real needs of operational systems,
- represent the type of service an operational system might provide,
- be diverse,
- differ in their coverage, e.g. broad or narrow topic queries,
- be assessed by the topic author.

4.2 Topic Format

The INEX MM track topics are Content Only + Structure (CO+S) topics, like in the Ad Hoc track. While in multimedia the term content often refers to visual content, in INEX it means textual or semantic content of a document part. The term content-only is used within INEX for topics or queries that use no structural hints.

The 2006 CO+S topics consist of the following parts, which are explained in detail below:

<title> in which Content Only (CO) queries are given

<castitle> in which Content And Structure (CAS) queries are given

<description> from which NLP queries are derived

<narrative> in which the definitive definition of relevance and irrelevance are given

<ontopic_keywords> in which terms that are expected in relevant elements are listed

<offtopic_keywords> in which terms that are expected in non-relevant elements are listed

4.2.1 <narrative>

A clear and precise description of the information need is required in order to unambiguously determine whether or not a given element fulfills the given need. In a test collection this description is known as the narrative. It is the only true and accurate interpretation of a user's needs. Precise recording of the narrative is important for scientific repeatability - there must exist, somewhere, a definitive description of what is and is not relevant to the user. To aid this, the <narrative> should explain not only what information is being sought, but also the context and motivation of the information need, i.e., why the information is being sought and what work-task it might help to solve.

Many different queries could be drawn from the <narrative>, and some are better than others. For example, some might contain phrases; some might contain ambiguous words; while some might even contain domain specific terms, structural constraints or visual hints. Regardless of the query, the search engine results are not necessarily relevant. Even though a result might contain search terms from the query, it might not match the explanation given in the <narrative>. Equally, some relevant documents might not be found, but they remain relevant because they are described as so by the <narrative>.

The different CO+S topic parts relate to different scenarios that lead to different types of queries.

4.2.2 <title>

The topic <title> simulates a user who does not know (or does not want to use) the actual structure of the XML documents in a query and who does not have (or want to use) example images or other visual constraints. The query expressed in the topic <title> is therefore a Content Only (CO) query. This profile is likely to fit most users searching XML digital libraries.

4.2.3 <castitle>

Upon discovering their <title> query returned many irrelevant hits, a user might decide to add structural hints (to rewrite as a CAS query). This is similar to a user adding + and - to a web query when too many irrelevant pages are found. At INEX, these added structural constraints (+S) are specified using the formal syntax called NEXI (see the INEX website for the specification) and recorded in the topic <castitle>.

Example Suppose a user wants to find pictures of the Apple II computer. They enter the CO query:

```
Apple II figure
```

but discover that most results are figures of products for the Apple II. They decide to add structural hints:

```
//figure[about(., Apple II)]
```

restricting the results to figure elements only, known to contain the captions of figures.

In the MM track two special types of about clauses are allowed, both specifying visual hints or constraints. The first type was already introduced last year and is used for visual similarity. If a user wants to indicate results should have images similar to a given example image, this can be indicated in an about clause with the keyword *src:*. For example to find images similar to the image of cityscapes similar to the image with identifier 789744, one could type

```
\\image[about(.,cityscape) and about(.,src:789744)]
```

To keep things manageable, only example images from within the MM collection are allowed.

The second type of visual hints is directly related to the classification that is provided as an additional source of information. If a user thinks the results should be of a given concept, this can be indicated with an about clause with the keyword *concept:*. For example, to search for cityscapes one could decide to use the concept building:

```
\\image[about(.,cityscape) and about(.,concept:building)]
```


Terms following the keyword *concept*: are obviously restricted to the 101 concepts for which classification results are provided (cf. the INEX MM track website).

The three different types of about clauses (textual terms, visual examples and visual concepts) can be used in any combination. It is up to the systems how to use, combine or ignore this information; the relevance of a result item does not directly depend on these constraints, but it is decided by manual assessments based on the <narrative>.

The NEXI parser is extended for this purpose and available from the Multimedia track web-site.

4.2.4 <description>

As an alternative to entering queries into search engines, a user might ask a librarian to find the information to satisfy their need. Such a user would give a verbal description to the librarian using a natural language. The NLP track at INEX is examining the ability of a search engine to satisfy the information need given this natural language description (recorded in the topic <description>).

Just as there are many CO queries derivable from the <narrative>, there are many ways to express the need in natural language. However it is expressed, it is important that it matches the <narrative> while at the same time it is not the <narrative>.

4.2.5 <ontopic_keywords> and <offtopic_keywords>

The 2006 CO+S topics contain on-topic and off-topic keywords. These are keywords that are either likely to be found in relevant or irrelevant elements retrieved by the user's query. They are recorded in the <ontopic_keywords> and <offtopic_keywords> topic parts. These keywords are needed for special tests (at INEX 2006) into the possibility of doing automatic assessment.

Example On-topic keywords for the user's information need for pictures of the Apple II computer might be:

```
macintosh ; "personal computer" ; photos ; images ; posters
```

while off-topic keywords might be:

```
fruit ; "New York" ; Beatles ; granny
```

4.3 Procedure for Topic Development

Each participating group will have to submit 6 topics by the 7th July 2006. Submission is done by filling in the Candidate Topic Submission Form on the INEX web site: <http://inex.is.informatik.uni-duisburg.de/2006/> under Tasks/Tracks – MM – Topics.

Each participant is expected to define (at least) two topics for the *MMfragments* task, two topics for the *MMimages* task, and two more topics for a task of choice. We encourage the participants to define more than six topics, to increase the reliability of the results, as argued in Section 4.1

The topic creation process is divided into several steps. When developing a topic, use a print out of the on-line Candidate Topic Form to record all information about the topic you are creating.

Step 1: Initial Topic Statement Create a one or two sentence description of the information you are seeking. This should be a simple description of the information need without regard to retrieval system capabilities or document collection peculiarities. This should be recorded in the Initial Topic Statement field. Record also the context and motivation of the information need, i.e. why the information is being sought. Add to this a description of the work-task, that is, with what task it is to help (e.g. writing an essay on a given topic).

Step 2: Exploration Phase In this step the initial topic statement is used to explore the collection. Obtain an estimate of the number of relevant elements then evaluate whether this topic can be judged consistently. You may use any retrieval engine for this task, including your own or the TopX system (<http://infa05501.ag5.mpi-sb.mpg.de:8080/topx/>), provided through the INEX website. Make sure you select the appropriate collection for each task (Wikipedia Ad Hoc collection for *MMfragments* and wikipedia Image XML collection for *MMimages*).

While exploring the collection make a list of the on-topic and off-topic terms that might be used to distinguish between relevant and irrelevant results retrieved by the search engine.

Alternatively, you may choose to perform a visual exploration. For this you can use the provided content based information retrieval system to perform a similarity search, or the concept classifications to get an impression of the top 100 images for a given topic. Since these systems allow access to the images independent of the wikipedia collection, they are not suitable for the *MMfragments* task. For *MMimages* you can use these visual search options as an alternative for the textual exploration. We ask you to record the identifiers of the relevant images you find. The RMIT CBIR system can be accessed at <http://cuscus.cs.rmit.edu.au/>, the results for the UvA concept classifications are available at <http://inex.is.informatik.uni-duisburg.de/2006/downloads/UvAconcepts>. At the moment of writing it's unclear if the CBIR system will be available on time for topic development. If it's not, please use another search system and remember the CBIR service may be valuable when constructing your runs.

Step 2a: Assess Top 25 Results While you use one or more of the search engines to explore the collection, assess the relevance of retrieved elements use the following working definition: mark it relevant if it would be useful if you

were writing a report on the subject of the topic, or if it contributes toward satisfying your information need. Each result should be judged on its own merits. That is, information is still relevant even if it is the thirtieth time you have seen the same information. It is important that your judgment of relevance is consistent throughout this task. Using the Candidate Topic Submission Form record the number of found relevant elements and the path representing each relevant element. We ask you to look at the top 25 results for at least one search engine. Then if you found fewer than 2 relevant documents in total, or more than 20 using a single search engine, abandon the topic and use a new one. Otherwise, perform a feedback search (see below).

Step 2b: Feedback Search After assessing the top 25 elements, you should have an idea of which terms (if any) could be added to the query to make the query as expressive as possible for the kind of elements you wish to retrieve. You should also have an idea of which terms could be used to disambiguate relevant from irrelevant results and if visual clues in the query.

Use the expanded query and a single search engine of choice (preferably the one that produced the most relevant answers), to retrieve a new list of candidates. Judge the top 100 results (some are already judged), and record the number of relevant results in Candidate Topic Form. Record the expanded query in the title field of the Candidate Topic Submission Form. Now record the on-topic and off-topic terms on the Candidate Topic Submission Form.

Step 3: Write the <narrative> Having judged the top 100 results you should have a clear idea of what makes a component relevant or not. It is important to record this in minute detail as the <narrative> of the topic. The <narrative> is the definitive instruction used to determine relevance during the assessment phase (after runs have been submitted). Record not only what information is being sought, but also what makes it relevant or irrelevant. Also record the context and motivation of the information need. Include the work-task, which is: the form the information will take after having been found (e.g. written report). Make sure your description is exhaustive as there will be several months between topic development and topic assessment.

Step 4 CO+S: Optionally write the <castitle> Optionally re-write the title by adding structural constraints and target elements, visual examples and/or visual concepts. Record this as the <castitle> on the Candidate Topic Submission Form. Also record why you think the structural hints might help in the <narrative>. Please note that we aim at having castitles in most topics. Also note that since the *MMimages* task is a document retrieval task, here the target element should be an article element, thus all NEXI queries in this task should be of the form:

```
\article[X]
```

where X is a predicate using one or more about functions.

Step 5: Write the <description> Write the <description>, the natural language interpretation of the query. Ensure the information need as expressed in the <title>, and <castitle> is also expressed in the <description>. Make sure the <description> does not express any additional information needs.

Step 6: Add ontopic and offtopic keywords Based on the documents you have seen, add a set of ontopic keywords (terms that are likely to be found in relevant documents and elements), and offtopic keywords (terms that are likely to be found in irrelevant documents and elements). The more terms the better.

Step 7: Refining Topic Statements Finalize the topic <title>, <castitle>, <description>, and <narrative>. It is important that these parts all express the same information need; it should be possible to use each part of a topic in a stand-alone fashion (e.g. title for retrieval, description for NLP, etc.). In case of dispute, the <narrative> is the definitive definition of the information need - all assessments are made relative to the <narrative> and the <narrative> alone.

Step 8: Topic Submission Once you are finished, fill out and submit the online Candidate Topic Submission Form on the INEX website <http://inex.is.informatik.uni-duisburg.de/2006/> under Tasks/Tracks Ad hoc Topics. After submitting a topic you will be asked to fill out an online questionnaire (this should take no longer than 5-10 minutes). It is important that this is done as part of the topic submission as the questions relate to the individual topic just submitted and the submission process. This is part of an effort to collect more context for the INEX topics as discussed at the Dagstuhl workshop.

Please make sure you submit all candidate topics no later than the 7th July 2006.

4.4 Topic Selection

From the received candidate topics, the INEX organizers will decide which topics to include in the final set. This is done to ensure inclusion of a broad set of topics. The data obtained from the collection exploration phase is used as part of the topic selection process. The final set of topics will be distributed for use in retrieval and evaluation.

5 Runs

Once the topics are distributed, participants can start working on their runs. Each group can submit up to 6 runs per task. At least one of the runs should be a *title* only run. For the other runs, the participants are free to use any combination of the fields defined for a topic.

For each run we would like to know which sources are used (Ad Hoc Collection, ImgXML collection, visual features, classification data, CBIR system). We would encourage groups to do a baseline run that uses online the <title> part of

the query and no sources of information except for the target collection (image XML collection for *MMimages* task, Ad Hoc Collection for *MMfragments* task).

Runs for the *MMfragments* task can be interpreted as what is known in the INEX Ad Hoc track as THOROUGH, this means a result list may contain overlapping elements. The *MMimages* task is a document retrieval task, the only results allowed there are full documents (ie., articles) from the image XML collection. This means the path of each of the results for this task should be `/article[1]`.

5.1 Submission Format

The submission format for the MM tracks is highly similar to the format for the Ad Hoc tracks. The only changes are in the names of the task and the topic-fields. The submission format for both MM tasks is defined in the following DTD:

```
<!ELEMENT inex-submission (topic-fields, description, collections, topic+)>
<!ATTLIST inex-submission
  participant-id CDATA #REQUIRED
  run-id        CDATA #REQUIRED
  task          (MMfragments|MMimages) #REQUIRED
  query         (automatic|manual) #REQUIRED
  submission-type CDATA #REQUIRED
>
<!ELEMENT topic-fields EMPTY>
<!ATTLIST topic-fields
  title      (yes|no) #REQUIRED
  castitle   (yes|no) #REQUIRED
  description (yes|no) #REQUIRED
  narrative   (yes|no) #REQUIRED
>
<!ELEMENT resources EMPTY>
<!ATTLIST resources
  wikipedia      (yes|no) #REQUIRED
  wikipedia_IMG  (yes|no) #REQUIRED
  UvAfeatures    (yes|no) #REQUIRED
  UvAconcepts    (yes|no) #REQUIRED
  RMIT_GIFT      (yes|no) #REQUIRED
>
<!ELEMENT description (#PCDATA)>
<!ELEMENT topic (result*)>
<!ATTLIST topic topic-id CDATA #REQUIRED >
<!ELEMENT collections (collection+)>
<!ELEMENT collection (#PCDATA)>
<!ELEMENT result (in?,file, path, rank?, rsv?)>
<!ELEMENT in (#PCDATA)>
<!ELEMENT file (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT rank (#PCDATA)>
<!ELEMENT rsv (#PCDATA)>
```

Each submission must contain the participant ID of the submitting institute (available at the INEX web-site <http://inex.is.informatik.uni-duisburg.de/2006/ShowParticipants.html>), a run ID (which must be unique for the submissions sent from one organization – also please use meaningful names as much as possible), the identification of

the task (*MMfragments* or *MMimages*), and the identification of whether the query was constructed automatically or manually from the topic. Furthermore, the used topic fields must be indicated in the `<topic-fields>` tag as well as the used resources. Furthermore each submitted run must contain a description of the retrieval approach applied to generate the search results. A submission contains a number of topics, each identified by its topic ID (as provided with the topics). For compatibility with the heterogeneous collection track, the `<collections>` tag is mandatory. There should be with `<collections>` at least one `<collection>` tag, which should be set to 'wikipedia' for the *MMfragments* task and to 'wikipedia_IMG' for the *MMimages* task. The `<in>` tag is optional for the MM track (`<in>` states from which collection each result comes from). For each task a maximum of 1500 result elements may be included per topic. 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. Here is a sample submission file for the *MMfragments* task TASK:

```
<inex-submission participant-id="12" run-id="VSM_Aggr_06"
  task="MMfragments" query="automatic">
  <topic-fields title="no" castitle="yes" description="no" narrative="no"/>
  <resources wikipedia="yes" wikipedia_IMG="yes" UvAfeatures="no" UvAconcepts="no" RMIT_GIFT="yes"/>
  <description>Using VSM to compute RSV at leaf level combined with
    aggregation at retrieval time, assuming independence and using
    augmentation weight=0.6.</description>
  <collections>
    <collection>wikipedia</collection>
  </collections>
  <topic topic-id="01">
    <result>
      <file>9996</file>
      <path>/article[1]</path>
      <rsv>0.67</rsv>
    </result>
    <result>
      <file>9996</file>
      <path>/article[1]/name[1]</path>
      <rsv>0.1</rsv>
    </result>
    [ ... ]
  </topic>
  <topic topic-id="02">
    [ ... ]
  </topic>
  [ ... ]
</inex-submission>
```

Rank and RSV The rank and rsv elements are provided for submissions based on a retrieval approach producing ranked output. The ranking of the result elements can be described in terms of:

- Rank values, which are consecutive natural numbers, starting with 1. 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. If neither is given, the document order is used.

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. The file names in the Wikipedia collections uniquely define an article, so there is no need for including the directory in which the file resides (in contrast with the earlier IEEE collection). The extension .xml must be left out. Example:

9996

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 ::= ElementName Index
AttributeNode ::= '@' AttributeName
Index ::= '[' integer ']'
```

Example:

```
/article[1]/body[1]/section[2]/p[1]
```

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 “body” element, within which we select the second “section” element, and finally within that element we select the first “p” element. Important: XPath counts elements starting with 1 and takes into account the element type, e.g. if a section had a title and two paragraphs then their paths would be given as: `../title[1]`, `../p[1]` and `../p[2]`.

A result element may then be identified unambiguously using the combination of its file name and element path. Example:

```
<result>
  <in>wikipedia</in>
  <file>9996</file>
  <path>/article[1]/body[1]/section[2]/p[1]</path>
</result>
```

Once again, *MMimages* is a document retrieval task, the only submitted `<path>` allowed for this task is `/article[1]`. In practise we will ignore the submitted `<path>` and only use the `<file>` field of the result (duplicate files in a single run are not allowed).

5.2 Result Submission Procedure

To submit a run, please use the following link: <http://inex.is.informatik.uni-duisburg.de/2006/> Then go to Tasks/Tracks, Multimedia, Submissions. The online submission tool will be available soon.

6 Assessments

To be decided

References

- [1] Birger Larson and Andrew Trotman. Inex 2006 guidelines for topic development. Unpublished document distributed to INEX 2006 participants.
- [2] RMIT University School of Computer Science and Information Technology. Wikipedia cbir system for the multimedia track. <http://www.cs.rmit.edu.au/>.
- [3] C.G.M. Snoek, M. Worring, J.C. van Gemert, J.M. Geusebroek, and A.W.M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *ACM Multimedia Conference*, 2006. under review.
- [4] Jan C. van Gemert, Jan-Mark Geusebroek, Cor J. Veenman, Cees G.M. Snoek, and Arnold W.M. Smeulders. Robust scene categorization by learning image statistics in context. In *CVPR Workshop on Semantic Learning Applications in Multimedia*, New York, USA, June 2006.
- [5] R. van Zwol, G. Kazai, and M. Lalmas. Inex 2005 multimedia track. In *Advances in XML Information Retrieval*, Lecture Notes in Computer Science. Springer, 2006.
- [6] Roelof van Zwol, Mounia Lalmas, and Gabriella Kazai. Inex 2005 multimedia track – working document. Unpublished document distributed to INEX 2005 MM participants.

Entity Ranking – Guidelines

DRAFT v1

Arjen P. de Vries
CWI, Amsterdam, The Netherlands
arjen@acm.org

Nick Craswell
Microsoft Research, Cambridge, UK
nickcr@microsoft.com

May 26, 2006

1 Introduction

Search engines are ever more interested in returning *entities* instead of ‘just’ web pages. INEX has started a pilot track called *Entity Ranking* to provide a forum where researchers may compare and evaluate techniques for engines that return lists of entities. The goal (for this year at least) is not to evaluate how well systems identify instances of entities; the set of entities is assumed known.

2 Data

The track uses the wikipedia data, where we exploit the *category* metadata on pages to construct the entity sets. For example, consider categories ‘Dutch politicians’ or ‘Art museums and galleries’. The entities in such a set are assumed to correspond to those wikipedia pages that are labeled with this category.

Obviously, this is not a perfect solution: many wikipedia pages will not have been categorised, and, pages that do not correspond to the entity set may have been assigned the category because of a different relationship. We expect however that in spite of these issues, the data set provides a sufficiently useful collection as a starting point for the purpose of a pilot track. The challenge is to make use of the rich information from text, structure and links to perform this task.

3 Tasks

3.1 Entity Ranking

The motivation for Entity Ranking is to return entities that satisfy a topic described in natural language text. With ‘Art museums and galleries’ as the entity set and a topic text ‘Im-

pressionist art in the Netherlands', we expect answers like the 'Van Gogh museum' and the 'Kröller-Müller museum'.

3.2 List Completion

If we provide the system with the topic text and a number of examples, the task of *list completion* refers to the problem of completing the partial list of answers. As an example, when ranking 'Countries' with topic text 'European countries where I can pay with Euros', and examples like 'France', 'Germany', 'Spain', then 'The Netherlands' would be a correct completion, but 'Great Britain' would not.

3.3 Associative Ranking

The final task is to generate or complete a second list, that differs from a given list of entities in some attribute. For example, given Dutch impressionist art musea, we could request the system to provide a list of modern art musea in The Netherlands.

4 Topics

Participants are asked to create a small number (say 5) of (partial) entity lists with corresponding topic text. Candidate entities correspond to pages that are assigned to (combinations of) categorised in the Wikipedia corpus.

5 Evaluation

We anticipate a form of participant judging based on voting.

Each participant gets assigned a subset of the topics, and they vote on the correctness of the pooled answers. The answers that get a sufficient number of votes will be assumed the correct ones.

We make the assumption that an entity corresponds to a wikipedia page, so the answer pool correspond to a list of links into the collection. This list will be ordered based on their frequency in the pool. Each participant will be asked to go as deep into the list as they can in a fixed amount of time.

The above procedure is designed for optimal assessment efficiency. We assume that the nature of the task is such that it is feasible to assess answer correctness quickly. We will analyse the data obtained for their accuracy.