Norbert Fuhr
Mounia Lalmas
Andrew Trotman (Eds.)

LNCS 4518

Comparative Evaluation of XML Information Retrieval Systems

5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006
Dagstuhl Castle, Germany, December 2006
Revised and Selected Papers

Springer

Lecture Notes in Computer Science 4518

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Norbert Fuhr Mounia Lalmas
Andrew Trotman (Eds.)

Comparative Evaluation of XML Information Retrieval Systems

5th International Workshop of the Initiative
for the Evaluation of XML Retrieval, INEX 2006
Dagstuhl Castle, Germany, December 17-20, 2006
Revised and Selected Papers

Springer

Volume Editors

Norbert Fuhr
Department of Informatics
University of Duisburg-Essen, 47048 Duisburg, Germany
E-mail: norbert.fuhr@uni-due.de

Mounia Lalmas
Department of Computer Science
Queen Mary, University of London, London, UK
E-mail: mounia@dcs.qmul.ac.uk

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

Library of Congress Control Number: 2007932681

CR Subject Classification (1998): H.3, H.4, H.2

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN 0302-9743
ISBN-10 3-540-73887-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-73887-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12098521 06/3180 5 4 3 2 1 0

Preface

Welcome to the fifth 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.

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 Processing, and Entity Ranking. The Use Case and Entity Ranking tracks were new in 2006. The consolidation of the existing tracks, and the expansion to new areas offered by the two new tracks, allowed INEX to grow in reach.

The aim of the INEX 2006 workshop was 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 concluded the results of this large-scale effort, summarized and addressed the encountered issues and devised a work plan for the future evaluation of XML retrieval systems.

INEX was funded by the DELOS Network of Excellence on Digital Libraries, to which we are very thankful. We gratefully thank the organizers of the various tasks and tracks, who did a superb job. Finally, special thanks go to the participating organizations and individuals for their contributions.

March 2007

Norbert Fuhr
Mounia Lalmas
Andrew Trotman

Organization

Project Leaders

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

Contact Persons

Saadia Malik	University of Duisburg-Essen, Germany
Zoltn Szlavik	Queen Mary, University of London, UK

Wikipedia Document Collection

Ludovic Denoyer	Universite Paris 6, France
Martin Theobald	Max Planck Institute for Informatics, Germany

Use Case Studies

Andrew Trotman	University of Otago, New Zealand
Nils Pharo	Oslo University College, Norway

Topic Format Specification

Andrew Trotman	University of Otago, New Zealand
Birger Larsen	Royal School of LIS, Denmark

Task Description

Jaap Kamps	University of Amsterdam, The Netherlands
Charles Clarke	University of Waterloo, Canada

Online Relevance Assessment Tool

Benjamin Piwowarski	Yahoo! Research Latin America, Chile

Metrics

Gabriella Kazai	Microsoft Research Cambridge, UK
Stephen Robertson	Microsoft Research Cambridge, UK
Paul Ogilvie	Carnegie Mellon University , USA

Relevance Feedback Task

Yosi Mass	IBM Research Lab, Israel
Ralf Schenkel	Max Planck Institute for Informatics, Germany

Natural Query Language Task

Shlomo Geva	Queensland University of Technology, Australia
Xavier Tannier	XEROX, France

Heterogeneous Collection Track

Ingo Frommholz	University of Duisburg-Essen, Germany
Ray Larson	University of California, Berkeley, USA

Interactive Track

Birger Larsen	Royal School of LIS, Denmark
Anastasios Tombros	Queen Mary, University of London, UK
Saadia Malik	University of Duisburg-Essen, Germany

Document Mining Track

Ludovic Denoyer	Universite Paris 6, France
Anne-Marie Vercoustre	Inria-Rocquencourt, France
Patrick Gallinari	Universite Paris 6, France

XML Multimedia Track

Roelof van Zwol	Yahoo! Research, Spain
Thijs Westerveld	CWI, The Netherlands

XML Entity Search Track

Arjen de Vries	CWI, The Netherlands
Nick Craswell	Microsoft Research Cambridge, UK

Table of Contents

Methodology

Ad Hoc Track

Natural Language Processing Track

Heterogeneous Collection Track

Multimedia Track

Interactive Track

Use Case Track

Document Track

Overview of INEX 2006

Saadia Malik[1], Andrew Trotman[2], Mounia Lalmas[3], and Norbert Fuhr[1]

[1] University of Duisburg-Essen, Duisburg, Germany
{malik,fuhr}@is.informatik.uni-duisburg.de
[2] University of Otago, Dunedin, New Zealand
andrew@cs.otago.ac.nz
[3] Queen Mary, University of London, London, UK
mounia@dcs.qmul.ac.uk

Abstract. Since 2002, INEX has been working towards the goal of establishing an infrastructure, in the form of a large XML test collection and appropriate scoring methods, for the evaluation of content-oriented XML retrieval systems. This paper provides an overview of the work carried out as part of INEX 2006.

1 Introduction

The continuous growth in XML[1] information repositories has been matched by increasing efforts in the development of XML retrieval systems (e.g. [1,2]), in large part aiming at supporting content-oriented XML retrieval. These systems exploit the available structural information, as marked up in XML, in documents, in order to return document components – the so-called XML elements – instead of complete documents in response to a user query. This is of particular benefit for information repositories containing long documents or documents covering a wide variety of topics (e.g. books, user manuals, legal documents), where users' effort to locate relevant content can be reduced by directing them to the most relevant parts of these documents. For example, in response to a user's query on a collection of scientific articles marked-up in XML, an XML retrieval system may return a mixture of paragraph, section, article, or other elements that have been estimated as best answers to the user's query. As the number of XML retrieval systems increases, so does the need to evaluate their effectiveness.

The INitiative for the Evaluation of XML retrieval (INEX)[2], which was set up in 2002, established an infrastructure and provided means, in the form of large test collections and appropriate scoring methods, for evaluating how effective content-oriented XML search systems are. As a result of a collaborative effort during the course of 2006, the INEX test collection has been further extended with the addition of the Wikipedia collection, new topics, and new assessments. Using the constructed test collection and the developed set of measures, the retrieval effectiveness of the participants' XML search engines were evaluated and their results compared.

This paper presents an overview of INEX 2006. Section 2 gives a brief summary of this year's participants. Section 3 provides an overview of the test collection. Section 4

[1] http://www.w3.org/XML/
[2] http://inex.is.informatik.uni-duisburg.de/

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 1–11, 2007.

outlines the retrieval tasks in the main ad hoc track. Section 5 reports some statistics of the submitted runs. Section 6 describes the relevance assessment phase. The different measures used to evaluate retrieval performance are described in a separate paper [6]. Section 7 provides a short description of the tracks at INEX 2006.

2 Participating Organizations

In reponse to the call for participation, issued in March 2006, 68 organizations registered. Throughout the year a number of groups dropped out due to resource requirements, while 23 groups joined later during the year. The final 50 active groups along with details of their participation is summarized in Table 1.

3 The Test Collection

Test collections consist of three parts: a set of documents, a set of information needs called topics and a set of relevance assessments listing the relevant documents for each topic. Although a test collection for XML retrieval consists of the same three parts, each component is rather different from its traditional information retrieval counterpart.

In traditional information retrieval test collections, documents are considered as units of unstructured text, queries are generally treated as bags of terms or phrases, and relevance assessments provide judgments whether a document as a whole is relevant to a query or not. XML documents organize their content into smaller, nested structural elements. Each of these elements in the document's hierarchy, along with the document itself (the root of the hierarchy), represent 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, e.g. users can restrict their search to specific structural elements within the collection. Consequently, the relevance assessments for an XML collection must also consider the structural nature of the documents and provide assessments at different levels of the document hierarchy.

3.1 Documents

INEX 2006 uses a different document collection than in previous years [9], made from English documents from the Wikipedia[3]. The collection is made up of the XML full-texts of 659,388 articles of the Wikipedia project, covering a hierarchy of 113,483 categories, and totaling more than 60 Gigabytes (4.6 Gigabytes without images) and 30 million elements. The collection has a structure containing text, more than 300,000 images and some structured parts corresponding to the Wikipedia templates (about 5000 different tags). The collection has a structure similar to the IEEE collection, which was used up to 2005 in INEX. On average an article contains 161.35 XML nodes, where the average depth of an element is 6.72. For a detailed description of the document collection used for the ad hoc and other tracks at INEX 2006 see [3].

[3] http://en.wikipedia.org

Table 1. List of active INEX 2006 participants

Organizations	Submitted topics	Submitted runs	Assessed topics
Utrecht University	6	11	3
University of California, Berkeley	1	2	3
University of Otago	6	0	3
Queensland University of Technology	6	24	3
Queen Mary University of London	4	12	3
Ecoles des Mines de Saint-Etienne	6	9	3
University of Granada	6	2	3
Indian Statistical Institute	0	2	3
University of Tampere	6	0	3
La Trobe University	6	0	3
University of Kaiserslautern,	6	24	3
City University London	6	13	3
RMIT University	6	12	3
IRIT	9	23	3
Max-Planck-Institut fuer Informatik	6	19	3
University of Cambridge	6	0	3
CSIRO	4	8	3
University of Wollongong in Dubai	5	7	3
University of Amsterdam	8	13	3
Fondazione Ugo Bordoni	6	6	3
The Hebrew University of Jerusalem	6	24	3
Royal School of LIS	6	0	3
University of Toronto	6	1	3
Universität Duisburg-Essen	2	0	1
Oslo University College	3	7	3
University of Waterloo	0	0	3
University of Massachusetts Amherst	6	0	3
Kyungpook National University	0	0	3
University of Rostock	6	3	3
LIP6	5	12	3
CWI and University of Twente	6	23	3
University of Helsinki	4	3	3
The Robert Gordon University	6	6	3
IBM Haifa Research Lab	0	18	3
LIPN	1	0	3
CLIPS-IMAG	6	0	3
Université de Saint-Etienne	6	3	3
Justsystem Corporation	0	12	3
University of South-Brittany	0	20	3
Joint Research Centre	0	0	3
University of Minnesota Duluth	6	14	3
Huazhong University of Science & Technology	0	0	3
Dalhousie University	0	0	3
University College of Boras	0	0	3
Université Libre de Bruxelles	0	0	3
Universidad de Chile			
Organizations participated only in XML document mining track			
INRIA			
Western Kentucky University			
University of Wolongong			
Organization participated only in interactive track			
Rutgers University			

3.2 Topics

Querying XML documents with respect to content can be with or without respect to structure. Taking this into account, INEX identifies two types of topics:

Table 2. Statistics on CO+S topics on the INEX 2006 test collection

	CO+S
No. of topics	125
Average length of title (in words)	4.2
Use of boolean operators (and/or) in title	14
Use of (+/-) in title	61
Use of phrases in title	120
Use of boolean operators (and/or) in castitle	65
Use of (+/-) in castitle	49
Use of phrases in castitle	120
Average length of narrative (in words)	94
Average length of topic description (in words)	14
Average length of topic ontopic_keywords (in words)	6

- Content-only (CO) topics are requests that do not include reference to the document structure. They are, in a sense, the traditional topics used in information retrieval test collections. In XML retrieval, the results to such topics can be elements of various complexity, e.g. at different levels of the XML documents' structure.
- Content-and-structure (CAS) topics are requests that contain conditions referring both to content and structure of a document. These conditions may refer to the content of specific elements (e.g. the elements to be returned must contain a section about a particular topic), or may specify the type of the requested answer elements (e.g. sections should be retrieved).

In previous years a distinction was made between CO and CAS topics. Topic were also designed for use in multiple tracks (such as the natural language track and interactive track) and so contained multiple variant queries for each purpose. Since 2006, these have all been combined into a single topic type: the Content Only + Structure (CO+S) topic. All the information needed by the different tasks and tracks are expressed in each topic, but in different parts of that topic.

Topic Format. Topics are made up of several parts; these parts explain the same information need, but for different purposes.

<narrative>:A detailed explanation of the information need and the description of what makes an element relevant or not. The <narrative> explains 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 are made on compliance to the <narrative> alone.

<title>: A short explanation of the information need. It serves as a summary of the content of the user's information need. A word in the <title> can have a $+$ or $-$ prefix, where $+$ is used to emphasize an important concept, and $-$ is used to denote an unwanted concept.

<castitle>: A short explanation of the information need, specifying any structural requirements. As with a topic <title>, a word in the <castitle> can have a $+$ or $-$ prefix,

where + is used to emphasize an important concept, and − is used to denote an unwanted concept. The <castitle> is expressed in the NEXI query language [14].

<description>: A brief description of the information need written in natural language – used in the natural language track. The description is as precise, concise, and as informative as the <title> and <castitle> combined.

<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 <ontopic_keywords> might be: "element, XML".

The DTD of the topics is shown in Figure 1. The attributes of a topic are: topic_id (which in INEX 2006 ranges from 289 to 413) and ct_no, which refers to the candidate topic number (ranging from 1 to 218[4]). An example topic can be seen in Figure 2.

```
<!ELEMENT inex_topic  (title,castitle?,description,
    narrative,ontopic_keywords)>
<!ATTLIST inex_topic
  topic_id     CDATA     #REQUIRED
  ct_no        CDATA     #REQUIRED
>
<!ELEMENT title              (#PCDATA)>
<!ELEMENT castitle           (#PCDATA)>
<!ELEMENT description        (#PCDATA)>
<!ELEMENT narrative          (#PCDATA)>
<!ELEMENT ontopic_keywords   (#PCDATA)>
```

Fig. 1. Topic DTD

Topics were created by participating groups. Each participant was asked to submit up to 6 candidate topics. A detailed guideline was provided to the participants for the topic creation [10]. Several steps were identified for this process: 1) initial topic statement creation, 2) exploration phase, 3) topic refinement, and 4) topic selection. The first three steps were performed by the participants themselves while the selection of topics was performed by the INEX organizers.

During the first step, participants created their initial topic statement. These were treated as a user's description of their information need and were formed without regard to system capabilities or collection peculiarities to avoid artificial or collection biased queries. During the collection exploration phase, participants estimated the number of relevant results to their candidate topics. The TopX XML retrieval system [12] was provided to participants to help with this task. Participants were asked to judge the top 25 retrieved results and record for each found relevant result its file name and its XPath. Those topics having at least 2 relevant results but less than 20 results were to be submitted as candidate topics. In the topic refinement stage, the topics were finalised ensuring coherency and that each part of the topic could be used in stand-alone fashion.

[4] This number is exceeding the total candidate topic number (203) due to the deletion of some candidate topics by topic authors.

```
<inex_topic topic_id="408"  ct_no="202">

<title>
"electroconvulsive therapy" depression
</title>

<castitle>
//*[about(.,"electroconvulsive therapy" depression)]
</castitle>

<description>
Find me information about the treatment of depression with
electroconvulsive therapy
</description>

<narrative>
An old friend of mine suffers from depressions. Usually
medication keeps him well, but  occasionally he is admitted
to hospital with heavy depressions. The treatments often
involve shock therapy (electroconvulsive therapy or ECT).
I am worried about the long term effect of ECT and would
like to learn why passing an electrical current through
the brain can help cure depressions, how it  works, and
if there are any alternatives. Relevant elements will
discuss one of these issues. Elements  that deal with ECT
for other mental illnesses than depresseion are not relevant.
The purpose of the  search is to find information that will
make me better capable of understanding my friends illness.
</narrative>

<ontopic_keywords>
ECT, electroshock, "induced convulsion",  seizure
</ontopic_keywords>

</inex_topic>
```

Fig. 2. A CO+S topic from the INEX 2006 test collection

After the completion of the first three stages, topics were submitted to INEX. A total of 203 candidate topics were received, of which 125 topics were selected. The topic selection was based on a combination of criteria such as 1) balancing the number of topics across all participants, 2) eliminating topics that were considered too ambiguous or too difficult to judge, 3) uniqueness of topics, 4) considering their suitability to the different tracks, and 5) syntactic correctness.

4 Retrieval Tasks

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 literature [15], ad hoc

retrieval is described as a simulation of how a library might be used and involves the searching of a static set of documents using a new set of topics. Here the collection consists of XML documents composed of different granularities of nested XML elements, each of which represents a possible unit of retrieval. The user's query may also contain structural constraints or hints in addition to the content conditions. In addition, the output of an XML retrieval system may follow the traditional ranked list presentation, or may extend to non-linear forms, such as grouping of elements per document.

Within the ad hoc XML retrieval task, four sub-tasks were defined based on the different assumptions regarding a search system's output and learning aims.

4.1 Thorough Task

The core system task underlying most XML retrieval strategies is the ability to estimate the relevance of retrievable elements in the collection. Hence, the thorough task 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.

The aims for this task included establishing: How good systems are at estimating the relevance of XML elements, how well systems can locate all the relevant elements in the collection, and how much structural constraints improve retrieval.

4.2 Focused Task

The scenario underlying this task is the return, to the user, of a ranked list of elements for the topic of request. The task requires systems to find the most focused elements that satisfy an information need, without returning "overlapping" elements (e.g. a paragraph and its container section). That is, for a given topic, elements in the result list may not contain text already contained in previous element. The task is similar to the thorough task in that it requires a ranking of XML elements, but here systems are required not only to estimate the relevance of elements, but also to decide which elements are the most focused non-overlapping.

The learning aims for this task include establishing: how the focused task differs from the thorough task, if the focused task can be reduced to a straightforward filter on the thorough task, which techniques are effective at early ranks, and how structural constraints help retrieval.

4.3 Relevant in Context Task

The scenario underlying this task is the return of relevant information (captured by a set of elements) within the context of the full article. A result, an article devoted to the topic of request, will contain a lot of relevant information across many elements. The task requires systems to find a set of elements that corresponds to (all) relevant information in each article. The set of result elements should not contain overlaps.

The learning aims for this task include establishing: how the relevant in context task differs from the thorough and focused tasks, which techniques are effective at locating relevance within an article, and how structural constraints help retrieval.

4.4 Best in Context Task

The scenario underlying this task is finding the best entry point from which to start reading a relevant document. Even a document completely devoted to the topic of request will only have one best starting point to read, even if this is the start of the document. This task requires systems to find the XML elements that correspond to these best entry points.

The learning aims for this task include establishing: how the best in context task differs from the relevant in context task, how best entry points relate to the relevance of elements, and how structural constraints help retrieval.

5 Submissions

Participating organizations evaluated the 125 INEX 2006 topics against the Wikipedia document collection and produced an ordered list of XML elements as the retrieval results for each topic. Participants could use either the <title> or <castitle> of the CO+S topics. The top 1500 elements of each topic's retrieval results were then submitted to INEX. For each topic, around 500 articles along with their elements were pooled from all the submissions in a round robin fashion for assessment. Table 3 shows the pooling effect on the CO+S topics.

Table 3. Pooling effect for CO+S topics

	CO+S topics
number of documents submitted	126111
number of documents in pools	63684
number of elements submitted	281761
number of elements in pools	137559

Table 4. Number of runs submitted to the four ad hoc tasks

Tasks	runs
Thorough	106
Focused	85
Relevant in context	65
Best in context	77

6 Assessments

Relevance in INEX is defined according to the notion of specificity, which is the extent to which an element focuses on the topic. Previously, INEX used a more complex

definition of relevance but a number of studies showed that specificity alone was sufficient to determine an unambigious rank order of search systems with respect to their effectiveness (see, for example, [11]).

The specificity of an element is determined by an assessor using the highlighting method introduced at INEX 2005. In this approach the specificity of any (partially highlighted) elements can be calculated automatically as some function of the contained relevant and irrelevant content (for example the ratio of one to the other). Specificity is, thus, measured on a continuous scale in the range $[0, 1]$, where 1 represents a fully specific (relevant) element and 0 a non relevant element.

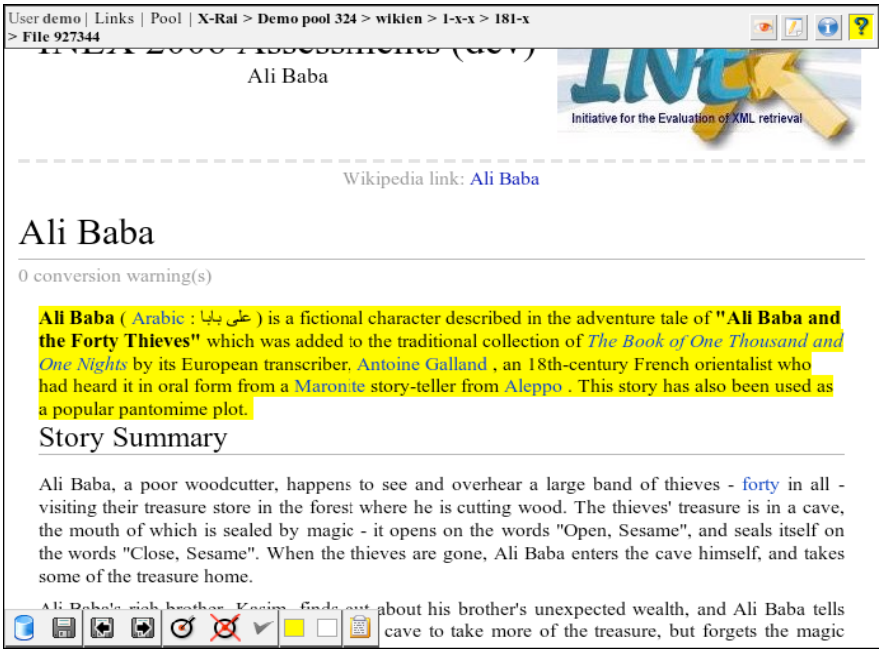


Fig. 3. X-RAI Article view

Assessment was done using an online assessment tool developed specifically for this purpose (see Figure 3). A relevance assessment guide [8] explaining how to assess relevance was distributed to the participants. In short, the assessors had only to highlight relevant text fragments from articles identified as candidates using the pooling strategy. These highlighted passages were then automatically converted into element specificity scores. Assessors were also asked to identify best entry points, one per relevant article.

7 INEX 2006 Tracks

In addition to the main ad hoc track, six research tracks were included, each studying a different aspect of XML information retrieval: Interactive, Relevance Feedback,

Heterogeneous, Natural Language Processing, Multimedia, and Document Mining. Two new tracks were added in 2006: Use Case and Entity ranking.

In its fourth year, the **Interactive Track** (iTrack) put emphasis on comparing XML element retrieval with passage retrieval, and on investigating differences between multiple dimensions of the search tasks. This year the track required substantial work and data collection, so was not completed before the INEX 2006 workshop. The track continues into 2007 [7].

The **Relevance Feedback track** investigated approaches to relevanc feedback that considered the structural hints. To limit the number of submissions a subset of ad hoc track tasks were chosen for participants to test their algorithms. These include the thorough task with CO and CAS topics. The reported evaluation score for each relevance feedback submission measures the relative and absolute improvement of the relevance feedback run over the original base run and the significance level under the t-test and the Wilxocon signed-rank test.

The **Heterogeneous Collection track** was setup to cope with the challenges posed by heterogenous collections, which are syntactic (collections based on different DTDs), semantic (collections covering diverse topics) and genre (different document types) in heterogeneity. This year the track focused on finalising the heterogeneous collection and on topic definition. Based on this, the track will continue in 2007 with the evaluation of submitted runs. This year's track details can be found in [5].

The **Natural Language Processing (NLP) track** focused on whether it is possible to express topics in natural language, to be then used as a basis for retrieval. Two tasks were defined NLQ2NEXI and NLQ. NLQ2NEXI requires the translation of a natural language query, provided in the <description> element of a topic, into a formal INEX <castitle> element. The NLQ task has no restrictions on the use of any NLP technique to interpret the queries as they appear in the <description> element of a topic.The objective is not only to compare between different NLP based systems, but to also compare the results obtained with natural language queries with the results obtained with NEXI queries by any other system in the ad hoc track. During the topic creation stage, it was ensured that the description component of the topics were equivalent in meanings to their corresponding NEXI title, so it was possible to re-use the same topics, relevance assessments and evaluation procedures as in the ad hoc track. The descriptions were used as input to natural language processing tools, which would process them into representations suitable for XML search engines.

The main objective of the **Multimedia track** was to provide an evaluation platform for structured document access systems that do not only include text in the retrieval process, but also other types of media, such as images, speech, and video. Full details of the track can be found in [16].

The aim of the **Document Mining track**, run in collaboration with the PASCAL network of Excellence[5], was to develop machine learning methods for structured data mining and to evaluate these methods for XML document mining tasks. Full details of the track can be found in [4].

The aim of the **Use Case track** was to identify the potential users, scenarios and use-cases of XML retrieval systems. As a result, commercial XML search engines have

[5] http://www.pascal-network.org/

now been identified. XML (and other semi-structured formats) are being used behind the scenes in some on-line search engines without user knowledge. Book search engines that follow a thorough retrieval strategy have also been identified [13].

The aim of the **Entity Ranking track** was to examine list completion and associative ranking. In the former, entities of the same kind as those in a given list were to be extracted from the document collection. In the latter, a similar list to a given list, but on a different topic, was to be extracted from the document collection. Guidelines were drafted, a wide range of potential participants actively approached us, but this did not materialize into sufficient support to run a full track. Given the interest in the track at the INEX workshop, the track will continue in 2007.

References

1. Baeza-Yates, R., Fuhr, N., Maarek, Y.: XML and information retrieval, SIGIR workshop SIGIR Forum, vol. 36(2) (2002)
2. Blanken, H., Grabs, T., Schek, H.-J., Schenkel, R., Weikum, G. (eds.): Intelligent Search on XML Data, Applications, Languages, Models, Implementations, and Benchmarks (2003)
3. Denoyer, L., Gallinari, P.: Wikipedia XML corpus at INEX 2006. In: INEX 2006 Proceedings (2007)
4. Denoyer, L., Gallinari, P.: Report on the XML Mining Track at INEX 2005 and INEX 2006 - Categorization and Clustering of XML Documents. In: INEX 2006 Proceedings (2007)
5. Frommholz, I., Larson, R.: The heterogeneous collection track at INEX 2006. In: INEX 2006 Proceedings (2007)
6. Lalmas, M., Kazai, G., Kamps, J., Pehcevski, J., Piwowarski, B., Robertson, S.: INEX 2006 Evaluation Measures. In: INEX 2006 Proceedings (2007)
7. Larsen, B., Malik, S., Tombros, A.: The interactive track at INEX 2006. In: INEX 2006 Proceedings (2007)
8. Lalmas, M., Piwowarski, B.: INEX 2006 relevance assessment guide. In: INEX 2006 Preroceedings (2006)
9. Lalmas, M., Tombros, A.: INEX 2002 - 2006: Understanding XML Retrieval Evaluation. In: DELOS Conference on Digital Libraries, Tirrenia, Pisa, Italy (2007)
10. Larsen, B., Trotman, A.: INEX 2006 guidelines for topic development. In: INEX 2006 Preproceedings (2006)
11. Ogilvie, P., Lalmas, M.: Investigating the exhaustivity dimension in content-oriented XML element retrieval evaluation. In: CIKM (2006)
12. Theobald, M., Schenkel, R., Weikum, G.: An efficient and versatile query engine for topx search. In: VLDB, pp. 625–636. ACM, New York (2005)
13. Trotman, A., Pharo, N., Lehtonen, M.: XML-IR users and use cases. In: INEX 2006 Proceedings (2007)
14. Trotman, A., Sigurbjornsson, B.: Narrowed extended XPATH I (NEXI). In: INEX 2004 Proceedings (2005)
15. Voorhees, E., Harman, D. (eds.): The Tenth Text REtrieval Conference (TREC 2001) (2001)
16. Westerveld, T., van Zwol, R.: The INEX 2006 multimedia track. In: INEX 2006 Proceedings (2007)

The Wikipedia XML Corpus

Ludovic Denoyer and Patrick Gallinari

Laboratoire d'Informatique de Paris 6
8 rue du capitaine Scott
75015 Paris

{ludovic.denoyer,patrick.gallinari}@lip6.fr
http://www-connex.lip6.fr/denoyer/wikipediaXML

Abstract. This article presents the general Wikipedia XML Collection developped for Structured Information Retrieval and Structured Machine Learning. This collection has been built from the Wikipedia Enclyclopedia. We detail particularly here which parts of this collection have been used during INEX 2006 for the Ad-hoc track and for the XML Mining track. Note that other tracks of INEX - multimedia track for example - have also been based on this collection.

1 Introduction

Wikipedia[1] is a well known 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 2006[2] and the XML Document Mining Challenge[3]. The article provides a description of the corpus.

The collections are downloadable on the website:

– *http://www-connex.lip6.fr/∼denoyer/wikipediaXML*

[1] http://www.wikipedia.org
[2] http://inex.is.informatik.uni-duisburg.de/2006
[3] http://xmlmining.lip6.fr

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 12–19, 2007.

2 Description of the Corpus

The corpus is composed of 8 main collections corresponding to 8 different languages[4] : 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.

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.

Table 1. General statistics about the *Main Collections*

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

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,etc.*) 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,etc.*) 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,etc.

[4] Some additional languages will be added during the next months.

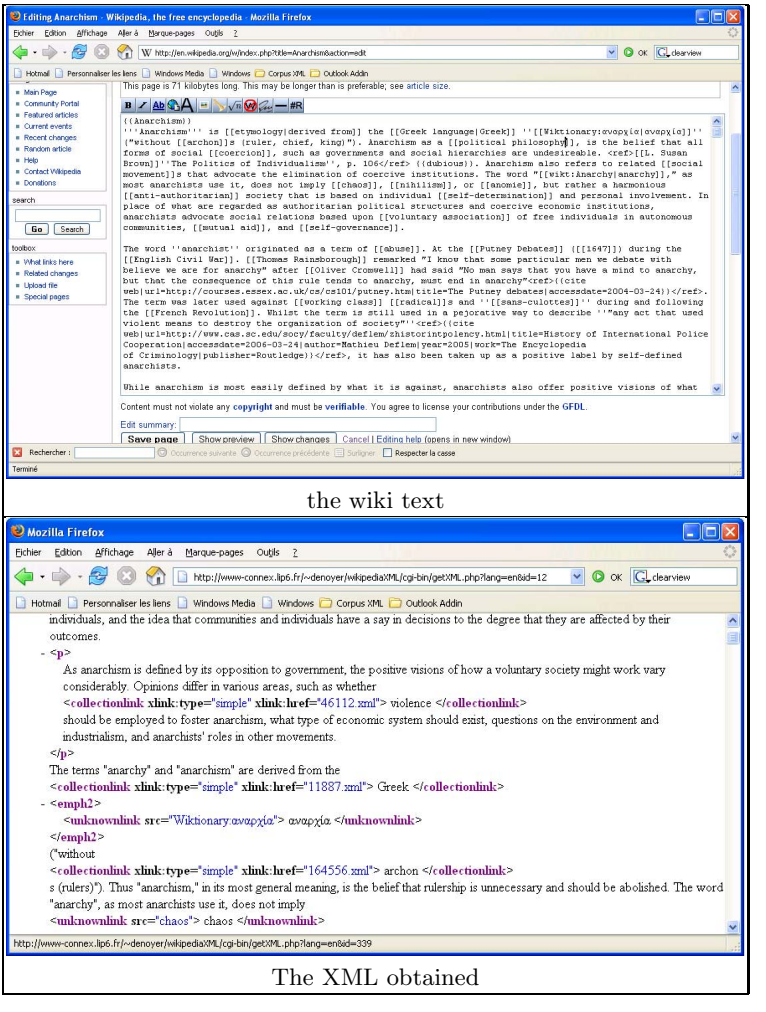


Fig. 1. Example of wiki → XML transformation for the *Anarchy* article (*12.xml*)

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. An example of template is given in figure 2.

The Table 2 gives a summary of most frequent tags of the english collection.

DTD of the collection. Due to the irregularities of the structures of the documents, it is not possible to build a relevant DTD for the wikipedia XML corpus.

Table 2. Distribution and description of the main different XML node labels in the collection

Tag	Number of XML nodes	Description
collectionlink	≈ 17 millions	Hyperlink to a document of the collection
unknownlink	≈ 4 millions	Hyperlink to a document that does not exist in wikipedia
outsidelink	≈ 850,000	Hyperlink to a website
wikipedialink	≈ 850,000	Hyperlink to a wikipedia page (which is not in the collection)
languagelink	≈ 800,000	Hyperlink to the same page in an other laguage
emph2	≈ 2 millions	Emphasis level 2
emph3	≈ 1.5 millions	Emphasis level 3
emph4	≈ 1,000	Emphasis level 4
emph5	≈ 81,000	Emphasis level 5
table	≈ 92,000	Table
row	≈ 1 millions	Row of a table
cell	≈ 3.7 millions	Cell of a table
template	≈ 2.5 millions	Template tags
title	≈ 1.6 millions	Title of articles
p	≈ 2.8 millions	Paragraph
item	≈ 5.6 millions	Item of a list or enumeration
....	....	...

We give in Figure 3 an idea of the schema underlying the collection using a graphical representation of the tags inclusion.

General statistics about the Wikipedia XML collection are given in table 3

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:

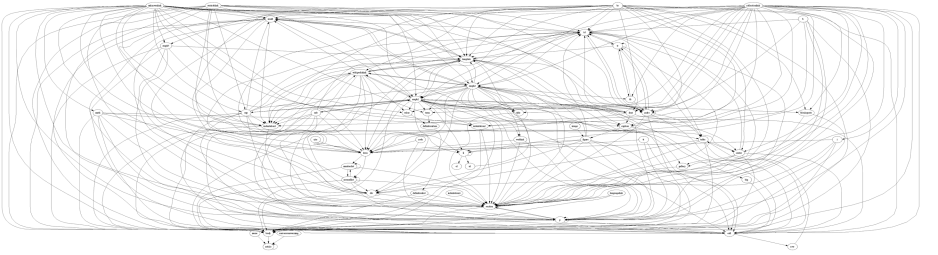
- the hierarchy of categories (file : categories_hcategories.csv)
- the categories of each articles (file : categories_categories.csv)
- the categories names (file : categories_name.csv)

Table 3. Statistics about the structure of the documents from the *Main Collections*

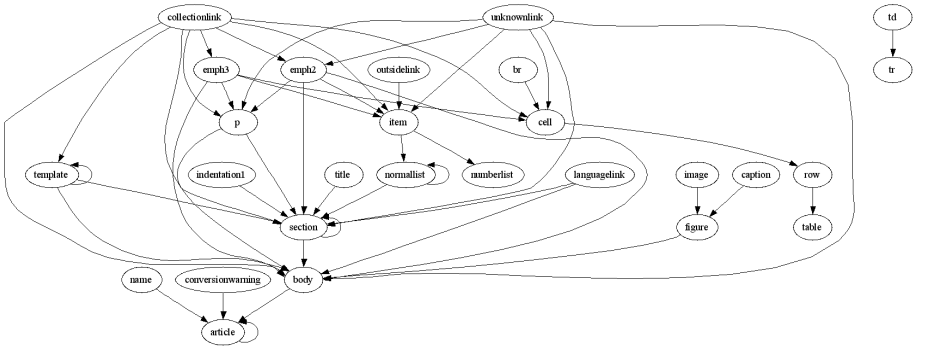
Language	Mean size of document (bytes)	Mean Document Depth	Number of Nodes/Document
English	7,261	6.72	161.35
French	6,902	7.07	175.54
German	7,106	6.76	151.99
Dutch	5,092	6.41	122.8
Spanish	6,669	6.65	165.74
Chinese	6,664	6.91	179.23
Arabian	4,826	5.85	182.1
Japanese	7,973	7.1	94.96

Table 4 gives statistics about the categories.

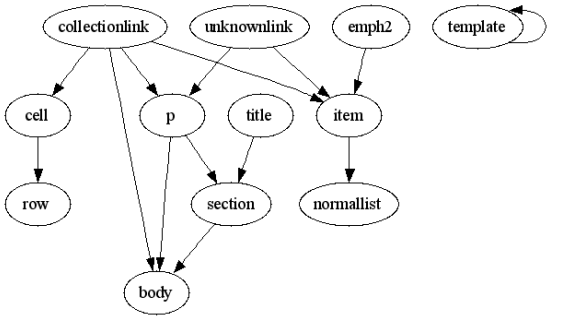
Fig. 2. Example of template conversion. A template (wiki) is converted using the *template_...* tags (xml). It correspond to a structured information of a wikipedia article (html).



N = 1,000



N = 100,000



N = 1,000,000

Fig. 3. This figure shows the schema of the documents (like a DTD). In this graph, each node corresponds to a possible XML node label. Two nodes (n_1, n_2) are connected if there exist at least N XML nodes with tag n_2 that have a child with tag n_1 in the corpus. For example, in the graph where N=1,000,000, the edge between *section* and *article* means that at least 1 million XML nodes with label *section* have a parent with label *article*.

Table 4. Statistics about the categories of the *Main Collections*

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

3 INEX 2006 Collections

3.1 Adhoc Collection

The collection used of the adhoc track during INEX 2006 is based on the english version of the wikipedia XML. Table 5 gives some general statistics over this collection and the Figure 4 gives the distribution of the documents with respect to their number of nodes.

Table 5. Statistics on the INEX 2006 AdHoc Corpus

Number of documents	659,388
Number of elements	$\approx$ 52 millions
Size of the vocabulary	$\approx$ 2 millions (depending on the preprocessing)
Number of tags	$\approx$ 1,200

We do not detail here the description of the assesments made by the participants of INEX 2006.

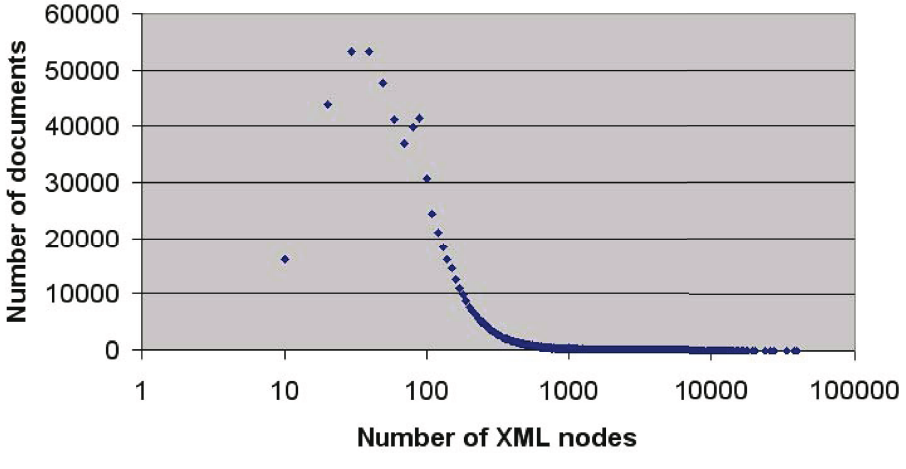


Fig. 4. Distribution of the documents wrt the number of doxels

Table 6. Statistics about the *XML Document Mining Challenge Collection* (*Single-Label Categorization Collection*)

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 *documents* contains all the corresponding articles. The directory *relfiles* contains one file per category giving the id of the documents that belongs to this category[5].

3.2 XML Mining Track Collection

We provide a specific collection where each document belongs to **exactly** one category. This collection was used for the last yerar of the XML Mining Track. 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 6). This collection is aimed at categorization/clustering benchmark.

4 Conclusion

This article report describes the main XML collections based on Wikipedia and developed for Structured Information Retrieval, Structured Machine Learning and Natural Language processing. Then, we detail the collection used during INEX 2006 for both the general adhoc retrieval track and the XML Mining track. Note that there exist some other corpus based on wikipedia XML for different tasks : Natural Language Processing, linguistic annotation, Question answering,etc.

Acknowlegment

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 article.

INEX 2006 Evaluation Measures

Mounia Lalmas[1], Gabriella Kazai[2], Jaap Kamps[3], Jovan Pehcevski[4], Benjamin Piwowarski[5], and Stephen Robertson[2]

[1] Queen Mary, University of London, United Kingdom
mounia@dcs.qmul.ac.uk
[2] Microsoft Research Cambridge, United Kingdom
{gabkaz,ser}@microsoft.com
[3] University of Amsterdam, The Netherlands
kamps@science.uva.nl
[4] INRIA Rocquencourt, France
Jovan.Pehcevski@inria.fr
[5] Yahoo! Research Latin America, Chile
bpiwowar@yahoo-inc.com

Abstract. This paper describes the official measures of retrieval effectiveness employed at the ad hoc track of INEX 2006.

1 Introduction

Since its launch in 2002, INEX has been challenged by the issue of how to measure an XML retrieval system's effectiveness. The main complication comes from how to consider the dependency between elements when evaluating effectiveness.

As discussed in Section 2, the ad hoc track at INEX 2006 has four retrieval tasks, namely focused task, thorough task, relevant in context task, and best in context task. INEX 2006 uses various sets of measures to evaluate these tasks:

- The XCG measures introduced at INEX 2005 [4] were used to evaluate the thorough and the focused retrieval tasks (Sections 4 and 5, respectively).
- A new generalized precision measure was introduced to evaluate the relevant in context retrieval task (Section 6). This measure is based directly on the text highlighted by the assessors, just as the HiXEval measures [9].
- A distance measure, BEPD, was defined to evaluate the best in context retrieval (Section 7).
- The EPRUM measures originally defined in [10] were adapted to also evaluate the best in context retrieval task (Section 7).

This paper is organized as follows. In Section 2, we describe the INEX 2006 ad hoc retrieval tasks, including their motivations. In Section 3, we describe how relevance is defined in INEX 2006. The evaluations of each task are described in the next four sections (Sections 4 to 7). We finish the paper with some discussions.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 20–34, 2007.

2 Ad Hoc Retrieval Tasks

The main INEX activity is the ad hoc retrieval task, where the collection consists of XML documents, composed of different granularity of nested XML elements, each of which represents a possible unit of retrieval. A major departure from traditional information retrieval is that XML retrieval systems need not only score elements with respect to their relevance to a query, but also determine the appropriate level of element granularity to return to users. The user's query may also contain structural constraints or hints in addition to the content conditions. In addition, the output of an XML retrieval system may follow the traditional ranked list presentation, or may extend to non-linear forms, such as grouping of elements per document.

Up to 2004, ad hoc retrieval was defined as the general task of returning, instead of whole documents, those XML elements that are most relevant to the user's query. In other words, systems should return elements that contain as much relevant information and as little irrelevant information as possible. Within this general task, several sub-tasks were defined, where the main difference was the treatment of the structural constraints.

However, within this general task, the actual relationship between retrieved elements was not considered, and many systems returned overlapping elements (e.g. nested elements). What most systems did was to estimate the relevance of XML elements, which is different to identifying the most relevant elements. This had very strong implications with respect to measuring effectiveness, where approaches that attempted to identify the most relevant elements, and to return only those, performed poorly. As a result, the focused task was defined in 2005, intended for approaches aiming at targeting the appropriate level of granularity of relevant content that should be returned to the user for a given topic. The aim was for systems to find the most relevant element on a path within a given document containing relevant information and return to the user only this most appropriate unit of retrieval. Returning overlapping elements was not permitted. The INEX ad hoc general task, as carried out by most systems up to 2004, was renamed in 2005 as the thorough task.

Within the focused and thorough tasks, the output of XML retrieval systems was assumed to be a ranked list of XML elements, ordered by their presumed relevance to the query. User studies [11] suggested that users were expecting to see returned elements grouped per document, and to have access to the overall context of an element. The fetch & browse task was introduced in 2005 for this reason. The aim was to first identify relevant documents (the fetching phase), and then to identify the most relevant elements within the fetched documents (the browsing phase). In 2005, no explicit constraints were given regarding whether returning overlapping elements within a document was allowed. The rationale was that there should be a combination of how many documents to return, and within each document, how many relevant elements to return.

In 2006, the same task, renamed the relevant in context task, required systems to return for each document an unranked set of non-overlapping elements,

covering the relevant material in the document. These elements could be shown to the users, for example, as highlighted text, or through the use of a heat-map.

In addition, a new task was introduced in 2006, the best in context task, where the aim was to find the best entry point, here a single element, for starting to read documents with relevant information. This new task can be viewed as an extreme case of the fetch & browse approach, where only one element is returned per document.

To summarize, INEX 2006 investigated the following four ad hoc retrieval tasks, defined as follows [1]:

- Thorough: This task asks systems to estimate the relevance of all XML elements in the searched collection and return a ranked list of the top 1500 elements.
- Focused: This task asks systems to return a ranked list of the most focused XML elements, where result elements should not overlap (e.g. a paragraph and its container section should not both be returned). Here systems are forced to choose from overlapping relevant elements those that represent the most appropriate units of retrieval.
- Relevant in context: This task asks systems to return to the user the most focused, relevant XML elements clustered by the unit of the document that they are contained within. An alternative way to phrase the task is to return documents with the most focused, relevant elements indicated (e.g. highlighted) within.
- Best in context: This task asks systems to return a single best entry point to the user per relevant document.

For all tasks, systems could use the title field of the topics (content-only topics) or the castitle of the topics (content-and-structure topics) - see [8] for description of the INEX 2006 topics.

3 Relevance Assessments

In INEX 2006, relevance assessments were obtained by assessors highlighting relevant text fragments in the documents, which correspond to wikipedia articles (see [8] for description of the document collection). XML elements that contained some highlighted text were then considered as relevant (to varying degree). A default assumption here is that if an XML element is relevant (to some degree), then its ascendant elements will all be relevant (to varying degrees) due to the subsumption of the descendant elements' content. For each relevant XML element, the size of the contained highlighted text fragment (in number of characters) is recorded as well as the total size of the element (again, in number of characters). These two statistics form the basis of calculating an XML element's relevance score, which in 2006 corresponds to its specificity score [7].

The specificity score, $spec(e_i) \in [0, 1]$, of an element e_i is calculated as the ratio of the number of highlighted characters contained within the XML element,

$rsize(e_i)$, to the total number of characters contained by the element, $size(e_i)$:

$$spec(e_i) = \frac{rsize(e_i)}{size(e_i)} \quad (1)$$

4 Evaluation of the Thorough Task

4.1 Assumptions

This task is based on the assumption that all XML elements of a searched collection can be ranked by their relevance to a given topic. The task of a system here is then to return a ranked list of the top 1500 relevant XML elements, in descreasing order of relevance. No assumptions are made regarding the presentation of the results to the user: the output of a system here can simply be considered as an intermediate stage, which may then be processed for displaying to the user (e.g. filtered, clustered, etc.). The goal of this task is to test a system's ability to produce the correct ranking. Issues, like overlap (e.g. when a paragraph and its container section are both returned) are ignored during the evaluation of this task.

4.2 Evaluation Measures

Two indicators of system performance were employed in the evaluation of the thorough task: effort-precision/gain-recall (ep/gr) graph and mean average effort-precision ($MAep$). These are both members of the eXtended Cumulated Gain (XCG) measures [4], which are extensions of the Cumulated Gain based measures [3]. These were developed specifically for graded (non-binary) relevance values and with the aim to allow information retrieval systems to be credited according to the retrieved documents' degree of relevance.

From the family of XCG measures, ep/gr and $MAep$ were selected as they provide an overall picture of retrieval effectiveness across the complete range of recall. The motivation for this choice is the recall-oriented nature of the task, e.g. rank all elements of the collection and return the top 1500 results. $MAep$ summarizes retrieval effectiveness into a single number, while an ep/gr graph allows for a more detailed view, ploting ep at 100 recall points $\{0.01, 0.02, ..., 1\}$.

Gain value. The definition of all XCG measures is based on the underlying concept of the value of gain, $xG[i]$, that a user obtains when examining the i-th result in the ranked output of an XML retrieval system. Given a ranked list of elements, where the element IDs are replaced with their relevance scores, the cumulated gain at rank i, denoted as $xCG[i]$, is computed as the sum of the relevance scores up to that rank:

$$xCG[i] = \sum_{j=1}^{i} xG[j] \quad (2)$$

Assuming that users prefer to be returned more relevant elements first, an ideal gain vector, xI, can be derived for each topic by filling the rank positions with the relevance scores of the relevant elements in decreasing order of their relevance scores. The corresponding cumulated ideal gain vector is denoted as xCI and is calculated analogue to $xCG[i]$. Both $xG[i]$ and $xI[j]$ are calculated using the element's specificity value:

$$xG[i] = spec(e_i) \tag{3}$$

$$xI[j] = spec(e_j) \tag{4}$$

where e_i is the i-th element in the system ranking, e_j is the j-th element in the ideal ranking, and the specificity score is given in Equation 1.

Effort-precision/gain-recall. Effort-precision at a given cumulated gain value, r, measures the amount of relative effort (where effort is measured in terms of number of visited ranks) that the user is required to spend when scanning a system's result ranking compared to the effort an ideal ranking would take in order to reach the given level of gain (illustrated by the horizontal line drawn at the cumulated gain value of r in Figure 1):

$$ep[r] = \frac{i_{ideal}}{i_{run}} \tag{5}$$

i_{ideal} is the rank position at which the cumulated gain of r is reached by the ideal curve and i_{run} is the rank position at which the cumulated gain of r is reached by the system run.

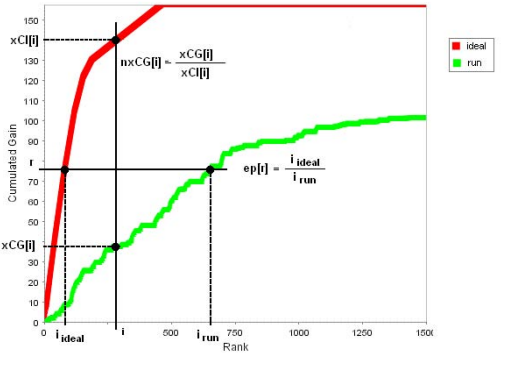


Fig. 1. Calculation of $nxCG$ and effort-precision ep

By scaling the recall axis to $[0, 1]$ (i.e. dividing by the total gain), effort-precision can be measured at arbitrary recall points, $gr[i]$ [5]:

$$gr[i] = \frac{xCG[i]}{xCI[n]} = \frac{\sum_{j=1}^{i} xG[j]}{\sum_{j=1}^{n} xI[j]} \tag{6}$$

where n is the total number of relevant elements in the full recall-base of the given topic. The range for i is $[0, 1500]$, where 1500 is the maximum length of a result list that participants could submit.

As with standard precision/recall, for averaging across topics, interpolation techniques are necessary to estimate effort-precision values at non-natural gain-recall points, e.g. at standard recall points $\{0.1, .., 1\}$.

The non-interpolated mean average effort-precision, denoted as *MAep*, is calculated by averaging the effort-precision values obtained for each rank where a relevant document is returned. For not retrieved relevant elements, a precision score of 0 is used.

4.3 Results Reported at INEX 2006

For the thorough task we report the following measures over all topics:

- non-interpolated mean average effort-precision (*MAep*)
- effort-precision/gain-recall up to rank 1500 (*ep/gr*)

5 Evaluation of the Focused Task

5.1 Assumptions

In this task, systems are asked to return the ranked list of the top 1500 most focused, relevant XML elements for each given topic, without returning overlapping elements. The task is similar to the thorough task in that it requires a ranking of XML elements, but here systems are required not only to estimate the relevance of elements, but also to decide which element(s), from a tree of relevant elements, are the most focused non-overlapping one(s).

5.2 Evaluation Measures

The normalized cumulated gain $nxCG[RCV]$ measure, from the XCG family of measures, was used in the evaluation of the focused task. System performance was reported at several rank cutoff values (RCV).

Normalized cumulated gain. For a given topic, the normalized cumulated gain measure is obtained by dividing a retrieval run's xCG vector by the corresponding ideal xCI vector (see Section 4 for the definition of these two vectors):

$$nxCG[i] := \frac{xCG[i]}{xCI[i]} \tag{7}$$

$xCG[i]$ takes its values from the full recall-base of the given topic and $i \in [0, 1500]$ where 1500 is the maximum length of a result list that participants could submit. $xCI[i]$ takes its values from the ideal recall-base (described below) and i ranges from 0 and the number of relevant elements for the given topic in the ideal recall-base. The gain values $xI[j]$ used in $xCI[i]$ are given by Equation 4. The gain values used in $xCG[i]$ are normalized as follows. For the j-th retrieved element, where j ranges from 1 to i:

$$xG_{norm}[j] = min(xG[j], xG[j_{ideal}] - \sum_{S} xG[k]) \tag{8}$$

where $xG[\cdot]$ is given by Equation 3, j_{ideal} is the rank of the ideal element that is on the same relevant path as the j-th relevant element, and S is the set of elements that overlap with that ideal element and that have been retrieved before rank j. The normalization ensures that a system retrieving all descendant relevant elements of an ideal element cannot achieve a better overall score then if it retrieved the ideal element.

For a given rank i, $nxCG[i]$ reflects the relative gain the user accumulated up to that rank, compared to the gain he/she could have attained if the system would have produced the optimum ranking. As illustrated in Figure 1, $nxCG$ is calculated by taking measurements on both the system and the ideal rankings' cumulated gain curves along the vertical line drawn at rank i. Here, rank position is used as the control variable and cumulated gain as the dependent variable.

Recall-bases. The evaluation of the focused retrieval task requires two recall-bases. The full recall-base is the list of all elements that contains any relevant information (which therefore includes all parents of any such element), already used in the thorough task. The ideal recall-base is a subset of the full recall-base, where overlap between relevant reference elements is removed so that the identified subset represents the set of ideal answers, i.e. the most focused elements that should be returned to the user.

The selection of ideal elements into the ideal recall-base is done by traversing an article's XML tree and selecting from the set of overlapping relevant elements, those with the highest gain value. The methodology to traverse an XML tree and select the ideal elements is as follows [10]: Given any two elements on a relevant path,[1] the element with the higher score is selected. In case two elements' scores are equal, the one higher in the tree is chosen (i.e. parent/ascendant). The procedure is applied recursively to all overlapping pairs of elements along a relevant path until one element remains. After all relevant paths in a document's tree have been processed, a final filtering is applied to eliminate any possible overlap among ideal elements, keeping from two overlapping ideal paths the shortest one.

[1] A relevant path is a path in an article file's XML tree, whose root element is the article element and whose leaf element is a relevant element.

5.3 Results Reported at INEX 2006

For the focused task we report the following measures over all topics:

– normalized cumulative gains at low RCV (i.e. early ranks) ($nxCG$[5,10,25,50])

6 Evaluation of the Relevant in Context Task

6.1 Assumptions

The relevant in context task is document (here Wikipedia article) retrieval, where not only the relevant articles should be retrieved but also a set of XML elements representing the relevant information within each article. In this task, there is a fixed result presentation format defined. Systems are expected to return, for each relevant XML Wikipedia article, a set of elements that focused on the relevant information within the article. The Wikipedia articles should be ranked in decreasing order of relevance, but there should not be a ranking of the contained XML elements. The set of result elements should not contain overlapping elements.

6.2 Evaluation Measures

The evaluation of this task is based on a ranked list of articles, where per article we obtain a score reflecting how well the retrieved set of elements corresponds to the relevant information in the article.

Score per article. For a retrieved article, the text retrieved by the selected set of elements is compared to the text highlighted by the assessor [9]. We calculate the following:

– Precision, as the fraction of retrieved text (in bytes) that is highlighted;
– Recall, as the fraction of highlighted text (in bytes) that is retrieved; and
– F-Score, as the combination of precision and recall using their harmonic mean, resulting in a score in [0,1] per article.

More formally, let a be a retrieved article, and let e be an element that belongs to $\mathcal{E}_a$, the set of retrieved elements from article a. Let $rsize(e)$ be the amount of highlighted (relevant) text contained by e (if there is no highlighted text in the element, $rsize(e) = 0$). Let $size(e)$ be the total number of characters (bytes) contained by e, and let $Trel(a)$ be the total amount of (highlighted) relevant text for the article a.

We measure the fraction of retrieved text that is highlighted for article a as:

$$P(a) = \frac{\sum_{e \in \mathcal{E}_a} rsize(e)}{\sum_{e \in \mathcal{E}_a} size(e)} \tag{9}$$

The $P(a)$ measure ensures that, to achieve a high precision value for the article a, the set of retrieved elements for that article needs to contain as little non-relevant information as possible.

We measure the fraction of highlighted text that is retrieved for article a as:

$$R(a) = \frac{\sum_{e \in \mathcal{E}_a} rsize(e)}{Trel(a)} \tag{10}$$

The $R(a)$ measure ensures that, to achieve a high recall value for the article a, the set of retrieved elements for that article needs to contain as much relevant information as possible.

The final score per article is calculated by combining the two precision and recall scores in the standard F-score (the harmonic mean) as follows:

$$F(a) = \frac{2 \cdot P(a) \cdot R(a)}{P(a) + R(a)} \tag{11}$$

The resulting F-score varies between 0 (article without relevance, or none of the relevance is retrieved) and 1 (all relevant text is retrieved and nothing more).[2] For retrieved non-relevant articles, $P(a) = R(a) = F(a) = 0$.

Scores for ranked list of articles. We have a ranked list of articles, and for each article we have an F-score $F(a_r) \in [0, 1]$, where a_r is the article retrieved at rank r. Hence, we need a generalized measure, and we utilise the most straightforward generalization of precision and recall as defined in [5].

Over the ranked list of articles, we calculate the following:

- generalized precision ($gP[r]$), as the sum of F-scores up to an article-rank r, divided by the rank r; and
- generalized recall ($gR[r]$), as the number of articles with relevance retrieved up to an article-rank r, divided by the total number of articles with relevance.

More formally, let us assume that for an INEX 2006 topic there are in total $Numrel$ articles with relevance, and let us also assume that the function $rel(a_r) = 1$ if article a_r contains relevant information, and $rel(a_r) = 0$ otherwise. At each rank r of the list of ranked articles, generalized precision is defined as:

$$gP[r] = \frac{\sum_{i=1}^{r} F(a_i)}{r} \tag{12}$$

[2] This task is very similar to the INEX assessors' task, who are highlighting relevant information in a pooled set of articles. Note that the assessors can highlight sentences, whereas systems can only return XML elements. This makes it impossible for a system to obtain a perfect score of 1 (although the theoretical maximum will be close to 1).

At each rank r of the list of ranked articles, generalized recall is defined as:

$$gR[r] = \frac{\sum_{i=1}^{r} rel(a_i)}{Numrel} \tag{13}$$

These generalized measures are compatible with the standard precision/recall measures used in traditional information retrieval. Specifically, the average generalized precision (AgP) for an INEX 2006 topic can be calculated by averaging the generalized precision at natural recall points where generalized recall increases. That is, averaging the generalized precision at ranks where an article with relevance is retrieved (the generalized precision of non-retrieved articles with relevance is 0). When looking at a set of topics, the mean average generalized precision ($MAgP$) is simply the mean of the average generalized precision scores per topic.

6.3 Results Reported at INEX 2006

For the relevant in context task we report the following measures over all topics:

- mean average generalized precision ($MAgP$)
- generalized precision at early ranks (gP[5, 10, 25, 50])

The official evaluation is based on the overall $MAgP$ measure.

7 Evaluation of the Best in Context Task

7.1 Assumptions

In this task, systems are required to return a ranked list of best entry points (BEP), one per article, to the user, representing the point in the article where they should start reading the relevant information in the article. The aim of the task is to first identify relevant articles, and then to identify the elements corresponding to the best entry points for the returned articles. Articles should be ranked according to their relevance.

7.2 Evaluation Measures

The evaluation of this task is performed with two measures, a distance measure, BEPD (for BEP-distance), and an extension of precision/recall (EPRUM) [10]. Both measures give a score of 0 for a ranked article that is not relevant, i.e. does not contain a BEP for the current topic.

BEPD. This measure is constructed as follows. For each document in a ranked list, $s(x, b)$ will measure how close the system-proposed entry point x is to the BEP b (as above, s is 0 if the article is not relevant). Closeness is assumed to be an inverse function of distance, with a maximum value of 1 if and only if the system hits the BEP and a minimum value of zero. We first measure the distance $d(x, b)$ in arbitrary units (characters). Next we remove the arbitrariness by normalizing d by the average article length L in characters ($d' = d/L$). Finally we make an inverse transformation to a $[0, 1]$ scale ($f(d') = A/(A + d')$), with a controlling parameter $A > 0$, which can be turned up to allow longer distances without much penalty, or down to reward systems which get very close to the BEP. The resulting formula is:

$$s(x, b) = A \times \frac{L}{A \times L + d(x, b)} \tag{14}$$

A value of $A = 10$ will give a score close to 1 for any answer in a relevant article; a value such as $A = 0.1$ will favour systems that return elements very close to the BEP.

BEPD for a single topic/ranked list is the sum of s values for the articles in the list divided by the total number of BEPs for this topic. Thus a system is penalized both for not retrieving the right articles and (to some extent controlled by A) for not pointing to the right places in the articles it does retrieve. This measure is averaged in the usual way over topics.

EPRUM-BEP-Exh-BEPDistance. The EPRUM measure is an extension of precision/recall developed for structured document collections and fine-grained user models [10]. While standard precision-recall assumes a simple user model, where the user consults retrieved elements (elements returned by the retrieval system) independently, EPRUM can capture the scenario where the user consults the context of retrieved elements. Most measures assume that a user sees the elements in their order of appearance in the result list. EPRUM on the other hand considers these elements as entry points to the collection from where the user can navigate to find relevant elements.

As in the classical precision at a given recall definition, the recall value R is the number of relevant elements the user wants to see. The recall level ℓ ($0 < \ell \leq 1$) is defined as the ratio of a recall R to the total number T of relevant units. The generalisation lies in the definition of the minimum number of ranks m the user needs to consult in the list to reach a recall level ℓ, or said otherwise a recall value of ℓT.

The user starts considering the first rank of the list. If (s)he finds more than ℓT relevant elements at this rank, then the information need is satisfied and (s)he stops. In this case, the user effort has been restricted to the consultation of the first rank of the list (m is 1). If not, (s)he proceeds to the second rank, etc. The definition of precision is based on the comparison of two minimum values: the minimum rank that achieves the specified recall over all the possible lists, and over the evaluated list. For a given recall level ℓ, precision is defined as:

$$\text{Precision@}\ell = \mathbb{E}\left[\begin{array}{c}\text{achievement}\\ \text{indicator}\\ \text{for a recall } \ell\end{array} \times \frac{\begin{array}{c}\text{minimum number of}\\ \text{consulted list items for}\\ \text{achieving a recall } \ell \text{ over all lists}\end{array}}{\begin{array}{c}\text{minimum number of}\\ \text{consulted list items for achieving}\\ \text{a recall } \ell \text{ over the evaluated list}\end{array}}\right] \quad (15)$$

where the achievement indicator is used to set the precision to 0 if the recall level cannot be reached for the evaluated list. This is compatible with the classical definition of precision at a given recall where the precision is set to 0 if the list does not contain enough relevant elements.

Similarly, we can extend the definition of precision at a given rank r with this definition:

$$\text{Precision@}r \quad = \quad \frac{1}{r} \quad \times \quad \mathbb{E}\left[\begin{array}{c}\text{minimum number of consulted}\\ \text{list items (over all lists)}\\ \text{for achieving the same level of}\\ \text{recall as the evaluated run}\end{array}\right] \quad (16)$$

EPRUM is defined by three parameters stating: (1) the rewarded elements, i.e. here the BEPs. (2) the relevance value of an element, which is set to 1 since there was only one relevance level (i.e. exhaustivity value [7]) in INEX 2006; and (3) the probability that the user goes from one element in the list to a target (BEP). For the best in context retrieval task, this probability is defined as $s(x, b)$, as defined in Equation 14, for any BEP b. This behaviour is defined stochastically, i.e. we only know that a random user sees the BEP with probability $s(x, b)$ if presented the element x in the list. With these settings, a ranking only made of BEPs will obtain a constant precision of 1 for all recall levels. The performance slowly decreases when returned elements are further away from the BEPs, and reach 0 when returned elements are not in relevant articles.

7.3 Results Reported at INEX 2006

For the best in context task we report the following measures over all topics.

- BEPD
- EPRUM-BEP-Exh-BEPDistance precision recall graph
- EPRUM-BEP-Exh-BEPDistance precision averaged over all recall values (mean average precision)

Although we reported results for the values 0.01, 0.1, 1, 10, 100 for the parameter A, the official evaluation is based on the value $A = 0.1$.

8 Discussions

8.1 Too Small Elements

Because of how relevance was assessed in INEX 2006, a high number of fully highlighted elements – the figure reported at the INEX workshop was 18% – (which will then obtain a specificity score of 1) were of link type (i.e. collectionlink, wikipedialink, redirectlink, unknownlink, outsidelink, weblink, etc.). This led to concerns regarding the use of such a set of relevance assessments to evaluate retrieval performance using the XCG measures.

Using the INEX 2005 assessment process would have avoided this problem because any element with some highlighted (relevant) content would have to be further assessed according to how exhaustive it was. The exhaustivity value of "?" was used to express that an element was too small to provide any meaningful information.

We therefore created a second set of assessments, where all link element types were ignored. For both the focused and the thorough tasks, the XCG measures were applied using this filtered assessment set. We then examined correlation when using the two sets of relevance assessments (the full set and the filtered set) by calculating the Kendall τ correlation [2] between their resulting respective system rankings. Previous work has considered all rankings with correlations greater than 0.9 as equivalent and rankings with correlation less than 0.8 as containing noticeable differences [12].

Table 1. Correlation of the XCG measures using the full and the filtered assessments

focused task	
$nxCG[5]$	0.9292135
$nxCG[10]$	0.933427
$nxCG[25]$	0.8989332
$nxCG[50]$	0.8748597
thorough task	
$MAep$	0.9484281

Table 1 shows that for the focused retrieval task, as evaluated by the XCG measures, how to consider the so-called too small elements seems important, as they can affect, to some extent, effectiveness measures. The change between the full to filtered set of assessments does not affect the relevant in context, and best in context tasks, because the metrics used to evaluate these tasks were not affected by the problem of too small elements. The official results of INEX 2006 for the focused and thorough tasks are based on the filtered assessments.

8.2 Ideal Recall-Base

For the focused retrieval task, as described in Section 5, the cumulated gain $xCG[i]$ at rank i take its values from the full recall-base, whereas the cumulated

gain $xCI[i]$ (for the ideal vector) takes its values from the ideal recall-base. One possible approach is for $xCI[i]$ to also take its value in the full recall-base. The resulting system ranking was compared to the initial one, also using Kendall τ correlation measure.

Table 2. Correlation of the $nxCG[DCV]$ results for the focused task, when the ideal recall-base is build as defined in Section 5 and when it corresponds to the full recall-base

$nxCG[10]$	0.8025281
$nxCG[25]$	0.844944
$nxCG[5]$	0.8185393
$nxCG[50]$	0.8420758

We can see that there is a small difference in the ranking of retrieval approaches. Given that the ideal gain vector is then built from the full set of relevant elements, i.e. the full recall-base, which contains overlapping XML elements, systems can never achieve 100% recall (as the task did not allow runs to return overlapping results). This, however, presents less of an issue when performance is measured at low rank cutoffs. A more serious problem is that what is measured here does not correspond to the task. Since all relevant elements in the full recall-base are considered ideal, systems are in effect rewarded for the retrieval of any relevant element, not just the most focused elements. Therefore, any improvement in performance cannot be attributed to a system's ability in locating the most focused element. We can only say that well performing systems were able to return relevant elements within the top RCV ranks. Furthermore, given that measuring performance at low rank cutoffs is highly sensitive to the individual measuring points, especially those very early in the ranking, the retrieval of relevant, but not ideal elements, can impact on the score quite significantly.

References

1. Clarke, C., Kamps, J., Lalmas, M.: INEX 2006 retrieval task and result submission specification. In: Fuhr, N., Lalmas, M., Trotman, A., (eds), INEX 2006 Workshop Pre-Proceedings, pp. 381–388 (2006)
2. Conover, W.: Practical Non-Parametric Statistics, 2nd edn. John Wiley & Sons, Inc, New York, NY (1980)
3. Järvelin, K., Kekäläinen, J.: Cumulated Gain-based evaluation of IR techniques. ACM Transactions on Information Systems (ACM TOIS) 20(4), 422–446 (2002)
4. Kazai, G., Lalmas, M.: eXtended Cumulated Gain Measures for the Evaluation of Content-oriented XML Retrieval. ACM Transactions on Information Systems (ACM TOIS) 24(4), 503–542 (2006)
5. Kekäläinen, J., Järvelin, K.: Using graded relevance assessments in IR evaluation. Journal of the American Society for Information Science and Technology 53(13), 1120–1129 (2002)

6. Lalmas, M.: INEX 2005 retrieval task and result submission specification. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 385–390. Springer, Heidelberg (2006)
7. Lalmas, M., Tombros, A.: INEX 2002 - 2006: Understanding XML Retrieval Evaluation. In: DELOS Conference on Digital Libraries, Tirrenia, Pisa, Italy (2007)
8. Malik, S., Trotman, A., Lalmas, M., Fuhr, N.: Overview of INEX 2006. In: Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006 (2007)
9. Pehcevski, J., Thom, J.A.: HiXEval: Highlighting XML retrieval evaluation. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 43–57. Springer, Heidelberg (2006)
10. Piwowarski, B., Dupret, G.: Evaluation in (XML) information retrieval: Expected precision-recall with user modelling (EPRUM). In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (2006)
11. Tombros, T., Larsen, B., Malik, S.: The interactive track at INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, Springer, Heidelberg (2005)
12. Voorhees, E.M.: Evaluation by highly relevant documents. In: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 74–82. ACM Press, New York (2001)

Choosing an Ideal Recall-Base for the Evaluation of the Focused Task: Sensitivity Analysis of the XCG Evaluation Measures

Gabriella Kazai

Microsoft Research,
Cambridge, UK
gabkaz@microsoft.com

Abstract. The paper investigates the measures of retrieval effectiveness employed in the ad-hoc track of INEX 2006. In particular, it looks at how the evaluation of the Focused task is affected when different methodology is employed in generating the so-called ideal recall-base, which forms the ground-truth of the evaluation. The results show that the choice of methodology can impact on the obtained performance scores and the relative ranking of systems in relation to each other, especially when the effectiveness scores are uniformly low across all systems. Most XCG measures show very similar levels of sensitivity to changes in the ideal recall-base.

1 Introduction

INEX 2006 defined four retrieval tasks within the ad-hoc track: Thorough, Focused, Relevant in Context, and Best in Context tasks. In the Thorough task, systems are to estimate the relevance of all XML elements in the collection and return a ranked list of the top 1500 results. The Focused task asks systems to return a ranked list of the "most focused" XML elements that satisfy the information need, without returning overlapping elements (e.g. a paragraph and its container section element). The Relevant in Context task is much like the Focused task, but here the ranked list consists of groups of the most focused elements, clustered by the unit of an article in the Wikipedia collection. Finally, in the Best in Context task, systems are required to return a ranked list of best entry points (one per article) to the user, representing the point in the article where users should start reading.

Apart from the latter task, for which separate best entry point judgements are obtained, the evaluation of the first three tasks relies on the (same) set of relevance assessments collected from human judges. These assessments are collected in the form of highlighted text passages, which are then automatically converted into assessments on XML elements. Due to the inherent nesting of XML elements, the resulting "full" recall-base consists of overlapping relevant XML elements (e.g. both a relevant paragraph element and its container section element will be included).

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 35–44, 2007.

The full recall-base is then used directly to evaluate the Thorough task, whereby the ranked list of elements in the recall-base is compared against the rankings produced by the retrieval systems. However, due to the introduced overlap of XML elements, this recall-base is not directly suitable for the evaluation of the Focused task [2], which requires an overlap-free recall-base, containing only the most focused, relevant XML elements.

In [1] a procedure was proposed to filter the full recall-base and generate a so-called ideal recall-base, where overlap is removed. The methodology has since been questioned and alternative methods have also been proposed, e.g. in [3]. In this paper, we examine and compare a number of different methods for the construction of the ideal recall-base. The question we investigate is not so much the problem of which method is correct, but how the conclusion of the evaluation is actually affected by the choice of this methodology. If the evaluation measures are sensitive to changes in the ideal recall-base then greater care needs to be taken to define the criteria for creating an ideal recall-base. On the other hand, if there is high agreement on the relative system rankings produced by the different methods, then one can spend less effort on tuning this parameter of the evaluation.

The paper is structured as follows. In Section 2 we introduce the problem in more detail, in Section 3 we describe the various methods for deriving an ideal recall-base, and in Section 4 we examine the correlation between the conclusions of the evaluation based on these different ideal recall-bases. We close with conclusions in Section 5.

2 Relevance Assessments

The relevance assessment procedure for INEX 2006 is based on a yellow-marker design and involves the highlighting of relevant text fragments in the articles of the Wikipedia collection by human judges. The collected relevance assessments are, hence, in the form of arbitrary sized text passages, which are not constrained by XML element boundaries. This is illustrated in Figure 1, where the highlighted relevant text fragment spans across a couple of paragraphs, starting in the middle of the first paragraph.

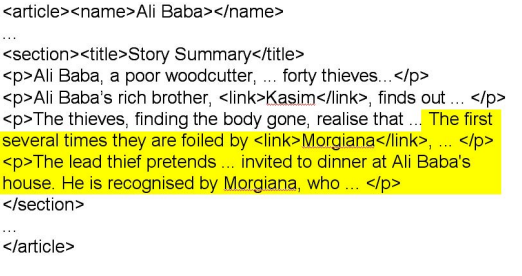


Fig. 1. Relevance assessments are collected as highlighted text fragments

The obtained relevant passages are then converted into assessments on XML elements. The conversion involves adding all XML elements that contain any highlighted text fragment to the recall-base. Due to the nested organisation of XML elements in an XML document's tree, a single highlighted text fragment typically adds several relevant XML elements to the recall-base[1]. For example, the excerpt from one of the INEX 2006 topic's relevance assessments in Figure 2 shows that a single highlighted text passage within the Wikipedia article of 2267781 is converted into four overlapping, relevant XML elements.

For each relevant XML element in the recall-base, the number of highlighted characters contained within the element (`@rsize`) is recorded, as well as the element's length in number of characters (`@size`).

```
<file collection="wikipedia" name="2267781">
  <passage start="/article[1]/body[1]/section[1]/p[5]/text()[1].89"
           end="/article[1]/body[1]/section[1]/p[5]/text()[1].199"
           size="111"/>
  <element path="/article[1]/body[1]/section[1]/p[5]" exhaustivity="2"
           size="569" rsize="111"/>
  <element path="/article[1]/body[1]/section[1]" exhaustivity="2"
           size="8470" rsize="111"/>
  <element path="/article[1]/body[1]" exhaustivity="2"
           size="20356" rsize="111"/>
  <element path="/article[1]" exhaustivity="2" size="20376" rsize="111"/>
</file>
```

Fig. 2. Excerpt from an INEX'06 topic's relevance assessments file. The `@size` attribute stores the XML element's length (in characters), while the `@rsize` attribute corresponds to the number of highlighted characters within the element.

Unlike in previous years, where relevance assessments were collected along two dimensions, *specificity* and *exhaustivity*, INEX 2006 only measures specificity. Exhaustivity, which is defined as the extent to which the document component (XML element) discusses the topic of request, is assumed a constant factor bearing no effect on the relevance score of an XML element. Specificity, which is defined as the extent to which a document component focuses on the topic of request, is calculated automatically as: the ratio of the number of highlighted characters contained within the XML element and the length of the element ($rsize/size$). Specificity hence can take any value in $[0, 1]$.

Since exhaustivity is a constant, the relevance score of an XML element is a function of the specificity score only.

[1] Since an ascendant node subsumes its descendant nodes' content, if a given XML node is relevant (to some degree), then all its ascendant nodes will also be relevant (to varying degrees).

3 Construction of an Ideal Recall-Base

A result of the adopted procedure to convert highlighted text fragments into relevance assessments on XML elements is that the derived full recall-base consists of overlapping XML elements. While this full recall-base can be used to evaluate the Thorough task, it is not appropriate for the evaluation of the Focused task.

In order to evaluate the Focused task, where overlap is not allowed, it is necessary to remove overlapping elements from the full recall-base. For this purpose, we define an ideal recall-base as a subset of the full recall-base, where overlap is removed so that the identified subset represents the set of ideal answers: the "most focused" relevant XML elements.

A range of methods can be defined to construct such an ideal recall-base: one may start from the highlighted relevant passages and select those XML elements as ideal nodes that completely cover the highlighted content while maximise the specificity score. Alternatively, we may select all leaf nodes that contain some highlighted content. Other methods may incorporate heuristics which consider "too small" nodes when deciding about the appropriate level of granularity. Alternative methods, e.g. [2,1,3], start from the full recall-base and aim to select nodes based on their combined exhaustivity and specificity values.

Rather than focusing on how these, or other, methods relate to particular criteria concerning the definition of what a "most focused" (or most appropriate granularity) relevant XML element might be, we are interested in the effect that the choice of such a method will have on the evaluation. For this reason, we define – and later experiment with – the following ideal recall-bases:

- Standard: The selection of ideal nodes into the ideal recall-base is done through the definition of a scoring function and a methodology for traversing an article's XML tree. The scoring function is used to associate with each node a relevance score, indicating its value to the user. In the case of INEX 2006, this score is the obtained specificity score. The methodology to traverse an XML tree and select the ideal nodes is as follows [3]: Given any two components on a relevant path[2], the component with the higher score is selected. In case two components' scores are equal, the one higher in the tree is chosen (i.e. parent/ascendant). The procedure is applied recursively to all overlapping pairs of components along a relevant path until one element remains. After all relevant paths in a document's tree have been processed, a final filtering is applied to eliminate any possible overlap among ideal components, keeping from two overlapping ideal paths the shortest one. We refer to this method as the Standard method, as it was used in the official evaluation of the Focused task at INEX 2006. From the sample tree shown in Figure 3 this method selects the nodes: `c` and `d`.
- Standard Descendant (S. Desc.): This follows the Standard method above, only from two components with equal scores, it selects the descendant node

[2] A relevant path is a path in an article file's XML tree, whose root node is the article element and whose leaf node is a relevant component.

(instead of the ascendant) [2]. This method was used in the official evaluation of the Focused task at INEX 2005. From the sample tree in Figure 3 this method selects the nodes: `d`, `g` and `h`.
- Leaf Nodes: This method simply selects all leaf nodes from all relevant paths. In Figure 3, this method selects the nodes: `d`, `g` and `h`.
- Parent of Leaf Nodes (P. of Leaf): This method selects as ideal node the parent node of a relevant leaf node. In Figure 3, this method selects the nodes: `b`, `e` and `f`.
- Article Nodes: This method selects the article's root element as ideal node. In Figure 3, this method selects the node: `a`.
- Full Recall-Base (Full RB): For comparison reasons, we also use the full recall-base where all relevant nodes, including overlapping nodes, appear in the ideal set. From the sample tree in Figure 3, this method selects all relevant nodes: `a`, `b`, `c`, `d`, `e`, `f`, `g` and `h`.

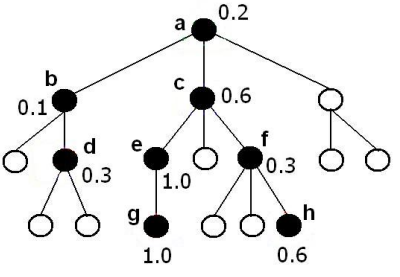


Fig. 3. An example XML tree. Relevant nodes are indicated by filled circles. The specificity score is shown next to each relevant node.

The Leaf and Article Nodes methods represent the two extremes of all possible selection mechanisms.

In the case when the full recall-base is used (which contains overlapping XML elements), systems can never achieve 100% recall (as the task did not allow runs to return overlapping results). Despite this, the evaluation may still be able to give us the correct relative ranking of systems' performances. Furthermore, the issue of 100% recall could be considered less of an issue when performance is measured at low document cutoff values (DCV). Hence, we also investigate the conclusions of the evaluation based on the full recall-base to gauge if these scores can be used to predict system performance within the Focused task.

4 Experiments

To test the effect of the ideal recall-base construction methodology on the conclusions of the evaluation, we used all 85 runs submitted to the Focused task[3]

[3] Note that some of these runs were disqualified from the official results as they contained overlapping components. This is however does not cause an issue in our investigations as the XCG measures handle overlap.

and evaluated these using the XCG measures listed in Table 1. A detailed explanation of the XCG measures can be found in [1].

Table 1. XCG measures

XCG measure	Description
$nxCG[DCV]$	Normalised cumulated gain at a given document cutoff value, calculated as the ratio of gain accumulated by the system up to the given rank and the gain that could be accumulated by the optimum best ranking.
$MAnxCG[DCV]$	Average of $nxCG[i]$ values over $i \in [1, DCV]$, calculated for each query, and then averaged over the set of test queries.
ep/gr graph	Effort-precision/gain-recall graph, where ep is calculated for 100 recall points in $[0, 1]$ and where a given value of ep reflects the amount of relative effort required compared to an ideal scenario in order to reach a predefined level of gain.
$MAep$	Average ep, calculated for each query as the average of the ep values obtained for each rank where a relevant document is found, then averaged over the set of test queries. For not retrieved relevant documents a precision score of 0 is assigned.
Q	Extended version of Sakai's Q-measure [4], which incorporates the rank position when calculating the cumulated gain, ensuring that performance is calculated against an always increasing ideal value.
R	Extended version of Sakai's R-measure [4], measuring performance at R rank documents, where R is the total number of relevant documents.

We used version 4 of the relevance assessments files.

Tables 2-6 show the correlation statistics for Kendall's τ (and Pearson's ρ) over the effectiveness scores obtained based on the the various ideal recall-bases and the full recall-base.

All measures project a very similar message: the choice of an ideal recall-base methodology can impact on the conclusions of the evaluation. The highest disagreement and least correlation – as expected – is found between the Article Node and the Leaf Nodes methods (as these represent the two extreme strategies). This means that the relative ranking of systems' will vary the most if the ideal recall-base is changed from containing relevant leaf nodes to article elements. Kendall's correlation values range for most measures from 0.31 to 0.39. The most stable measures (least sensitive to such a change of the ideal recall-base) are Q (0.49) and $MAnxCG[5]$ (0.44). The most sensitive measure (and hence least stable) is the ep/gr graph. Here the relationship between the system rankings produced by the Article Node and the Leaf Nodes methods is totally random (-0.01): the fact that a system is ranked high by one method provides no clue as to how the other method would rank it.

The highest correlation is found between the Standard Descendant method and the Leaf Nodes method. All of the measures show strong agreement

(≥ 0.91) between the conclusions of the evaluation based on these two methods. A reason for this could be that these methods – although based on rather different mechanisms for selecting ideal nodes – may have a large number of ideal nodes in common. Although we did not check for this explicitly, the fact that they both have an average depth of approximately 5.2 supports this argument (Leaf Nodes has 73007 ideal nodes in total for all queries with total specificity score of 144256.8, while S. Desc. has a total of 71815 ideal nodes with total specificity score of 141992).

The Standard ideal recall-base is, however, different from the above two (with a total of 18725 nodes of average depth of 3.7, and a total specificity score of 35811.96 over all queries), yet for $nxCG[1500]$ and $MAnxCG[1500]$, the Standard method highly agrees with the conclusions of the evaluation produced both by the Standard Descendant and the Leaf Nodes ideal recall-bases. For the rest of the measures, the Standard method shows better alignment with the Standard Descendant and the Leaf Nodes methods, compared to the Parent of Leaf Nodes (average depth of 3.07) and the Article Node (all nodes here have depth of 1) methods.

Comparing the $nxCG$ measures at various cutoffs, we see that as the cutoff is increased, the results produced by the Standard, S. Desc., Leaf Node and P. of Leaf Nodes methods tend to agree more with each other's conclusions, and tend to disagree with the results of the Article Node method. The same can be said for the $MAnxCG$ measures. Between the $nxCG$ and $MAnxCG$ measures, we can see that the latter is more resilient to changes in the ideal recall-base at lower cutoffs. At higher cutoffs, both measures are stable, and at $DCV = 1500$ they actually produce the same correlation scores (they are affected by changes in the ideal recall-base at the same rate).

The correlation figures for the overall performance measures (ep averaged over the 11 standard recall points, $MAep$, Q and R) indicate that the conclusions of the evaluation are again sensitive to changes in the ideal recall-base, especially where the effectiveness scores are very low across all systems. This is best seen for the ep/gr graphs, where for higher recall points, almost all systems score 0. The R measure has similar sensitivity properties to $MAep$, while Q shows the most resilience.

Looking at the conclusions of the evaluations based on the various ideal recall-bases compared to the conclusions drawn based on the full recall-base, we see that the latter could only be used as a reasonable estimate of relative system performances when the ideal recall-base is the largest subset of the full recall-base. For example, there is strong agreement between the relative system rankings obtained with the Full Recall-Base and the Standard Descendant and Leaf Nodes ideal recall-bases. The relative ranking of system performances obtained based on the Standard method, however, does not correlate very highly (≤ 0.9) with the conclusions of the evaluation based on the Full Recall-Base. This means that systems ranked high by one method may not necessarily be judged the same way by the other method. Therefore, we cannot safely rely on the conclusions of the evaluation based on the full recall-base when the "most focused" elements are defined through the Standard method.

Table 2. Kendall (and Pearson) correlation statistics for *nxCG* and *MAnxCG* at various cutoffs

$DCV = 5$	S. Desc.	Leaf Node	P. of Leaf	Article	Full RB
Standard *nxCG*	0.82 (0.98)	0.81 (0.98)	0.59 (0.93)	0.48 (0.81)	0.80 (0.98)
MAnxCG	0.88 (0.99)	0.86 (0.99)	0.66 (0.96)	0.53 (0.85)	0.85 (0.99)
S. Desc. *nxCG*		**0.98 (1.00)**	0.45 (0.86)	0.38 (0.74)	**0.95 (1.00)**
MAnxCG		**0.97 (1.00)**	0.56 (0.93)	0.46 (0.82)	**0.95 (1.00)**
Leaf Node *nxCG*			0.44 (0.85)	0.36 (0.73)	**0.96 (1.00)**
MAnxCG			0.54 (0.92)	0.44 (0.81)	**0.97 (1.00)**
P. of Leaf *nxCG*				0.82 (0.95)	0.44 (0.86)
MAnxCG				0.77 (0.95)	0.54 (0.93)
Article *nxCG*					0.37 (0.73)
MAnxCG					0.43 (0.81)
$DCV = 10$	S. Desc.	Leaf Node	P. of Leaf	Article	Full RB
Standard *nxCG*	0.83 (0.98)	0.82 (0.97)	0.59 (0.91)	0.44 (0.76)	0.80 (0.97)
MAnxCG	0.84 (0.99)	0.82 (0.98)	0.64 (0.94)	0.51 (0.82)	0.80 (0.98)
S. Desc. *nxCG*		**0.98 (1.00)**	0.45 (0.82)	0.32 (0.65)	**0.96 (1.00)**
MAnxCG		**0.98 (1.00)**	0.50 (0.89)	0.40 (0.76)	**0.95 (1.00)**
Leaf Node *nxCG*			0.43 (0.81)	0.31 (0.64)	**0.97 (1.00)**
MAnxCG			0.49 (0.88)	0.39 (0.75)	**0.97 (1.00)**
P. of Leaf *nxCG*				0.78 (0.94)	0.43 (0.81)
MAnxCG				0.79 (0.95)	0.48 (0.88)
Article *nxCG*					0.31 (0.65)
MAnxCG					0.38 (0.76)
$DCV = 50$	S. Desc.	Leaf Node	P. of Leaf	Article	Full RB
Standard *nxCG*	0.87 (0.97)	0.87 (0.96)	0.71 (0.92)	0.47 (0.70)	0.85 (0.96)
MAnxCG	0.86 (0.97)	0.85 (0.97)	0.64 (0.92)	0.47 (0.75)	0.83 (0.97)
S. Desc. *nxCG*		**0.99 (1.00)**	0.60 (0.81)	0.38 (0.57)	**0.96 (0.99)**
MAnxCG		**0.99 (1.00)**	0.51 (0.83)	0.35 (0.64)	**0.96 (0.99)**
Leaf Node *nxCG*			0.59 (0.81)	0.37 (0.56)	**0.96 (0.99)**
MAnxCG			0.50 (0.82)	0.34 (0.63)	**0.97 (0.99)**
P. of Leaf *nxCG*				0.73 (0.91)	0.59 (0.81)
MAnxCG				0.75 (0.93)	0.51 (0.83)
Article *nxCG*					0.38 (0.57)
MAnxCG					0.35 (0.64)
$DCV = 1500$	S. Desc.	Leaf Node	P. of Leaf	Article	Full RB
Standard *nxCG*	**0.93 (0.98)**	**0.91 (0.98)**	0.77 (0.92)	0.47 (0.65)	**0.90 (0.97)**
MAnxCG	**0.92 (0.98)**	**0.91 (0.98)**	0.77 (0.92)	0.47 (0.64)	**0.90 (0.97)**
S. Desc. *nxCG*		**0.98 (1.00)**	0.70 (0.84)	0.40 (0.54)	**0.95 (0.99)**
MAnxCG		**0.99 (1.00)**	0.70 (0.84)	0.40 (0.54)	**0.95 (0.99)**
Leaf Node *nxCG*			0.69 (0.83)	0.39 (0.52)	**0.97 (0.99)**
MAnxCG			0.69 (0.83)	0.39 (0.53)	**0.96 (0.99)**
P. of Leaf *nxCG*				0.70 (0.89)	0.70 (0.84)
MAnxCG				0.68 (0.89)	0.70 (0.84)
Article *nxCG*					0.40 (0.54)
MAnxCG					0.41 (0.55)

Table 3. Kendall (and Pearson) correlation statistics for *ep* averaged over the standard 11 recall points

Average *ep*	S. Desc.	Leaf Node	P. of Leaf	Article	Full RB
Standard	0.58 (0.76)	0.60 (0.71)	0.44 (0.64)	0.14 (0.08)	0.56 (0.54)
S. Desc.		**0.92 (0.96)**	0.29 (0.31)	0.03 (-0.03)	0.69 (0.71)
Leaf Node			0.25 (0.25)	-0.01 (-0.05)	0.70 (0.71)
P. of Leaf				0.52 (0.53)	0.15 (0.10)
Article					-0.06 (-0.08)

Table 4. Kendall (and Pearson) correlation statistics for *MAep*

MAep	S. Desc.	Leaf Node	P. of Leaf	Article	Full RB
Standard	0.86 (0.93)	0.85 (0.93)	0.60 (0.82)	0.45 (0.57)	0.84 (0.90)
S. Desc.		**0.98 (1.00)**	0.49 (0.61)	0.34 (0.34)	**0.95 (0.97)**
Leaf Node			0.48 (0.59)	0.33 (0.32)	**0.95 (0.97)**
P. of Leaf				0.82 (0.92)	0.49 (0.58)
Article					0.35 (0.33)

Table 5. Kendall (and Pearson) correlation statistics for *Q*

Q	S. Desc.	Leaf Node	P. of Leaf	Article	Full RB
Standard	0.87 (0.95)	0.85 (0.94)	0.71 (0.90)	0.62 (0.77)	0.83 (0.89)
S. Desc.		**0.99 (1.00)**	0.59 (0.73)	0.51 (0.58)	**0.94 (0.96)**
Leaf Node			0.58 (0.72)	0.49 (0.56)	**0.95 (0.96)**
P. of Leaf				**0.91 (0.97)**	0.59 (0.69)
Article					0.50 (0.53)

Table 6. Kendall (and Pearson) correlation statistics for *R*

R	S. Desc.	Leaf Node	P. of Leaf	Article	Full RB
Standard	0.85 (0.95)	0.84 (0.94)	0.61 (0.87)	0.48 (0.74)	0.79 (0.92)
S. Desc.		**0.99 (1.00)**	0.47 (0.70)	0.35 (0.54)	**0.92 (0.98)**
Leaf Node			0.47 (0.69)	0.35 (0.52)	**0.92 (0.98)**
P. of Leaf				0.86 (0.97)	0.44 (0.65)
Article					0.33 (0.49)

5 Conclusions

In this paper we investigated the effect that the ideal recall-base construction methodology has on the conclusions of the evaluation of the Focused task at INEX 2006.

The results show that the methodology can impact on the obtained performance scores and the relative ranking of systems in relation to each other, especially when the effectiveness scores are uniformly low across all systems. The

maximum disagreement was found between the two extreme strategies (Article Node and the Leaf Nodes methods), showing a correlation of only around 0.35 for most of the measures. This, however, indicates an upper bound, i.e. when no criterion is given for the definition of the concept of "most focused" XML element. However, given a criterion, the space of possible ideal recall-bases can be largely reduced (and hence the agreement increased).

We found that the most stable measures (least sensitive to such a change of the ideal recall-base) were Q (0.49) and $MAnxCG[5]$ (0.44). The most sensitive measure (and hence least stable) was the ep/gr graph due to the uniformly low scores at higher recall levels across all systems.

Our findings regarding the use of the full recall-base is that it can only be used as a reasonable estimate of relative system performances when the ideal recall-base is a sufficiently large subset of the full recall-base (e.g. when "most focused" elements are Leaf Nodes). However, we cannot safely rely on the conclusions of the evaluation based on the full recall-base if the "most focused" elements are defined through the Standard method.

Our future work will focus on quantifying the sensitivity of a measure based on the ratio of common nodes to all nodes contained within two ideal recall-bases.

References

1. Kazai, G., Lalmas, M.: eXtended Cumulated Gain Measures for the Evaluation of Content-oriented XML Retrieval. ACM Transactions on Information Systems (ACM TOIS) 24(4), 503–542 (2006)
2. Kazai, G., Lalmas, M., de Vries, A.: The overlap problem in content-oriented XML retrieval evaluation. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, July 2004, pp. 72–79. ACM, New York (2004)
3. Piwowarski, B., Gallinari, P., Dupret, G.: Precision recall with user modeling (PRUM): Application to structed information retrievel. ACM Transaction on Information System, vol. 25(1) (2007)
4. Sakai, T.: New performance metrics based on multigrade relevance: Their application to question answering. In: NTCIR Workshop 4 Meeting Working Notes (June 2004)

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 using the vector space model. To be more flexible for the query, we also developed a method of unification of multiple retrieved elements and a fragment indexing system. Our system is composed of an inverted file and an XML Path Language (XPath) path list. The validity of the method was tested as part of the ad hoc track in the Initiative for the Evaluation of XML Retrieval (INEX) 2006.

1 Introduction

In the research field of document information retrieval (IR), the unit of retrieval results returned by IR systems is a whole document or a document fragment, like a paragraph in passage retrieval. Traditional IR systems based on the vector space model compute feature vectors of the units and calculate the similarities between the units and the query. But nonetheless the problem of fragmentation of appropriate units still remains unsolved, which creates difficulties for the text segmentation task for IR, as with the automatic text summarization in document processing and the object segmentation in image processing.

Our research goal is to design and create an IR system which can be used for all types of user requirement. Therefore we think the unit of retrieval result should be neither a fixed portion nor a fixed element level of the XML document. If users have a variety of purposes, then preparing all possible units in advance is an expensive and even infeasible task, due to the combinatorial explosion in the number of element combinations. We propose an efficient index structure and flexible IR system for XML documents.

Overlap Removal. In an XML retrieval setting as shown in Figure 1, it is not an easy to identify the most appropriate elements to return to the user. IR systems have a difficult task to find out which are the most exhaustive and specific elements in the tree and return only these to the user, and producing result lists without overlapping elements. The problem is called *overlap removal*. So far, most of the approaches presented remove overlap by filtering the ranked lists by post-processing. Basically, by selecting the highest scored element from the ranked paths [8,11].

However, when the ideal unit from all element levels is needed by the user, it is the maximization problem of choosing possible elements which can fit into one unit. This problem is NP-hard as it is the knapsack problem. In addition, the space complexity is also high as it requires preparing all combinations in advance. To solve this problem,

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 45–56, 2007.

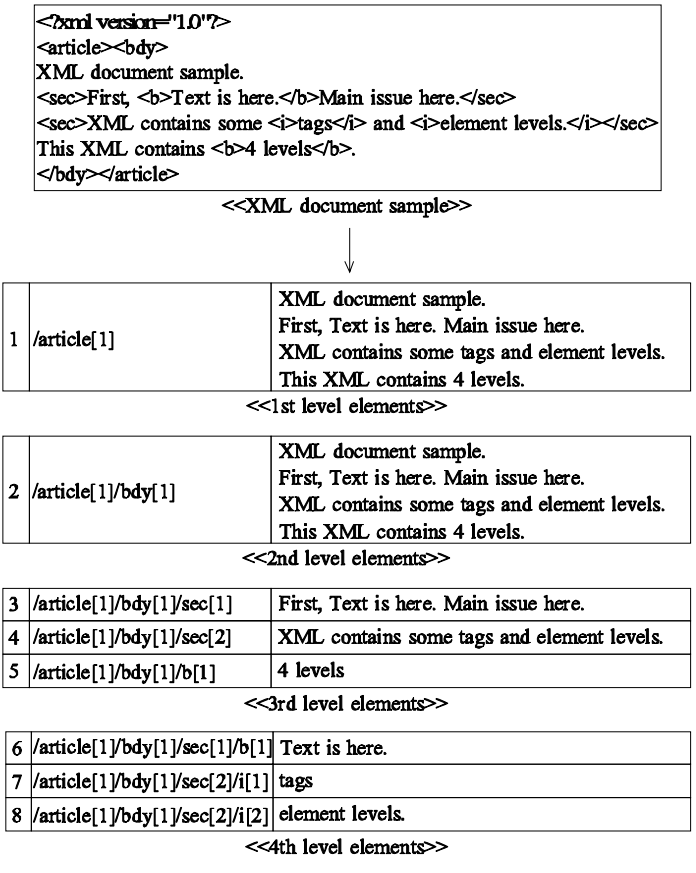
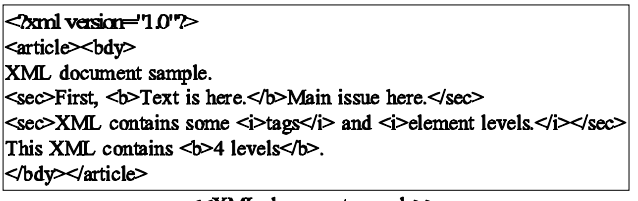


Fig. 1. XML document sample and multi-level element lists

one of the solution is the reduction of the amount of element levels, and the other is the improvement of memory efficiency when preparing all combinations.

Non-Overlapping. We propose a strategy for retrieving units of multi-level elements. Our strategy makes an index of atomic elements in order to obviate the difficulties of *overlap removal* and memory efficiency as shown in Figure 2. This means our strategy has non-overlapping elements. Meanwhile, our strategy must also determine the appropriate units in the retrieve phase. Hence, to unify the retrieved elements quickly, we also propose using a *sliding window* over the sequence of retrieved elements.

So far, we have proposed two ideas, the index of non-overlapping elements and the *sliding window*. But we must explore the possibility of them. In other words, our research purpose is to examine the "applicability" and the "scalability" about both the index of non-overlapping elements and the way of using a *sliding window*. This article describes the points and the validation of results of our proposed index structure and relevance calculation algorithm. Our motivations in INEX 2006 are listed bellow.



<<XML document sample>>

1	/article[1]	
2	/article[1]/bdy[1]	XML document sample. This XML contains .
3	/article[1]/bdy[1]/sec[1]	First, Main issue here.
4	/article[1]/bdy[1]/sec[1]/b[1]	Text is here.
5	/article[1]/bdy[1]/sec[2]	XML contains some and
6	/article[1]/bdy[1]/sec[2]/i[1]	tags
7	/article[1]/bdy[1]/sec[2]/i[2]	element levels.
8	/article[1]/bdy[1]/b[1]	4 levels

<<No overlapped elements>>

Fig. 2. XML document sample and non-overlapped element lists

1. To verify the "applicability" of the index structure and relevance calculation algorithm. The accuracy of retrieving results is tested.
2. To test the "scalability" of the index structure and relevance calculation algorithm, the following size and time are measured; the size of an inverted file (*space complexity*) and the processing time of retrieval (*time complexity*).

The rest of the article is divided into three sections. In section 2, we describe an architecture of our indexing and retrieving system for XML documents. In section 3, we describe experimental results. And in section 4, we discuss results and future work.

2 System Description

In this section we describe the architecture of our IR system as shown in Figure 3. The IR models include both an indexing method which builds the index of non-overlapping elements and a scoring method which calculates the similarities by using a *sliding window*.

For XML documents, one issue is the parsing method of text, excluding XML tags. So our system *stems* and *casefolds*, and also converts to lowercase. Another issue is the parsing method of the XML structure. The specification of the INEX collection uses some compositions of a small number of fundamental statistical values:

— $f_{t,d}$, the frequency of term t in document d;
— $f_{t,q}$, the frequency of term t in the query q;

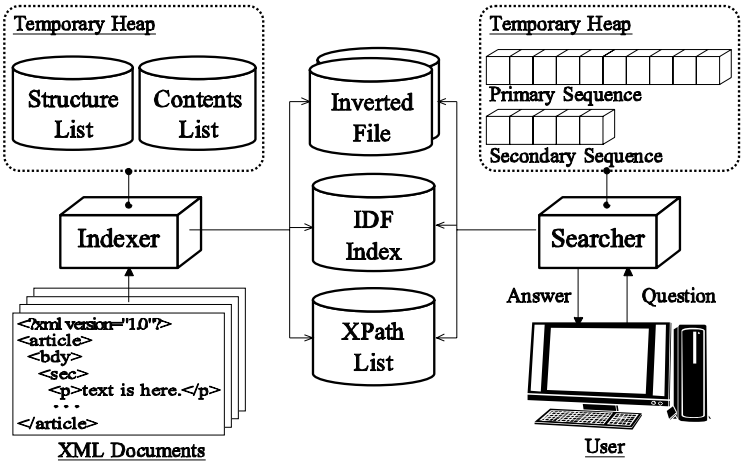


Fig. 3. System image diagram

— f_t, the number of documents containing one or more occurrences of term t;
— F_t, the number of occurrences of term t in the collection;
— m, the number of parsed terms in the query q;
— L, the number of elements in the collection.
— N, the number of documents in the collection; and
— n, the number of indexed terms in the collection.

The system contains a morphological analyzer using a hidden Markov model and XML parser written in Java (J2SE 1.4).

2.1 Indexing

We develop the IR system which is based on the vector space model using terms (as words)[2]. Our inverted file-based system (indexer and searcher) is partitioned (inverted files are spread across the processors). The indexer builds an XPath list to unify the retrieved elements, and an IDF index to reduce the calculation cost. During query evaluation, the query is decomposed into indexed terms by the searcher, and each indexed term is sent to the processor that holds the corresponding documents. Then, to calculate the similarities, the IDF index and the XPath list are used.

We develop two types of inverted file-based system for comparison, One is for XML articles, and the other one is for all XML elements.

Article Index. For XML articles, a unit is a whole XML document containing all terms appearing at any nesting level within the <article> tag. This type of inverted file-based system is commonly used by a traditional inverted file for document retrieval[13].

The theoretical complexity is $O(Nn)$, regarding *space complexity* for storing an inverted file, which is derived from two values. The intention is that *space complexity*

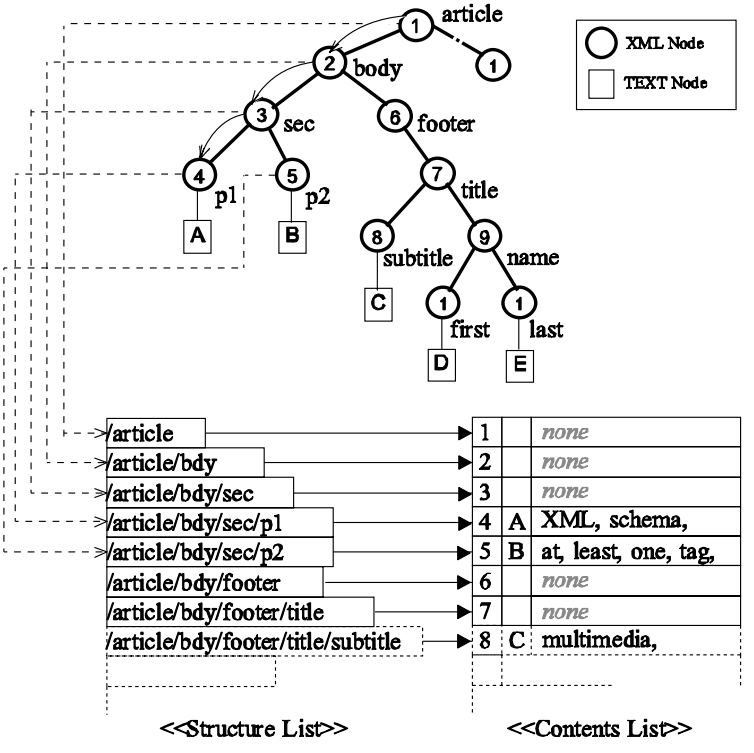


Fig. 4. Fragment indexing algorithm image

depends on the number of documents N and indexed terms n in the collection, to store the inverted file.

Element Index. For all XML elements, a unit is an XML element within the <article> tag. Every unit is non-overlapping. But in this case, to build retrieved results, or to unify the retrieved elements for a query, all units need to identify text and also the address of the XML element, as shown in Figure 4.

The theoretical complexity is $O(Ln + Lb)$, regarding *space complexity* for storing an inverted file and an XPath list, which depends on two factors. These are the size of the inverted file and the XPath list. For the inverted file, the size of the inverted file is Ln. Let a be an average number of elements in documents (within an <article> tag). Then, the number of elements L is a times as much as the number of documents N (i.e. $Ln = aNn$; Element Index is larger then Article Index). Furthermore, for the XPath list, let b be an average depth of absolute XPath, then the size of an XPath list is Lb.

XPath List. XPath is a language for addressing parts of an XML document[1]. In our scheme, an element in an XML document is identified by two parts, a file path of the XML document as article and an absolute path of the element in the XML document (our system does not support XML tag attributes). Hence, the file path identifies the XML document in the collection. And the absolute XPath expression identifies the

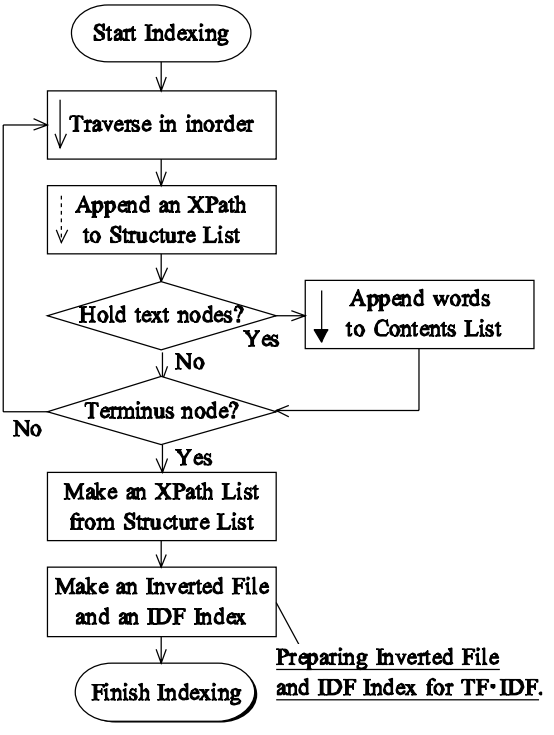


Fig. 5. Fragment indexing algorithm flowchart

XML element within an XML document, by indicating 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]

If the multi-level element index is overlapping, the size of inverted file is b times as much as the size of the inverted file Ln. This means that several elements have duplication of same terms. Hence, the worst-case *space complexity* of storing an inverted file is $O(Lnb+Lb)$. The relation between non-overlapping O_{no} and overlapping O_{ov} is as below,

$$Ln + Lb = L \cdot (n + b), \quad Lnb + Lb = L \cdot (nb + b),$$
$$O_{no}(L \cdot (\underline{n} + b)) < O_{ov}(L \cdot (\underline{nb} + b))$$

we have inequality, since $1 < b \ll n$, non-overlapping is consistently smaller than overlapping in terms of *space complexity*.

Fragment Indexing. Figure 5 shows an algorithm of making an inverted file for all XML elements with a coordinated XPath list. We call the indexing method the fragment indexing algorithm. The inverted file is composed of three members: a word, a node

identifier, and a frequency of the word in the node. The XPath list is numbered at all nodes of the XML trees inorder, separately from the inverted file.

The inorder traversal is utilized as a tool to assess the relationships of parent-child or sibling. To illustrate with the following example,

XPath 1 : /article[1],
XPath 2 : /article[1]/bdy[1],
XPath 3 : /article[1]/bdy[1]/sec[1],
XPath 4 : /article[1]/bdy[1]/sec[2],
XPath 5 : /article[1]/bdy[2],

starting from the root node, XPath 1 is as an origin of XPath list (XPath 1 $\subset$ *others*), XPath 2 is the child of XPath 1 (XPath 1 $\subset$ XPath 2), XPath 3 is the child of XPath 2 (XPath 2 $\subset$ XPath 3), XPath 4 is also the child of XPath 2 and the sibling of XPath 3 (XPath 2 $\subset$ XPath 4), and XPath 5 is the child of XPath 1 and the sibling of XPath 2 (XPath 1 $\subset$ XPath 5).

When the scan method is selected appropriately, the retrieval time is expected to be reduced. But this system uses the simple method of scanning the XPath list on HDD written in Java (J2SE 1.4).

2.2 Retrieval Model

Figure 6 shows an overview of this retrieval framework. This system is based on the vector space model with the *tf-idf* (term frequency-inverse document frequency) weight.

Vector Space Model. The system employs the traditional vector space model with *tf-idf* for XML document retrieval. And the system has a typical problem, which is to merge retrieved elements from the inverted files and to sort retrieved results. Our approach uses two sequences (Primary Sequence and Secondary Sequence) as shown in Figure 6. All retrieved elements are stored in the primary sequence, and then the top x ranking elements are selected in the primary sequence, by the priority queue algorithm on the secondary sequence.

Tf-idf Weighting. In the case of Article Index, let N be the total number of documents in the system and f_i be the document frequency of term t_i. The normalized frequency is $tf^A_{i,j}$ of term t_i in document d_j and inverse document frequency idf^A_i for t_i. And, let w_i be the weight in a query $\boldsymbol{q} = (w_1, w_2, \cdots, w_m)$, where $w_i \geq 0$, $w_i = f_{i,q}$ and m is the total number of parsed terms in the query. The summation is computed over all terms which are mentioned in the text of the document d_j. If the term t_i does not appear in the document d_j, then $tf^A_{i,j} = 0$. The best known term-weighting schemes and a simple *similarity* measure is *tf-idf* weighting, which are given by

$$tf^A_{i,j} = \frac{f_{i,j}}{\sum_{k=1}^{n} f_{k,j}}, \quad idf^A_i = \log \frac{N}{f_i}$$
$$sim^A(d_j, q) = \sum_{i=1}^{m} w_i \cdot tf^A_{i,j} \cdot idf^A_i$$
$$score^A(d_j, q) = sim^A(d_j, q)$$

The theoretical *time complexity* of retrieving documents is $O(m + Nlog(N))$ derived from two values. The intention is that *time complexity* depends on m which is the number of terms in the query used to retrieve the documents, and also depends on N is the

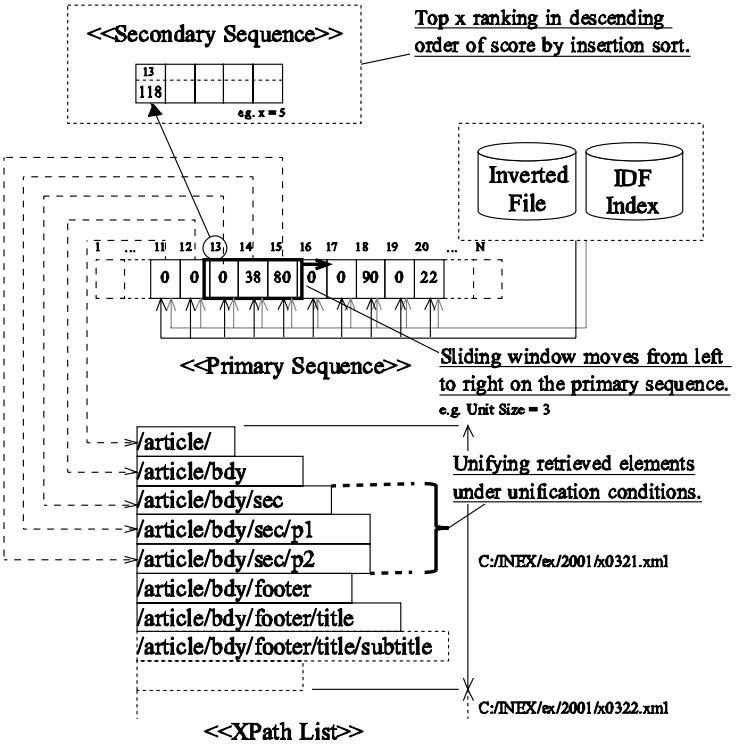


Fig. 6. Preferential unification algorithm image

number of documents that need to be sorted in the results list, at worst all documents in the collection.

In the case of the Element Index, additionally, to consolidate retrieved elements (i.e. L is the number of retrieved elements in the worst case), let D_l be the lth set of elements, which will become a meaningful unit to the user. Then *tf-idf* weights are given by

$$tf^E_{i,j} = \frac{f_{i,j}}{\sum_{k=1}^{n} f_{k,j}}, \quad idf^E_i = \log \frac{L}{f_i}$$
$$sim^E(d_j, q) = \sum_{i=1}^{m} w_i \cdot tf^E_{i,j} \cdot idf^E_i$$
$$score^E(D_l, q) = \sum_{d_j \in D_l} sim^E(d_j, q)$$

then *time complexity* depends on three values, which include an additional processing time of checking connectivity between two elements, in all combinations as worst. By using *brute force*, the theoretical complexity is $O(m + Llog(L) + L^2)$. Term one is *time complexity*, which depends on the number of terms in the query m, and term two depends on the complexity of sorting all elements L. In addition, term three depends on the number of a pair of elements L^2.

Unification Algorithm. In the case of the Element Index, a method to unify the retrieved elements and to combine the scores is required. A *sliding window* method is

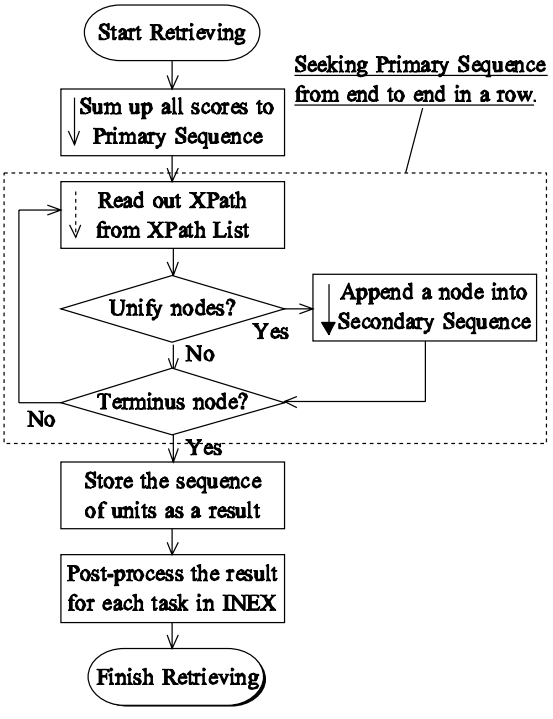


Fig. 7. Preferential unification algorithm flowchart

used to unify the retrieved elements and calculate the total similarity between the query and the unit, by scanning both the primary sequence and the XPath list concurrently, which is called the preferential unification algorithm as shown in Figure 7.

When the *sliding window* has applicable conditions, the meaningful units might be constructed appropriately. Specifically, the *sliding window* scans sequentially the primary sequence from end to end. The *sliding window* checks the relationships of retrieved elements, then if the conditions are matched, the scores of retrieved elements are combined and the top level element is stored in the secondary sequence.

By using *sliding window*, the theoretical complexity are modified: *time complexity* of retrieval using *sliding window* is $O(m + Llog(L) + (x + L))$. Here term three is a different expression from using *brute force*. $L^2 \rightarrow (x + L)$, means if $2 < x \ll L$ then $L^2 > (x + L)$. Thus *sliding window* is smaller than *brute force* in terms of *time complexity*. When the relation between *sliding window* O_{sw} and *brute force* O_{bf} is as below,

$$O_{sw}(m + Llog(L) + (\underline{x + L})) < O_{bf}(m + Llog(L) + \underline{L^2})$$

Unification Conditions. The issue of unifying of the retrieved elements remains. There are many conditions available for unifying, for instance the following terms.

Table 1. Thorough task

Run ID	Description	MAep	Rank
VSM_1	TF-IDF	0.0031	90/106
VSM_2	NormTF-IDF	0.0047	89/106
VSM_3	unit=10	0.0064	79/106

∗ep-gr(Quantization:gen, Overlap=off).

Table 2. Focused task

Run ID	Description	nxCG@50	Rank
VSM_4	unit=10	0.1159	63/85
VSM_5	unit=50	0.0976	73/85
VSM_6	unit=100	0.1149	64/85

∗ Metric:nxCG, Quantization:gen, Overlap=on.

Table 3. All in context task

Run ID	Description	F[50]	Rank
VSM_7	TF-IDF	0.1193	20/62
VSM_8	NormTF-IDF	0.1238	13/62
VSM_9	unit=10	0.0762	49/62

∗ Metric:hixeval-article, Quantization: Overlap=off(invalid submissions are discarded.).

Table 4. 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

∗Metric:BEPD.

1. The upper limit of the number of elements.
2. The upper limit of the number of terms.
3. The upper limit of the depth of XPath.
4. The upper limit of the distance between next elements.
5. The relationship between adjacent elements. (sibling or child?).
6. The upper limit of the summation score in the *sliding window*.

Although, in the experimental conditions, only two conditions are applied (to simplify the problem). One of the conditions is the upper limit of the number of elements (Unit Size). And the other is the relationship between adjacent elements. This means a retrieved unit is the upper limit of arbitrary size, and also a bunch of XML elements. Here, a unification score is a sum of retrieved element scores.

3 Experimental Results

The system was only designed for content-only (CO) queries. And, the post-processing eliminates **overlapping elements in each article** for some sub-tasks of the ad hoc XML retrieval task.

3.1 Thorough Task

The THOROUGH TASK, asks systems to estimate the relevance of elements in the collection, and there are no further restrictions. Overlap is permitted. Thus, the system has no use for post-processing. We produced two runs (VSM_{1,2}) with the Article Index and one run (VSM_3) with the Element Index. The result consists of some retrieved elements from the Element Index, with upper size limit. Table 1 shows that the Element Index is better than the Article Index but results are not good in the task. This result is semantically correct.

3.2 Focused Task

The FOCUSED TASK asks systems to return a ranked list of elements to the user. For the same topic, results may not be overlapping. That is, overlap is not permitted in the submitted run. We produced three runs (VSM_{4,5,6}) using the Element Index with different upper size limits. It is not necessary to apply the post-processing to reduce overlapping elements. Table 2 shows the runs are bad at the task.

3.3 All in Context Task

The ALL IN CONTEXT TASK asks systems to return relevant elements clustered per article to the user. That is, articles may not be interleaved. We produced two runs (VSM_{7,8}) with the Article Index and one run (VSM_9) with the Element Index. Post-processing is applied to the results from the Element Index to reduce **overlapping elements per article**. Table 3 shows the runs of Article Index are pretty good compared to Element Index, this result suggests the basic performance of our retrieval system is pretty good.

3.4 Best in Context Task

The BEST IN CONTEXT TASK asks systems to return articles with one best entry point to the user. That is, only single result per article is allowed. We produced three runs (VSM_{10,11,12}) by retrieving from the Element Index. The system applied the post-processing so that **only a single element per article** is retrieved. Table 4 shows a significant difference in the rank of the three runs in the task, in spite of only differences of unit size.

3.5 Time and Size

Table 5 shows times, then sizes of the inverted files for the Article Index was 936 [MB], and for the Element Index was 2,100 [MB] excluding XPath list: 1,749 [MB]. The retrieval time from the Element Index was 30 times as long as from the Article Index, but the retrieval time from the Element Index was very reasonable for full scanning retrieved results. The size of the Element Index was four times the size of the Article Index, which is a reasonable size.

Table 5. Processing time

	Searching time [s]	Index size
Article Index	2.01	600,000 [files]
Element Index	66.2	57,754,571 [nodes]

∗Searching time is the average of retrieval time per query. Number of nodes per file is about 100 nodes/file. The number of terms (words) is about 2,500,000. Experiments were run on a Celeron 2GHz, 2GB RAM. Implementation was in Java 1.4.

4 Conclusions

We proposed the fragment indexing system and the preferential unification algorithm. Even though the system used simple conditions, the experimental results produced various retrieved results for each task from the same index (the Element Index).

The system has one research issue, that is, the preferential unification algorithm takes longer than no unification. The cause is the time to scan the XPath list. Further research is therefore the reduction of scan time for the XPath list, improvement of the accuracy for various measures, and generalization to XML document retrieval.

References

1. XML Path Language (XPath) Version 1.0., http://www.w3.org/TR/xpath
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval (Acm Press Series), pp. 1–69, 141–162. Addison-Wesley, London (1999)
3. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM. 18, 613–620 (1975)
4. Evans, D., Lefferts, R.: Design and Evaluation of the CLARIT-TREC-2 system. In Harman, D.K. (ed.) Proceedings of the Second Text REtrieval Conference (TREC-2). pp. 500–548. NIST Special Publication (1994)
5. Tanioka, H., Yamamoto, K.: A Distributed Retrieval System for NTCIR-5 Patent Retrieval Task. In: The 5th NTCIR Workshop Meeting (2005)
6. Grabs, T., Schek, H.-J.: Flexible Information Retrieval on XML Documents. In: Blanken, H.M., Grabs, T., Schek, H.-J., Schenkel, R., Weikum, G. (eds.) Intelligent Search on XML Data. LNCS, vol. 2818, pp. 95–106. Springer, Heidelberg (2003)
7. Kazai, G., Gvert, N., Lalmas, M., Fuhr, N.: The INEX evaluation initiative. In: Blanken, H.M., Grabs, T., Schek, H.-J., Schenkel, R., Weikum, G. (eds.) Intelligent Search on XML Data. LNCS, vol. 2818, pp. 279–293. Springer, Heidelberg (2003)
8. Sigurbjornsson, B., Kamps, J., de Rijke, M.: An element-based approach to XML retrieval. In: INEX Workshop Proceedings. pp. 19–26 (2003)
9. Geva, S., Leo-Spork, M.: XPath Inverted File for Information Retrieval. In: INEX Workshop Proceedings. pp. 110–117 (2003)
10. Kelly, W., Geva, S., Sahama, T., Loke, W.: Distributed XML Information Retrieval. In: INEX Workshop Proceedings. pp. 126–133 (2003)
11. Mihajlovic, V., Ramirez, 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. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 72–87. Springer, Heidelberg (2006)
12. Trotman, A., Geva, S.: Passage Retrieval and XML-Retrieval Tasks. In: Proceedings of the SIGIR 2006, Workshop on XML Element Retrieval Methodology. pp. 43–50 (2006)
13. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Computing Surveys (CSUR) 38(2), Article 6 (2006)

CISR at INEX 2006

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

[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 a field-weighted BM25 model in INEX 2005, we revert back to using the traditional BM25 weighting function. Our main research aims in this year are to investigate the effects of document filtering (by considering only the top-ranked documents), element filtering by length cut-off and the effect 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. Finally, we propose a novel method for BEST IN CONTEXT TASK, and present our initial results.

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 suggest that 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 further exploiting the field-weighted method, our work focused on investigating the effects of document filtering, element filtering and using phrases.

In traditional text retrieval systems, a document is usually treated as an independent unit. But for XML element retrieval, elements in the same document are usually semantic relevant and are not 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 impact on the effectiveness of XML element retrieval e.g. the impact a parent element has on a child. 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 two different methods to compute the parent element's weight by merging its sub-elements weight. Both of these 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 an interpolation method of merging the document weight into element

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 57–63, 2007.

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 approach to element retrieval. That is, we divided element retrieval into two phases: we conducted document level retrieval and set a cut-off for the retrieved results (i.e. number of top-ranked documents) and then used the filtered results to further execute element level XML retrieval. Our aim is to investigate whether using top weighted documents could produce good results. Because of time limitations and issues with the newly adopted test collection, we did not compare the two methods directly in our experiments this year.

In order to avoid the too-small element problem, we restrict our set of retrievable units to article, body, section and paragraph, and 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 describe the BM25 model used in our experiment. Section 3 presents our results in INEX 2006. In Section 4, we evaluate and compare our results. We conclude the paper with closing remarks and future research directions to extend our work..

2 BM25 Model

In this work we use the original BM25 model. This is in contrast to our previous work in the area at INEX 2005 [1]. We reverted back to the BM25 model so that we could use it in the first phase of the method we deployed in our INEX 2006 experiments. For adhoc retrieval, and ignoring any repetition of terms in the query, BM25 can be simplified to:

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

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 the *j*th term, *dl* is the document length, *avdl* is the average document length across the collection, N is the total number of documents in the collection and k_1 and *b* are tuning parameters. We used the following settings for the tunning parameters as follows: b=0.75, k_1 = 1.2. The formula for *idf* in the traditional version of BM25 may in some circumstances (a very frequent term in a small collection) go negative. We may avoid this possibility by reverting to the old form of *idf*, namely log(N/df_j) [9].

3 Description of the Experiments

Within the adhoc XML retrieval task there are four sub-tasks: BEST IN CONTEXT TASK, THOROUGH TASK, FOCUSED TASK and RELEVANT IN CONTEXT TASK . For each sub-task, we submitted 3 runs only for CO queries but none for CAS queries. Our purpose, in addition to taking part in INEX, is to investigate a two-stage approach to element retrieval. The details of these experiments are as follows:

3.1 Best in Context Task

BEST IN CONTEXT TASK is a new adhoc task which aims at locating the best entry point of XML retrieval. We used two methods for this task and submitted four runs. In the first method, we take the element with the highest weight score (best-match element) in each document as the best entry point. This method was used in the two submitted runs BEST-BM25-cutoff400 and BEST-BM25-filter1500.

In the second method, we propose a 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 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.

In INEX 2006, the two submitted runs BEST-BM25-400-1500-level-p and BEST-BM25-Level-filter1500 used this method. For the initial evaluation of the effect of element length cut-off, we tuned element length cut-off between 0 to 550(character length) on the INEX 2005's data collection by using metrics Manx(10), Manx(25), Manx(50). The effect of the cut-off on "gen" Manx measure is shown in Fig.2. From this figure, we can see that using element length cut-off is beneficial especially to Manx(10) and Manx(25). We found an increase of 8% for the best tuned Manx(10) value over the non-tuned value.

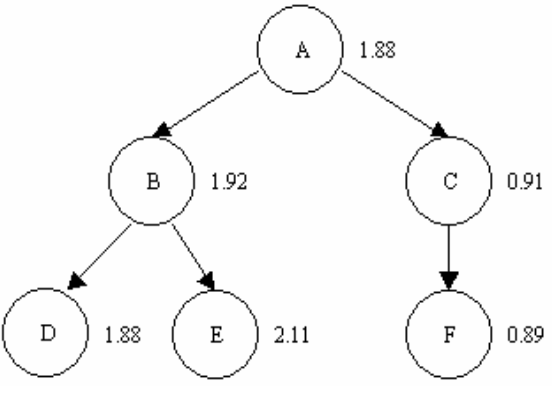


Fig. 1. XML element tree with element weight score

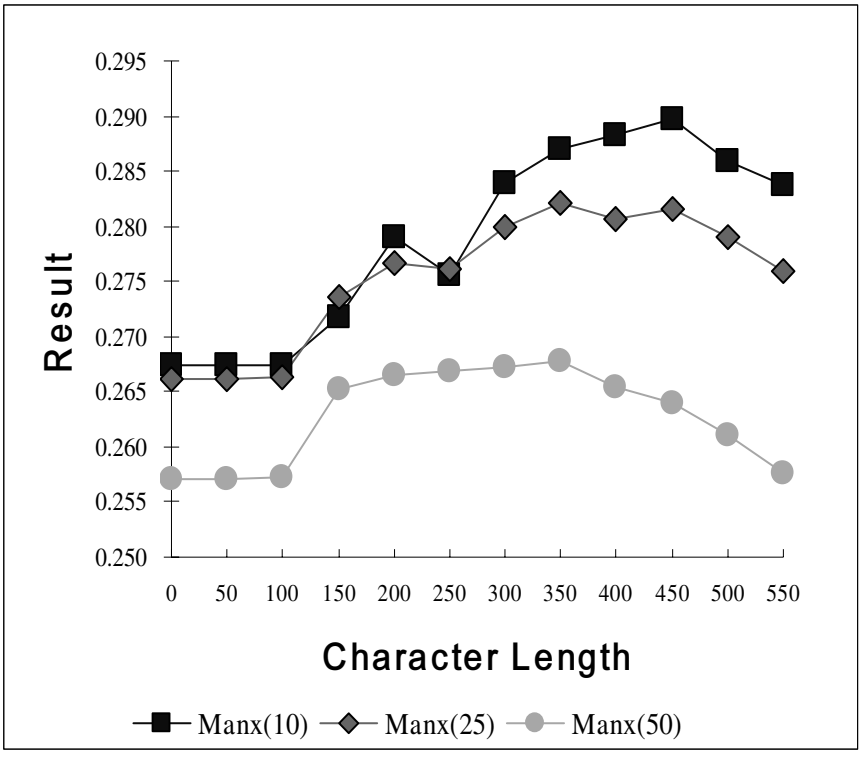


Fig. 2. Tuning results of element length cut-off on INEX 2005's data collection

3.2 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 the element length cut-off, which filters out elements shorter than a fixed length, is set to 400;
- THOR-BM25-400-1500-phrase uses the same element length cut-off, and it also set document result cut-off (1500) (i.e. restricted to the top 1500 ranked documents) and uses phrases instead of single words.

3.3 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 FOCU-BM25-cutoff400-filter1500 (above) except it uses phrases instead of single terms

3.4 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.

4 Evaluation

The evaluation results of our runs are shown in Tables 1 to 4. In Table 1, the run using the basic BM25 model was the best. In Tables 2 and 3, the runs using element length cut-off do best, which also rank them at the top of all INEX 2006 s corresponding submitted runs. In contrast, the runs using document filtering and phrases did 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, it can be seen that the two 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 three runs use the same document result set for locating the best entry point. Further investigation into why this happens is merited.

Table 1. Results for THOROUGH Task

Runs	Metric: ep/gr	
	un-filtered	**filtered**
THOR-BM25-nobody	**0.0228**	**0.0431**
THOR-BM25-nobody-cutoff400	0.0215	0.0407
THOR-BM25-400-1500-phrase	0.0118	0.0217

Table 2. Results for FOCUSED Task (Using un-filtered assessments)

Runs	Metric: nxCG (Overlap=on)			
	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. Results for RELEVANT IN CONTEXT Task

Runs	MAgP	Metric: gP			
		5	10	25	50
All-BM25-cutoff400	**0.1161**	**0.2936**	**0.2456**	**0.1622**	**0.1109**
All-BM25-cutoff400-filter1500	0.0583	0.2227	0.1725	0.1147	0.0741
All-BM25-cutoff400-filter1500-phrase	0.0602	0.2298	0.1734	0.1155	0.0746

Table 4. Results for BEST IN CONTEXT Task (Using un-filtered assessments)

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 INEX 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 that the latter two methods did not do well, while the first one did very well on FOCUSED TASK and RELEVANT IN CONTEXT TASK. Finally, in the THOROUGH TASK, the results were inconclusive as to whether or not the method was effective. 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. Given more time and resources, it would be useful to undertake a full scale study comparing the field weighting element retrieval used in last years' INEX and the two stage method

utilized for our experiments this year (it would also make sense to consider how to combine the two methods). In this context we could investigate the issue of sub-element cardinality and adjacency of element relevant to the information need.

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

References

[1] Lu, W., Robertson, S., Macfarlane, A.: Field-Weighted XML Retrieval Based on BM25. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, Springer, Heidelberg (2006)

[2] Robertson, S., Lu, W., Macfarlane, A.: XML-structured documents: retrievable units and inheritance. In: Larsen, H.L., Pasi, G., Ortiz-Arroyo, D., Andreasen, T., Christiansen, H. (eds.) FQAS 2006. LNCS (LNAI), vol. 4027, pp. 121–132. Springer, Heidelberg (2006)

[3] Lu, W., Robertson, S., Macfarlane, A.: Investigating Average Element Length for XML Retrieval by Using BM25 (submitted to review)

[4] Abolhassani, M., Fuhr, N., Malik, S.: HyREX at INEX, Proceedings of the Second Workshop of the Initiative for The Evaluation of XML Retrieval (INEX), December 15-17, 2003, Schloss Dagstuhl, Germany (2003)

[5] Geva, S.: GPX - Gardens Point XML Information Retrieval at INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, Springer, Heidelberg (2005)

[6] Ogilvie, P., Callan, J.: Hierarchical Language Models for XML Component Retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, Springer, Heidelberg (2005)

[7] Sigurbjornsson, B., Kamps, J., Rijke, M.: An element based approach to XML Retrieval. In: Proceedings of the Second Workshop of the Initiative for The Evaluation of XML Retrieval (INEX), December 15-17, 2003, Schloss Dagstuhl, Germany (2003)

[8] Mass, Y., Mandelbrod, M.: Component Ranking and Automatic Query Refinement for XML Retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, Springer, Heidelberg (2005)

[9] Robertson, S.: Understanding Inverse Document Frequency: On theoretical arguments for IDF. Journal of Documentation 60, 503–520 (2004)

Compact Representations in XML Retrieval

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 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.

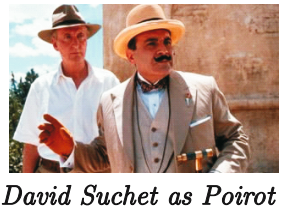
The remainder of this paper is organized as follows: Section 2 outlines our ideas for building a compact representation of an XML element. Section 3 describes the vector space model we used. The mixture language model is presented in section 4. Section 5 offers a detailed description and discussion of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

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 64–72, 2007.

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 a clue to the meaning and content of the original text. We therefore believe it can be used to enhance retrieval effectiveness. Furthermore, the XML structure of the INEX collection allows automatic locating of these words displayed in certain formats as they are marked-up in specific tags. Based on this consideration, we proposed to extract

Hercule Poirot is a fiction character, the primary detective of Agatha Christies novels. He appears in over 30 books. 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. Poirotis 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."



David Suchet as Poirot

Fig. 1. A sample text

these representative words from the original text to build a compact form of a text and emphasize its importance in identifying the relevance of the text. 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 based on the default similarity measure in Lucene[4]. For a collection C, the similarity between a document d and the query q is measured by:

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

where

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

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

$$norm_q = \sqrt{\sum_{t \in q} tf_{t,q}.idf_t^2} \tag{4}$$

$$norm_d = \sqrt{|d|} \tag{5}$$

$$coord_{q,d} = \frac{|q \cap d|}{|q|} \tag{6}$$

$weight_t = 1$ for all term t. 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.,

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

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

$$P(e|q) \propto P(e) \cdot P(q|e) \tag{8}$$

where e is an XML element; q is a query consisting of the terms t_1,...,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:

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

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.

$$P(t_i|C) = \frac{doc\_freq(t_i, e)}{\sum_{t' \in C} doc\_freq(t', C)} \tag{10}$$

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:

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

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[1,8] 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[1,6] 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, the first paragraph, the 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[7] 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.Our preliminary experiments indicated that

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

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[1,6] have already concluded that a successful element retrieval approach should be biased toward retrieving large elements.

Based on the above observation, 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 value of e. Assuming features are independent, we calculated the prior $P(e)$ by the following equation:

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

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.,

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

The decision to measure in characters instead of words was based on the consideration that smaller segments such as "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:

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

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

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

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[4]. 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 <title> or <description> 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 quoters in queries were simply removed. We processed the <description> part of topics by identifying and extracting noun phrases[5] to form queries.

5.3 Runs and Results

We submitted a total of six runs to the Thorough task.

1. VSM and nl-VSM: runs using vector space model based on full-text and compact representation of elements. For VSM, queries were created using terms in the <title> field. And queries for nl-VSM 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

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

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

Table 2 lists our results in the INEX official evaluation: the system-oriented MAep measure and the ranks among all submitted runs. Details of the evaluation metrics can be found in[3]. We were ranked at 39, 42, 66, 67, 68, and 69 out of 106 submissions. The vector space model based on combination of full-text and compact representation outperformed the language models. For runs using language models, estimates based on compact representation only(i.e., LM-1 and nl-LM-1) achieved comparable (and slightly better) performances with estimates based on the combination of full-text and compact representation (i.e., 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. Due to the pressure of time, we did not submit baseline runs for the retrieval models based on full-text 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 that natural language queries work quite well after some shallow pre-processing.

Table 2. Thorough results using filtered assessments

RunID	MAep	Rank
VSM	0.0294	39/106
nl-VSM	0.0290	42/106
LM-1	0.0180	66/106
nl-LM-1	0.0174	68/106
LM-2	0.0178	67/106
nl-LM-2	0.0172	69/106

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 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.

Acknowledgments

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

References

1. Kamps, J., Marx, M., de Rijke, M., Sigurbjornsson, B.: XML retrieval: What to retrieve? In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (2003)
2. Kamps, J., de Rijke, M., Sigurbjornsson, B.M.: Topic field selection and smoothing for XML retrieeval. In: Proceedings of the 4th Dutch-Belgian Information Retrieval Workshop (2003)
3. Kazai, G., Lalmas, M.: INEX 2005 evaluation metrics. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, Springer, Heidelberg (2006)

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

CSIRO's Participation in INEX 2006

Alexander Krumpholz and David Hawking

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

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

We relied on our own text-and-metadata retrieval system PADRE for indexing the data and processing the queries. Since PADRE is designed to retrieve documents rather than XML elements we preprocessed the collection to enable retrieval of sub-elements. From the set of queries we identified all the element-types required to be retrieved. We then added new pseudo documents corresponding to all the retrievable elements from the originals. Finally, this expanded collection was indexed with PADRE.

When processing queries, query elements were extracted from the INEX topics and pre-processed to generate queries in PADRE syntax. Results were post-processed to achieve the requirements of each particular task, such as elimination of super- and sub-elements of elements already retrieved.

Results obtained from this simple-minded approach were not particularly competitive but serve as a good basis for identification of necessary future enhancements.

1 Introduction

CSIRO's involvement in the INEX2006 [1] Ad Hoc Track[1] had three motivations. First, to explore the new Wikipedia collection [2], second to see how the naive splitting approach compares to current state-of-the-art XML retrieval engines[2], and third to establish a baseline for our future work.

PADRE, CSIRO's free-text and metadata retrieval system [4], implements a slightly modified Okapi BM25 algorithm [5], combined with Web-oriented evidence such as link counts, anchor text, URL length, and penalisation of multiple results from the same "site". In order for PADRE to retrieve sub-document parts like XML elements, the original XML files were split into smaller documents according to the XML elements relevant for retrieval. Thus, a much larger collection was indexed, comprising the original documents plus multiple, potentially overlapping, elements of those documents.

[1] Fully described elsewere in this volume.

[2] In our only previous participation (in 2002) [3] we successfully applied a similar approach but used manually generated queries, worked with a different collection and compared ourselves to a smaller, and presumably less sophisticated set of systems.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 73–81, 2007.

In all of the Ad Hoc tasks, an ideal retrieval system would be able to identify all the passages of text which are relevant to the query. This may potentially require that it be capable of recognizing alternative forms of a query word (e.g. run, ran and running) and also synonyms etc. (e.g. sprinted, jogged). In our submitted runs we didn't use any forms of stemming or query expansion.

In the Focused task, an ideal retrieval system needs to be able to choose which of a hierarchical set of elements should be retrieved – an element at too high a level will include too much irrelevant material, while an element at too low a level may be only part of a passage spanning adjacent elements. We implicitly relied on the Okapi BM25 scoring algorithm to rank elements within the same hierarchy (and across hierarchies).

PADRE has the ability to index metadata using metadata classes, which are represented by a single character. Default metadata classes are used for HTML to map the title, author and date to different classes. This way a PADRE query can restrict the search for keywords to specific elements. For example the query "t:architecture" would only return pages having a title element matching the term "architecture". In addition to the default metadata classes defined, PADRE allows the user to specify project specific configuration files containing the mapping of XML elements (and attributes) to metadata classes. The xml.cfg mapping file used for this experiment is shown in Listing **??**.

2 Approaches to INEX Ad-Hoc Tasks

This year the INEX community defined four challenges for the Ad-hoc track:

Thorough Task: The goal of the Thorough Task is to retrieve a ranked list of elements over the whole collection, regardless of overlap.

Focused Task: The Focused Task also asks for elements ranked over the whole collection, but does not allow overlaps. Overlaps occur when multiple retrieved search results contain identical XML elements. This happens when a sub-element or super-element of an already returned retrieval result is returned.

All In Context Task: The All In Context task aims for getting a list of ranked documents including the relevant, non-overlapping elements within the document.

Best In Context Task: The goal is to return a list of ranked documents with the best entry point for a reader.

Retrieval of structured data is quite related to database query languages and requires selection and projection operations.

A summary of the different runs is shown in Table 1.

The selection is achieved using the PADRE retrieval engine. In a pre-processing step we converted the NEXI queries into different PADRE queries which could be used by the Query Processor. For the two main query types i.e. Context and Structure (CAS) and Context Only (CO) the PADRE version of the NEXI query could be used. However, the keywords in the CAS and the CO topics are not identical

(see Table 3). In order to explore the effect of this difference a third query has been constructed by removing the structural hints from each CAS topic.

The projection aspect has been addressed in a post-processing phase. Using PADRE and the collection of element level sub-documents, we based our runs on a number of considerations: the standard PADRE version based on the Okapi BM25 algorithm delivers exactly what the Thorough Task (A) requires. The simplest possible way to achieve the Focused Task (B) would be to run the same query used for case (A), while suppressing overlaps by skipping results which are in fact descendants (sub-elements) or ancestors (super-elements) of a previously returned result. In order to generate a baseline for the Best In Context Task we took the simple assumption that Wikipedia articles are covering very specific topics and that the article itself would be a reasonable entry point for a user.

The following eight runs were submitted:

Table 1. CSIRO's runs

Run name	Selection based on	Projection based on
CSIRO-CAS1-A	CAS-Title	n/a
CSIRO-CAS2-A	CAS-Title	query specified elements
CSIRO-CO1-A	CO-Title	n/a
CSIRO-CO1-B	CO-Title	suppression of overlapping results
CSIRO-CO1-D	CO-Title	<article> elements
CSIRO-CO2-A	CAS-Title without structure	n/a
CSIRO-CO2-B	CAS-Title without structure	suppression of overlapping results
CSIRO-CO2-D	CAS-Title without structure	<article> elements

3 Architecture

The architecture used is shown in Figure 1.

3.1 Defining Metadata Classes

The xml.cfg specifies the elements indexed and the PADRE metadata classes used to represent them. The DOC element is an artificial element introduced to create a collection of sub-documents extracted from the original XML file. (see section 3.2)

3.2 Splitting Documents

The original XML documents were split into sub-documents in order to exploit PADRE's document retrieval capability to actually retrieve sub-documents. We split each XML document of the collection into sub-documents by extracting all elements matching one of the following XPath expressions and storing them in a new XML container document:

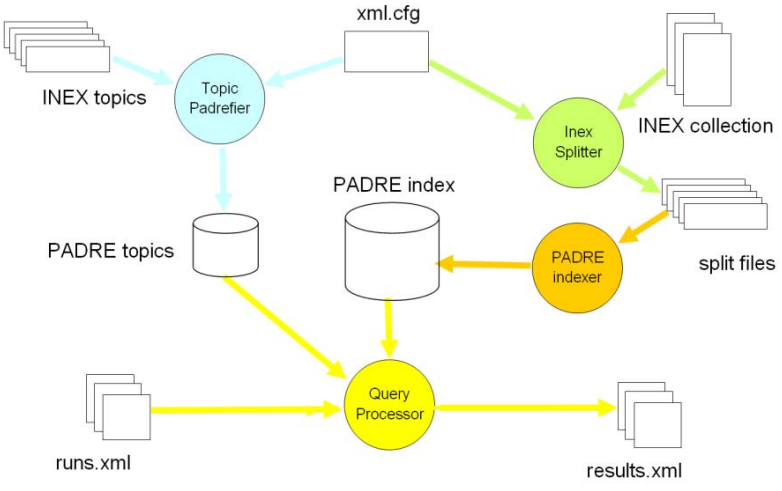


Fig. 1. Architecture

- /article
- //section
- //p
- //template

One problem with the current prototype is that sub-elements are not indexed with the super-element, i.e. metadata classes are disjunct. This would reduce the chance of a super-element being retrieved, because part of its text is effectively not indexed.

```
document,//DOC
a,1,,//DOC/article
b,1,,/body
e,1,,/caption
f,1,,/p
g,1,,/figure
j,1,,/title
l,1,,/table
n,1,,/template
o,1,,/section
p,1,,/template@name
q,1,,/collectionlink
r,1,,//DOC/article/name
w,1,,/link
x,0,,/DOCNO
```

Listing 1.1. The metadata classes defined in xml.cfg.

3.3 Transforming NEXI Topics

INEX topics are specified using the NEXI query language [6]. NEXI is XPath inspired, but allows the usage of vague selectors. The query `//section[about(.//p, security)]` matches sections that have a descendent element <p> containing the keyword 'security'.

Table 2. Comparison of the NEXI and PADRE query for topic 293

NEXI	//article[about(.,wifi)]//section[about(.,wifi security encryption)]
PADRE	a:wifi o:wifi o:security o:encryption

Table 3. A <title> and <castitle> element of topic 349

<title>	proprietary implementation +protocol +wireless +security
<castitle>	//article[about(., wireless)and about(.//p, security)] //link[about(., proprietary +implementation)]

```
<inex-submission participant-id="22" run-id="CSIRO-CO1-B"
task="Focused query="automatic">
  <topic-fields title="yes" castitle="no" description="no"
               narrative="no" ontopic_keywords="no"/>
  <description>Using the title as a query to padre,
              but suppressing overlapping results
  </description>
  <processing-instructions
    element-restriction="None"
    suppress="Overlap"
    padre-blocksize="1600"
  >
    <query>query.padre_title.uniq.join(' ')
    </query>
  </processing-instructions>
  <collections>
    <collection>wikipedia</collection>
  </collections>
</inex-submission>
```

Listing 1.2. The run configuration file 'run cfg'

In order to use the PADRE search engine we had to convert the official NEXI queries into PADRE queries. The script TopicPadrefier.rb (see figure 1) extracts the selection and projection component of each original CAS and CO topic, constructs the closest possible PADRE queries and stores them for future use. However, PADRE does not allow the definition of an infinite number of metadata classes and is therefore not expressive enough to correctly build PADRE queries for all possible NEXI queries. Some of the complex relationships of elements can therefore not be expressed in a generic manner.

3.4 Running INEX Topics

The result for each run has to be submitted as an XML file matching a DTD specified by the organizers. Since the Query processor needed processing instructions for each run, an extended version of the result DTD has been used as input for the Query processor. The Query processor extracts the processing instructions from the run specification file and replaces it with the result for the run. This way the run specification is automatically well documented.

3.5 Post-processing Results

The results have to be post-processed to meet the given tasks: i.e. suppressing multiple documents, or limiting to required element types or articles. For the Best In Context task all results other than articles have been suppressed in the runs CSIRO-CO1-D and CSIRO-CO2-D. For the Content and Structure task we suppressed all elements different than the structural request extracted from the query for the run CSIRO-CAS2-A, while we did not modify the result from PADRE for the runs CSIRO-CAS1-A, CSIRO-CO1-A, and CSIRO-CO2-A. For the runs CSIRO-CO1-B and CSIRO-CO2-B we matched each result with the results already delivered to filter out ascended or descended elements.

4 Results

Figures 2-4 show our results in respect to the other participant's submissions. Generally we obtained average results, with the following interesting observation: our structured results are worse than the unstructured ones. The graph and table in figure 2 show a significant gap between the CO*-A runs and the CAS*-A runs. (Note that the two CO*-A runs and the two CAS*-A lines in the graph essentially overlie each other.)

The tables list the runs submitted by CSIRO as well as the best and worst result for comparison. Notice, that the largest sample number at the bottom always shows the number of runs submitted for that task.

thorough		
Metric:ep-gr Quantization: gen,Overlap=off		
S.No	RunId	MAep
1		0.0709
33	CSIRO_CO2_A	0.0320
36	CSIRO_CO1_A	0.0313
78	CSIRO_CAS1_A	0.0122
80	CSIRO_CAS2_A	0.0111
106		0.0000

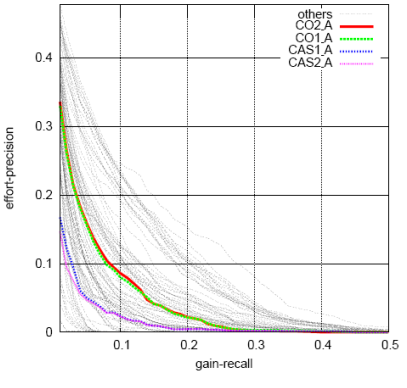


Fig. 2. Results: Thorough Task

All result sets show the related pairs of CSIRO runs almost side by side in the rankings, indistinguishable from each other in the small graphs attached. Figure 2 is the only one containing four runs by our team, one pair at position 33 and 36, the second pair at 78 and 80. We have not yet identified the reason why the Content Only runs clearly beat the Content And Structure runs. This is especially interesting since we would expect the additional information coming from the structure to improve retrieval quality. We suspect the disjunct metadata classes and the small number of element types we used for splitting contribute to that phenomenon.

The collocation of the other graphs are due to fact that the Context Only title and the Context And Structure title usually do not differ a lot.

The runs CAS1-A and CAS2-A share similar qualities as well. Apparently the element PADRE delivers automatically is often the one that would be specified in a CAS scenario.

focused		
Metric:nxCG Quantization: gen,Overlap=on		
S.No	RunId	nxCG@5
1		0.3944
18	CSIRO_CO1_B	0.3322
21	CSIRO_CO2_B	0.3304
85		0.0000

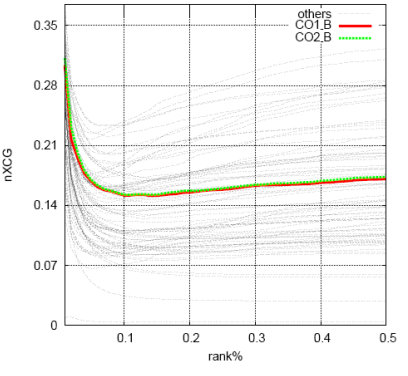


Fig. 3. Results: Focused Task

The Focused task turned out to be the best of our results, even though the approach of just suppressing duplicates is quite simplistic.

4.1 Failure/Success Analysis

Due to pressure from competing projects, we have been unable to make much progress on a failure/success analysis of our results.

We plan to construct some simple tools to facilitate comparison of our submitted result sets against the official results. We would like to identify cases:

Case 1 where we retrieved completely irrelevant items
Case 2 where we failed to retrieve items judged relevant
Case 3 where we retrieved items at higher levels in the XML hierarchy than the official answer.
Case 4 where we retrieved items at lower levels in the XML hierarchy than the official answer.

We are particularly interested in using these tools to answer the following questions:

BestInContext		
Metric: EPRUM-BEP-Exh-BEPDistance		
S.No	RunId	At A=0.01
1		0.0407
41	CSIRO_CO1_D	0.0166
43	CSIRO_CO2_D	0.0161
77		0.0000

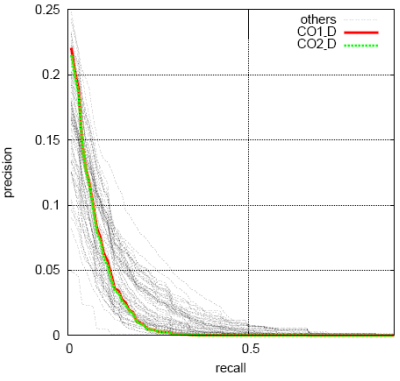


Fig. 4. Results: Best in Context Task

- Why were our attempts to exploit structure so unsuccessful? The scores for our CAS runs were substantially worse than our CO runs.
- For the Thorough task, Cases 1 and 2 are of particular interest. Did we need to apply stemming or query expansion? Could we exploit the available link structure? Were there bugs in our basic system or in pre and post processing scripts?
- For the Focused task, all four cases may adversely affect results.
- How much improvement might be gained by using a version of PADRE capable of indexing the same word occurrence as part of multiple overlapping elements?
- Were PADRE parameters set optimally? For example, were there adverse effects caused by the web-oriented ranking switched on by default in PADRE? Was the Okapi b parameter set to optimally take into account the relative length of documents?

5 Conclusions

One possible conclusion from our relatively unimpressive results this time is that the XML retrieval community has progressed significantly in recent years and that using a full text search engine to 'mimic' structured retrieval does not deliver competitive results.

This needs to be confirmed by the failure/success analysis described above.

We look forward to identifying the factors which are most significant in bridging the gap to state-of-the-art results.

References

1. DELOS Network of Excellence for Digital Libraries: Initiative for the Evaluation of XML retrieval http://qmir.dcs.qmw.ac.uk/inex/
2. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. SIGIR Forum (2006)

3. Vercoustre, A.M., Thom, J.A., Krumpholz, A., Mathieson, I., Wilkins, P., Wu, M., Craswell, N., Hawking, D.: Csiro inex experiments: Xml search using padre. In: INEX Workshop 2002, Schloss Dagstuhl, Germany (December 2002)
4. Hawking, D., Bailey, P., Craswell, N.: Efficient and flexible search using text and metadata. Technical Report TR2000-83, CSIRO Mathematical and Information Sciences (2000) http://www.ted.cmis.csiro.au/~dave/TR2000-83.ps.gz
5. Robertson, S.E., Walker, S., Hancock-Beaulieu, M.M., Jones, S., Gatford, M.: Okapi at TREC-3. In: Harman, D.K. (ed.) Proceedings of TREC-3, Gaithersburg MD, pp. 500–225. NIST special publication (November 1994) http://trec.nist.gov/pubs/trec3/papers/city.ps.gz
6. Trotman, A., Sigurbjörnsson, B.: Narrowed extended xpath i (nexi). In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 16–40. Springer, Heidelberg (2005)

Dynamic Element Retrieval in a Semi-structured Collection

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

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

Abstract. This paper describes our methodology for the dynamic retrieval of XML elements, an overview of its implementation in a structured environment, and the challenges introduced by applying it to the INEX Wikipedia [4] collection, which can more aptly be described as semi-structured. Our system is based on the vector space model [9] and its basic functions are performed using the Smart experimental retrieval system [8]. A major change in the system this year is the incorporation of a method for the dynamic computation of query term weights [6] to be correlated with the dynamically generated and weighted element vectors. Dynamic element retrieval requires only a single indexing of the document collection at the level of the basic indexing node (in this case, the paragraph). It returns a rank-ordered list of elements equivalent to that produced by the same query against an all-element index of the collection. (A detailed description of this method appears in [1].) As we move from a well structured collection, such as the INEX IEEE documents, to Wikipedia, changes in the structure of the articles must be accommodated.

1 Introduction

When we began our work with INEX in 2002, our goal was to assess the utility of Salton's vector space model [9] for XML retrieval. Familiarity with Smart [8] and faith in its capabilities led us to believe that this approach was promising if problems such as flexible retrieval (i.e., retrieval of elements at the desired degree of granularity) and ranking issues could be resolved. For the past several years, our research has focused on an approach for the dynamic retrieval of elements which provides a solution to both these problems.

The evolution of this approach is described in our earlier workshop papers, in particular [2] and [3]. In dealing with the INEX IEEE collections, we utilized Fox's extended vector space model [5], which allows for the incorporation of objective identifiers (such as date of publication) along with the normal content identifiers in the representation of a document. The body portion of the document (i.e., its text) is represented by one of the subjective subvectors in the extended vector representation.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 82–88, 2007.

The INEX Wikipedia collection [4] does not carry with it the corresponding information. Wikipedia articles are easily represented within the traditional vector space model, as seen below.

In INEX 2006 we use a system which generates and retrieves elements dynamically and returns a rank-ordered list of elements to the user. Results published elsewhere have demonstrated the successful utilization of this approach for structured retrieval [1]. Our current investigations center on how best to employ this approach in dealing with semi-structured data.

2 Background

This section presents a brief overview of the model and term weighting method upon which our system is based—i.e., the vector space model and *Lnu-ltu* term weighting. Details of this weighting scheme may be found in [10,11]. It is of particular interest in element retrieval where the elements often vary considerably in length, depending on type (e.g., paragraph versus section and body). *Lnu-ltu* weighting attempts to deal with the ranking issues resulting from disparity in document (element) length. (See [1] for a more detailed discussion of this issue.)

A basic model in information retrieval is the vector space model [9], wherein documents and queries are represented as weighted term vectors. The weight assigned to a term is indicative of the contribution of that term to the meaning of the document. The similarity between vectors (e.g., document and query) is represented by the mathematical similarity of the corresponding term vectors.

Of particular interest in this work is the issue of term weighting. We found in earlier experiments that best results were achieved when *Lnu-ltu* term weighting [11] was used with inner product as the similarity measure. *Lnu* term weights, used for the element vectors, are defined below:

$$\frac{(1+\log(term\_frequency)) \div (1+\log(average\_term\_frequency))}{(1-slope)+slope\times((number\_unique\_terms) \div pivot)}$$

where *tf* represents term frequency, slope is an empirically determined constant, and pivot is the average number of unique terms per document, calculated across the entire collection. Query terms are weighted using the *ltu* formula, as follows.

$$\frac{(1+\log(term\_frequency))\times\log(N \div n_k)}{(1-slope)+slope\times((number\_unique\_terms) \div pivot)}$$

Note that this formula depends both on N (the collection size) and n_k (the number of elements that contain the term).

3 System Description

This section describes first the current operation of the system and then a particular problem of interest that was solved during the past year by Ganapathibhotla—a method for the dynamic *ltu*-weighting of the query [6].

3.1 Dynamic Element Retrieval

XML text is processed in this system as follows. The documents are parsed using a simple XML parser which we wrote. We selected the paragraph—in our view, the smallest meaningful unit of text—as our basic indexing unit in the early stages of investigation. Thus a parsing of the documents into paragraphs is produced: paragraphs and queries are translated into Smart format and indexed by Smart. *Lnu-ltu* term weighting is applied. Retrieval takes place by running the *ltu*-weighted topics against the *Lnu*-weighted paragraph indexing of the collection using Smart. The result is a list of *elements* (paragraphs) ordered by decreasing similarity to the query.

Consider all the elements in this list having a non-zero correlation with the query. Each such element represents a terminal node (or paragraph) in the body of a document with some relationship to the query. Please note that although we have used the term *paragraph* here as a designator for smallest meaningful unit, in this context it means all the leaf nodes of a document tree. Thus the term *paragraph* can be used to refer to figure captions, lists, section titles, tables, abstracts—all the content-bearing elements that partition the document into mutually exclusive parts. Although some of these elements may not be leaf nodes according to their DTDs, they are treated as leaf nodes in this context because their child nodes are too small to be meaningful units in themselves.

For a particular query, Q, a search by Q against the paragraph index identified above produces a rank-ordered list of elements. Those elements having a positive correlation with Q identify the set of all documents of possible interest to it. We consider the n top-ranked elements in this list. Our method of dynamic element retrieval builds a tree representation for each document having an element in this list. Each tree is built based on a schema of the document (produced as a by-product of parsing). Given its set of terminal nodes in the form of term-frequency vectors, a document tree is built, bottom-up, according to its schema [3,7]. The content of each internal node is based solely on the content of its children. As each element vector is produced, it is *Lnu*-weighted and correlated with Q, which is itself *ltu*-weighted. After all element vectors, including the body element, have been generated, weighted and correlated with Q, the process continues with the next document. The resulting set of element vectors (i.e., all the elements from each document with a terminal node in the set of n top-ranked elements retrieved by Q) are then sorted and the top-ranked elements are reported.

3.2 Dynamic Query Weighting

Consider the formulas for term weighting given in Section 2. The *Lnu* term weighting of the element vectors at execution time is a relatively simple process. *Lnu* weights do not require information on global frequency that is not available in the dynamic

environment. The values of slope and pivot having been previously determined for the collection, *Lnu* weights are quickly computed.

The situation with the dynamic computation of the query term weights is quite different. Consider the *ltu* formula. At each level in the document tree, the values of N and n_k are element-dependent. Dynamic element retrieval is based on an initial retrieval against a paragraph indexing of the collection. The values of N (the number of paragraphs in the collection) and n_k (the number of paragraphs containing the term) are readily available as a by-product of the indexing process. In order for Q to be correctly weighted and correlated against each element vector in the document tree, the values of N and n_k associated with each query term must be the corresponding global values (i.e., the number of *elements* in the collection and the number of *elements* containing the query term).

The value of N is easily supplied by keeping track of the various types of elements encountered during the parsing process. Obtaining the value of n_k associated with a specific query term is more challenging. We have its (local) value at the level of the terminal node (i.e., paragraph). To determine its global value (i.e., the number of elements containing the term), we need information about the structure of each document tree in which the term is contained. In particular, for each occurrence of the term as a word type in a terminal node, we need to determine the number of parent elements in which it occurs. For example, suppose query term t_1 occurs in two different paragraphs of the same document. We need know whether both paragraphs occur as children of the same parent node (say subsection) or as children of different parents (two different subsections), and so on up the tree. And this process must be repeated for every document tree which contains t_1.

A very clever way to determine the number of containing elements for a particular term was devised by Ganapathibhotla [6], using the inverted file entry associated with the term in the paragraph indexing and a mapping between paragraph identifiers and their xpaths (required in our system for interaction between Smart and INEX formats). (See [1], [6] for details.) The calculation of n_k at execution time, clearly not feasible if it were required in the weighting of element vectors, is quite feasible in the weighting of query vectors, which are by their nature very short in comparison.

3.3 What About *n*?

There are very few parameters of interest associated with our method of dynamic element retrieval. Slope and pivot, used in the *Lnu-ltu* term weighting scheme, are collection dependent; determining slope requires some investigation and tuning. But the only truly interesting parameter (in the sense that it determines the number of trees generated and hence largely the time required for dynamic element retrieval) is n—the number of top-ranked paragraphs fed to our dynamic retrieval routine. It determines the upper bound on the number of trees built for each query. The actual number is determined by the number of paragraphs in this set belonging to the same document or set of documents.

Although not reported here in detail, our experiments with the 2004 and 2005 INEX IEEE collections reveal some interesting results. In these experiments, n varied

from 1 to 1000 (specifically, n = 1, 5, 10, 25, 50, 100, 250, 500, 1000). For the 2004 collection, under both generalized and strict quantizations and considering values of P@n for 10, 20, 50, 100, 500, and 1500 and average precision, dynamic element retrieval never required a value of n greater than 100 to produce a result equivalent to retrieval against the all-element index. For the 2005 collection, the results were very similar. The average number of trees built per query at n = 100 was 64 (for 2004) and 66 (for 2005). Looking at the average number of trees generated per query over all specified values of n greater than 50 indicates that, on average, fewer than 2/3 n trees are actually built.

For reasons discussed in the following section, corresponding experiments for the INEX Wikipedia collection are still in progress.

4 Problems Posed by Semi-structured Data

We encountered some interesting problems in adapting our method for dynamic element retrieval to the INEX Wikipedia collection. The IEEE collections are well structured. We found these traditional documents could be represented quite naturally using Fox's extended vector space model. Wikipedia documents, on the other hand, are easily represented using the traditional vector space model. The really significant difference between these two collections from our point of view, however, lies in how they are structured. Dynamic element retrieval depends on having all the terminal nodes of a document represented in the paragraph index. The initial paragraph retrieval gives us a good indication of which documents are of interest to the query in this case because all paragraphs that correlate highly with it are identified (thereby identifying their parent documents). The Wikipedia collection, on the other hand, contains untagged text which is distributed throughout the documents at the body and section levels. This untagged text cannot be retrieved except as a component of its parent element.

This is not a problem with respect to retrieval from an all-element index. The elements are parsed, collected, and indexed. Retrieval takes place in the normal manner. With dynamic element retrieval, untagged text impacts the method at two points: (1) during parsing, when untagged text must be identified (to be subsequently used in generating the document schemas so that the bottom-up generation of the document tree can take place properly with untagged text included at its parent level); and (2) during the initial retrieval against the paragraph (or terminal node) index, when documents potentially important to the query are identified. The interesting question here, which we have yet to answer, is whether the untagged text is important from a retrieval viewpoint.

Our current methodology deals with this problem by gathering all untagged text within an element and treating it as a separate child element (equivalent to a paragraph element) of its parent. Thus untagged text within a section becomes a separate element, (specially tagged as an <mt> element), which is attached to its parent section, and untagged text at the body level is treated similarly and attached as a child of the body element. Using this approach, dynamic element retrieval can proceed in the same manner used for structured text. The issue of interest here is

whether the inclusion of <mt> elements with the paragraph elements in the initial retrieval materially affects the elements retrieved dynamically and if so, to what extent.

Experiments are currently underway to determine the answers to this and other, related questions. The system requires tuning against the relevance assessments to determine an appropriate value of slope in the *Lnu-ltu* term weighting formula; the process, while straight-forward, is time-consuming. The size of this collection is also a factor. We have faced a number of space-related issues and hardware failures which have deterred progress on this work.

5 Conclusions

Our current system has achieved its major goal—it retrieves elements dynamically and returns a rank-ordered list of elements equivalent to that retrieved by a search of the corresponding all-element index. Exact *Lnu-ltu* term weights are utilized in this process. It requires only a single indexing of the collection at the paragraph level rather than either an all-element or multiple indexings, which are expensive to produce and maintain. As [1] shows, this method works well for structured retrieval. As we adapt our methods for utilization in the semi-structured environment of the INEX Wikipedia collection, we aim to determine the impact of this structural change on the retrieval process.

References

[1] Crouch, C.: Dynamic element retrieval in a structured environment. ACM Transactions on Information Systems 24(4), 437–454 (2006)

[2] Crouch, C., Mahajan, A., Bellamkonda, A.: Flexible retrieval based on the vector space model. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 292–302. Springer, Heidelberg (2005)

[3] Crouch, C., Khanna, S., Potnis, P., Daddapaneni, N.: The dynamic retrieval of XML elements. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 268–281. Springer, Heidelberg (2006)

[4] Denoyer, L., Gallineri, P.: The Wikipedia XML corpus. In: INEX Workshop Pre-Proceedings, pp. 367–372. (2006) http://inex.is.informatik.uni-duisberg.de/2006

[5] Fox, E.A.: Extending the Boolean and vector space models of information retrieval with p-norm queries and multiple concept types. Ph.D. Dissertation, Department of Computer Science, Cornell University (1983)

[6] Ganapathibhotla, M.: Query processing in a flexible retrieval environment. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth, Duluth, MN (2006) http://www.d.umn.edu/cs/thesis/Ganapathibhotla.pdf

[7] Khanna, S.: Design and implementation of a flexible retrieval system. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth, Duluth, MN (2005) http://www.d.umn.edu/cs/thesis/khanna.pdf

[8] Salton, G. (ed.): The Smart Rretrieval System—Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs (1971)

[9] Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Comm. ACM 18(11), 613–620 (1975)
[10] Singhal, A.: AT&T at TREC-6. In: The Sixth Text REtrieval Conf (TREC-6), NIST SP 500-240, pp. 215–225 (1998)
[11] Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: Proc. of the 19th Annual International ACM SIGIR Conference, pp. 21–29 (1996)

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* 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. Our main contribution is to investigate the use of summaries and the flexibility they provide when dealing with structural constraints. We describe algorithms for retrieval using summaries. 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

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 89–103, 2007.

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 precompute 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 [8,13,14]. Vague structural conditions were introduced in [23] 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 [13] is the only system that provides efficient support for finding the top-k results. Other recent proposals for XML ranked retrieval include [17] and [20]. The former uses dataguides and TA-style top-k algorithms [11], but differs from our work in that their experiments are limited to DB-like queries rather than XML retrieval queries. In contrast, [20] focuses on efficient evaluation of approximate structural matches without considering keyword search. The closest work to ours is TopX [24]. 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 existing scoring models such as TopX [24] and BM25E [18]. The main difference from TopX and BM25E is that tags (element names) 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 [25] 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)[19]. *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 [10] modified for XML. The *TReX* function is a generalization of the scoring function employed in the TopX query engine [24]. Its novelty is that the score of an element is given with respect to 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, with respect to 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 $t_1, \ldots, t_m$ of terms, an element set S and an element e in S, the BM25 score of e is given by the following formula.

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

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. One may note that any scoring function based on term-frequency could be used with no loss of generality in our approach. In comparison, *TReX* uses statistics within groups of elements defined by structural constraints. More formally, our BM25 formula uses frequency statistics with respect to an element set S rather than using only statistics with respect to 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 [10]. 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 with respect to 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

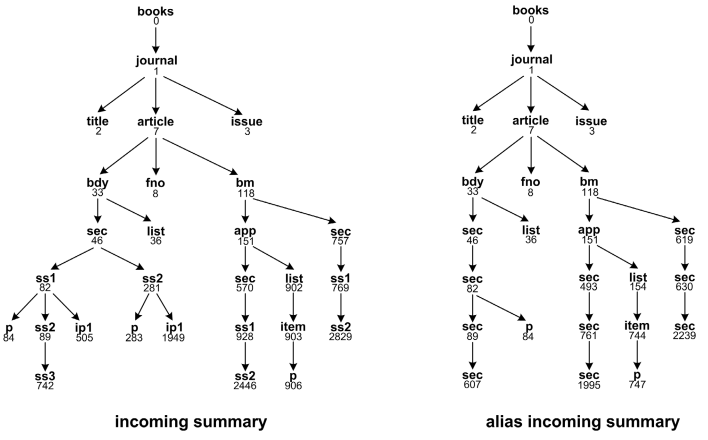


Fig. 1. Fragment of the incoming and alias incoming summary trees for the INEX IEEE collection

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 [12], the T-index family [21], ToXin [22], A(k)-index [16], F&B-Index and F+B-Index [15]. 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 {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 {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 $[sid_1, \ldots, sid_m]$ and a set of terms $[t_1, \ldots, t_n]$, the system retrieves the elements that *(1)* are in the extent of a node with sid in $sid_1, \ldots, sid_m$, and *(2)* contain at least one of the terms $t_1, \ldots, t_n$. For each such element e, term and element frequencies are computed and a BM25 score $score_s(e \mid t_1, \ldots, t_n)$ 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(SID, docID, endPos, length)
PostingLists(token, docID, offset, postingDataEntry)
RPL(token, iR, SID, docID, endPos, 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 $doc_1, o_1^1, \ldots, o_{i_1}^1, doc_2, o_1^2, \ldots, o_{i_2}^2, \ldots, doc_k, o_1^k, \ldots, o_{i_k}^k$ where

$doc_1, \ldots, doc_k$ is a sorted list of document identifiers, and each $o^j_1, \ldots, o^j_{i_j}$ 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 $sid_1, \ldots, sid_m$ and a list of terms $t_1, \ldots, t_n$. 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 $t_1, \ldots, t_n$ 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.

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

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 × 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 ← 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 ← I_{t_j}.nextPosition(t_j)
 9: repeat
10:    let x be the index for which pos_x = min{pos_1, ..., 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] ← 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, ..., n] then
18:             create a new list tf_{e_i} from the n values C[i][1, ..., n]
19:             add (e_i, tf_{e_i}) to L
20:             reset all the cells C[i][1, ..., n] to 0
21:          e_i ← I_{sid_i}.nextElementAfter(pos_x)
22:          if start(e_i) < pos_x < end(e_i) then
23:             C[i][x] ← C[i][x] + 1
24:    pos_x ← 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 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 $sid_1, \ldots, sid_m$ of sid's and a list $t_1, \ldots, t_n$ of terms, RPLs can be used to efficiently compute top-k answers. Let t be one of the terms $t_1, \ldots, t_n$. The top-k relevant elements with respect to$t$ and $sid_1, \ldots, sid_m$ can be easily retrieved from the RPL of t by iterating over this RPL and selecting the top elements whose sid is among $sid_1, \ldots, sid_m$. Note that the elements are provided sorted by their rank. We can then use threshold algorithm (TA), similar to the one used in TopX [24], 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.* [11] 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. Average 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. Average 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

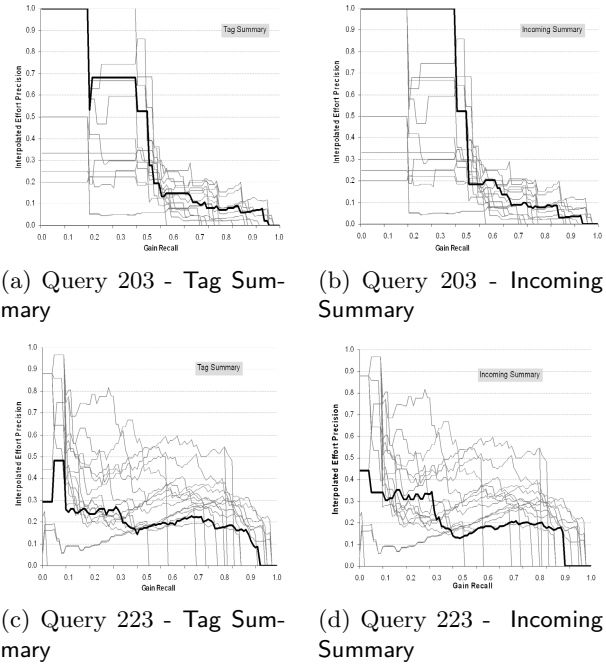
Table 5. Evaluation Time (in seconds) *ERA* Using Incoming for Wikipedia

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 4. 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.

Figure 4 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 4 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%



(a) Query 203 - Tag Summary

(b) Query 203 - Incoming Summary

(c) Query 223 - Tag Summary

(d) Query 223 - Incoming Summary

Fig. 4. Comparative effectiveness of *TReX* using EvalJ among other INEX 2005 participants

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. These results are promising but not definitive. In the future, we hope to address this issue with more broad-ranging experimental results. 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. (2005) http://inex.is.informatik.uni-duisburg.de:2005
2. EvalJ: INEX evaluation package (2006) http://evalj.sourceforge.net
3. Al-Khalifa, S., Yu, C., Jagadish, H.V.: Querying structured text in an XML databases. In: Proc. SIGMOD Conf., pp. 4–15 (2003)
4. Amer-Yahia, S., Botev, C., Shanmugasundaram, J.: TeXQuery: a full-text search extension to XQuer. In: Proc. WWW Conf., pp. 583–594 (2004)
5. Amer-Yahia, S., Lakshmanan, L.V.S., Pandit, S.: FleXPath: flexible structure and full-text querying for XML. In: Proc. SIGMOD Conf., pp. 83–94 (2004)
6. Barta, A., Consens, M.P., Mendelzon, A.O.: Benefits of path summaries in an xml query optimizer supporting multiple access methods. In: Proc. VLDB Conf., pp. 133–144 (2005)
7. Clark, J., DeRose, S.: XML Path Language (XPath) version 1.0. (1999) http://www.w3.org/TR/xpath
8. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: A semantic search engine for XML. In: Proc. VLDB Conf., pp. 45–56 (2003)
9. Consens, M.P., Milo, T.: Optimizing queries on files. In: Proc. SIGMOD Conf., pp. 301–312 (1994)
10. Robertson, et al.: Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In: Proc. SIGIR Conf., pp. 232–241 (1994)
11. Fagin, et al.: Optimal aggregation algorithms for middleware. In: Proc. PODS Conf., pp. 102–113 (2001)
12. Goldman, R., Widom, J.: Dataguides: Enabling query formulation and optimization in semistructured databases. In: Proc. VLDB Conf., pp. 436–445 (1997)
13. Guo, L., et al.: XRANK: Ranked keyword search over XML documents. In: Proc. SIGMOD Conf., pp. 16–27 (2003)
14. Hristidis, V., Papakonstantinou, Y., Balmin, A.: Keyword proximity search on XML graphs. In: Proc. ICDE Conf., pp. 367–378 (2003)
15. Kaushik, et al.: Covering indexes for branching path queries. In: Proc. SIGMOD Conf., pp. 133–144 (2002)
16. Kaushik, et al.: Exploiting local similarity for indexing paths in graph-structured data. In: Proc. ICDE Conf., pp. 129–140 (2002)
17. Kaushik, et al.: On the integration of structure indexes and inverted lists. In: Proc. SIGMOD Conf., pp. 779–790 (2004)
18. Lu, W., Robertson, S.E., MacFarlane, A.: Field-weighted xml retrieval based on bm25. In: Proc. INEX Workshop, pp. 161–171 (2006)

19. Malik, S., et al.: Overview of INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, Springer, Heidelberg (2006)
20. Marian, A., Amer-Yahia, S., Koudas, N., Srivastava, D.: Adaptive processing of top-k queries in XML. In: Proc. ICDE Conf., pp. 162–173 (2005)
21. Milo, T., Suciu, D.: Index structures for path expressions. In: Proc. ICDT Conf., pp. 277–295 (1999)
22. Rizzolo, F., Mendelzon, A.O.: Indexing XML data with ToXin. In: Proc. WebDB Workshop, pp. 49–54 (2001)
23. Schlieder, T., Meuss, H.: Querying and ranking XML documents. Journal of the American Society for Information, Science and Technology 53(6), 489–503 (2002)
24. Theobald, M., Schenkel, R., Weikum, G.: An efficient and versatile query engine for TopX search. In: Proc. VLDB Conf., pp. 625–636 (2005)
25. Trotman, A., Sigurbjornsson, B.: Narrowed extended XPath I (NEXI). In: Proc. INEX Workshop, pp. 16–39 (2004)

Evaluating Structured Information Retrieval and Multimedia Retrieval Using PF/Tijah

Thijs Westerveld[1], Henning Rode[2], Roel van Os[2], Djoerd Hiemstra[2], Georgina Ramírez[1], Vojkan Mihajlović[2], and Arjen P. de Vries[1]

[1] CWI, Amsterdam, The Netherlands
[2] University of Twente, Enschede, The Netherlands

Abstract. We used a flexible XML retrieval system for evaluating structured document retrieval and multimedia retrieval tasks in the context of the INEX 2006 benchmarks. We investigated the differences between article and element retrieval for Wikipedia data as well as the influence of an elements context on its ranking. We found that article retrieval performed well on many tasks and that pinpointing the relevant passages inside an article may hurt more than it helps. We found that for finding images in isolation the associated text is a very good descriptor in the Wikipedia collection, but we were not very succesful at identifying relevant multimedia fragments consisting of a combination of text and images.

1 Introduction

CWI and the University of Twente 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 [3], a system for flexible information retrieval from structured document collections. PF/Tijah integrates NEXI based IR functionality and 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. Second, we experimented with context weighting. Third, we looked at approaches to identify good entry-points in relevant articles for the AllInContext and BestInContext tasks.

For the multimedia track, we did not do any image processing, all our approaches are purely text and context based. For the Multimedia fragments task (MMfragments), we extended the Wikipedia collection with the metadata from the image collection. Also, we made sure any submitted result contained at least one image. We experimented with query variants based on just the title terms of the distributed topics, as well as with extending this with terms originating from castitle's or image examples.

The remainder of this paper is organised as follows. First, we introduce the PF/Tijah system in Section 2. Then, Sections 3 and 4 discuss our approaches and results for the Ad Hoc and Multimedia tracks. The paper ends with conclusions in Section 5.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 104–114, 2007.

2 The PF/Tijah System

PF/Tijah is a research project run by the University of Twente with the goal to create a flexible environment for setting up search systems. By integrating the PathFinder (PF) XQuery system [2] with the Tijah XML information retrieval system [4] it combines database and information retrieval technology. The PF/Tijah system is part of the open source release of MonetDB/XQuery developed in cooperation with CWI Amsterdam and the University of München. The system is available from SourceForge.

PF/Tijah includes out-of-the-box solutions for common tasks like index creation, stemming, stopword removal, and result ranking for structured queries (supporting several retrieval models), but it remains the same time open to any adaptation or extension.

The PF/Tijah system has a number of unique features that distinguish it from most other open source information retrieval systems:

- It supports retrieving arbitrary parts of the textual data, unlike traditional information retrieval systems for which the notion of a document or fields need to be defined up front at indexing time. A query can simply ask for any XML tag-name as the unit of retrieval without the need to re-index the collection.
- The system allows complex scoring and ranking of the retrieved results by directly supporting the NEXI query language.

```
return
pf:tijah-query($root, "//html[about(.,IR DB)]//p[about(.,XML)]")
```

- PF/Tijah embeds NEXI queries as functions in the XQuery language. This way the system supports ad hoc result presentation by means of its query language. For instance, when searching for a special issue of a journal, it is easy to print any information from that retrieval result on the screen in a declarative way (i.e., not by means of a general purpose programming language), such as the special issue title, its date, the editors and the preface. This is simply done by means of XQuery element construction. As another example, we can formulate a query that performs a whole INEX run and gathers the results in the required output format:

```
for $topic in doc("/INEX/topics2006.xml")//inex_topic
let $result := tijah-query-id($c, $topic/castitle/text())
return
<topic topic-id=.{$topic/@topic_id}">
{ for $r in tijah-nodes($result) return
<result>
<file> { $r/name/@id } </file>
<path> { local:getINEXPath($r) } </path>
<rsv> { tijah-score($result, $r) } </rsv>
</result> }
</topic>
```

– PF/Tijah supports text search combined with traditional database querying, including for instance joins on values. For instance, one could formulate the difficult INEX topic 14 from 2002 in the following way:
Find figures that describe the Corba architecture and the paragraphs that refer to those figures. Retrieved components should contain both the figure and the paragraph referring to it.

```
let $doc := doc("inex.xml")
for $p in tijah-query($doc, "//p[about(.,corba architecture)]")
for $fig in $p/ancestor::article//fig
where $fig/@id = $p//ref/@rid
return <result> { $fig, $p } </result>
```

Recent developments in the PF/Tijah search module mainly concerned stability, scalability and performance. We can index the current Wikipedia collection in 25 to 30 minutes on a 64 bits machine with a 2Ghz Opteron processor and 8 Gb of memory running Fedora Core 6. Querying times are shown in the following table:

Simple article query `//article[about(.,X)]` (top 10, ranking only)	2 sec
Full INEX `//article[about(.,X)]` query (top 1500, INEX results format)	28 sec
Full INEX `//*[about(.,X)]` query (top 1500, INEX results format)	141 sec
Complete INEX run	

3 Ad Hoc Track

The characteristics of the Wikipedia collection differ considerably from the IEEE collection used before. This inspired us to test some ideas that seem in particular suitable for this new collection, but also to revisit some of the approaches that were successful on IEEE and to test how well these techniques work on a very different set of documents. We studied element vs. article retrieval, context weighting and the entry-point tasks, each of these is discussed in a separate subsection below, but first we discuss our general approach.

3.1 Approach

For all our submissions, we employed the PF/Tijah system. We indexed the entire Wikipedia collection, without removing any structural information. The index was stemmed and stopwords were removed. All our submissions are based on the XML variant of the unigram language modelling approach to information retrieval [4]. Elements are scored based on a mixture of foreground or document statistics and background or collection statistic. When we need to combine information from different levels in the hierarchy, for example for queries like `//article[about(.,X)]//*[about(.,Y)]` we use a product of the element scores. All elements always have a score greater than zero because of the background statistics. Therefore, the product based combination functions as a weak AND.

We believe it is useless to show the same information twice to a user, and thus removed overlap from all our runs. As a result, all our runs can be interpreted as submissions for the Focused task and that's how we evaluate them below. Our focused runs are constructed from (thorough) pure language modelling runs, by iteratively going down the ranked lists and removing all lower ranked ancestor and descendant of a retrieved element.

3.2 Element vs. Article Retrieval

The IEEE collection contains mainly lengthy documents where it makes sense to retrieve parts rather than the whole document. For the Wikipedia collection this is the case to a much lesser extend. Wikipedia documents tend to be short and focused. Moreover, a substantial amount of real Wikipedia queries has a corresponding Wikipedia document whose title matches the query exactly. This indicates that for many queries, an entire document may be a good answer. We investigated this by comparing document retrieval and element retrieval approaches. To this end, we ran two types of title only queries against our PF/Tijah system:

- `//article[about(.,$title)]`
- `//*[about(.,$title)]`

Where $title is replaced by the terms from the `<title>` field. The results for these runs on the Focused task are shown in Figure 1.

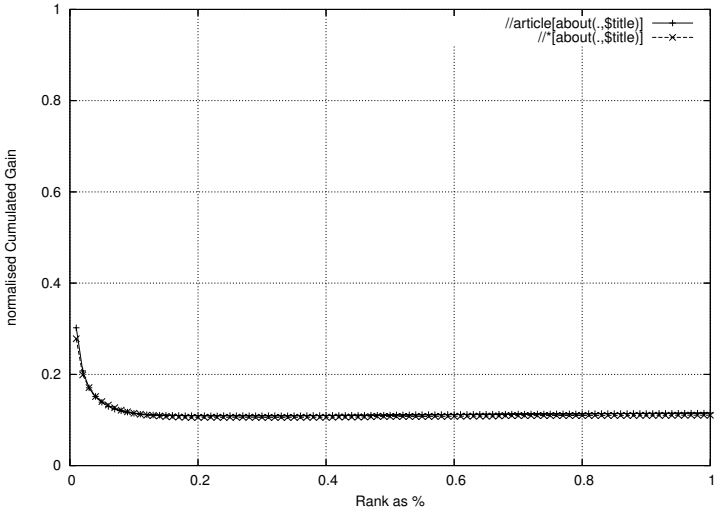


Fig. 1. Element vs. Article retrieval, normalised Cumulated Gain (overlap: on, quantisation: strict)

The runs are indistinguishable, indicating it makes no difference whether we retrieve full articles, or unrestricted elements. If we look closer at the retrieved elements, this makes sense. Figure 2a shows the element types we retrieved most in the element run. Almost half of the elements we retrieve are either *body* or *article*, and thus very close to full documents. Another 11% of the retrieved elements are of type *collectionlink*, and point indirectly to another article. We did not exploit this indirection, but these numbers indicate that effectively our element run was mainly pointing at full documents rather than smaller parts. This does not mean element retrieval is useless in this collection, though. Many of the relevant elements are of a much finer granularity, 45% of the relevant items are either paragraphs (*p*) or sections (*section*) (see Figure 2b).

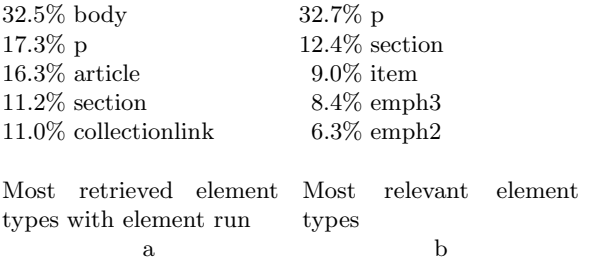


Fig. 2. Element types. Most retrieved in element run (`//*[about(.,$title)]`) (a) and most relevant (b) element types

3.3 Context Weighting

Previous INEX results have shown that it is important to take an element's context into account. In particular article weighting has been successful [1,4]. Article weighting takes the article context of an element into account; good elements in good articles will be ranked higher than good elements in bad articles. We investigated whether article weighting is useful for element retrieval in the new Wikipedia collection.

In addition, articles in wikipedia have a clear and concise title, which may help in identifying relevant articles. We experimented with using this name together with the article content for article retrieval (and as a side-effect for setting the context for element retrieval).

We compared article retrieval with and without names[1]:

- `//article[about(.,$title)]`
- `//article[about(.,$title) OR about(.//name,$title)]`

And we experimented with article weighting:

- `//*[about(.,$title)]`
- `//article[about(.,$title) OR about(.//name,$title)]//*[about(.,$title)]`

[1] A limited preliminary study indicated that a disjunctive combination of article content and name performs better than a conjunctive combination.

Figure 3 and 4 show the results for name and article context respectively. Context and article weighting appear to have no influence on retrieval effectiveness. Thus context appears to be less influential than in the IEEE collection.

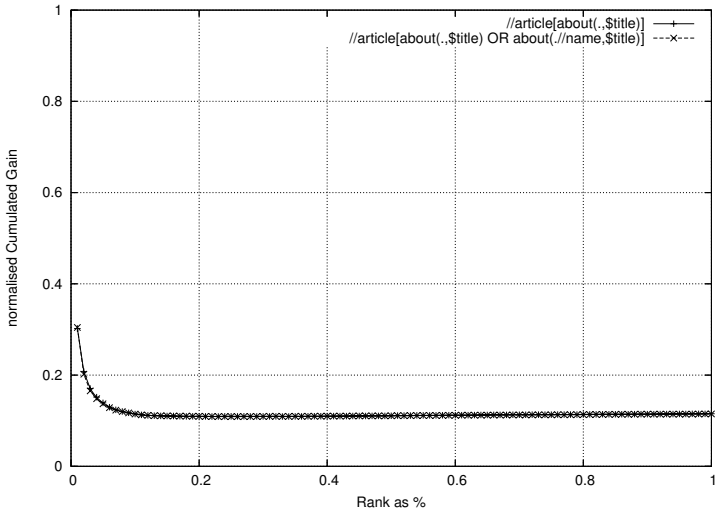


Fig. 3. Normalised Cumulative Gain for article and article OR name runs

3.4 Entrypoint Tasks

We submitted some basic runs for both the AllInContext and the BestInContext tasks. These submissions started from top 1500 focused runs without overlap.

For AllInContext we grouped the retrieved elements by article and scored the articles by aggregating the element scores (for top 1500 elements only). We experimented with both maximum and average as aggregation functions. Within each article, we simply kept all elements that made it to the top 1500 of the original focused run. Thus, our AllInContext runs are nothing more than a re-ordering of the Focused run.

For BestInContext we did something similar. Again, we group by article, but now we order articles by the sum of the element scores, since we want articles with a lot of good material on top. Our assumption is a user wants to see all relevant passages in an article, thus as a best entry point, we simply returned the first element in document order that was in the focused top 1500. We did not extend our best entry point run with articles that did not made it to the focused top 1500, thus our BestInContext runs typically contain fewer than the allowed 1500 entry-points.

The approach was applied to both the plain and article weighted elements runs. Figure 5 shows our results in the AllinContext task. The two element runs (ARTorNAME_STAR_AVG and ARTorNAME_STAR_MAX) are indistinguishable. The article run (ARTorNAME) is clearly better. Apparently, in this

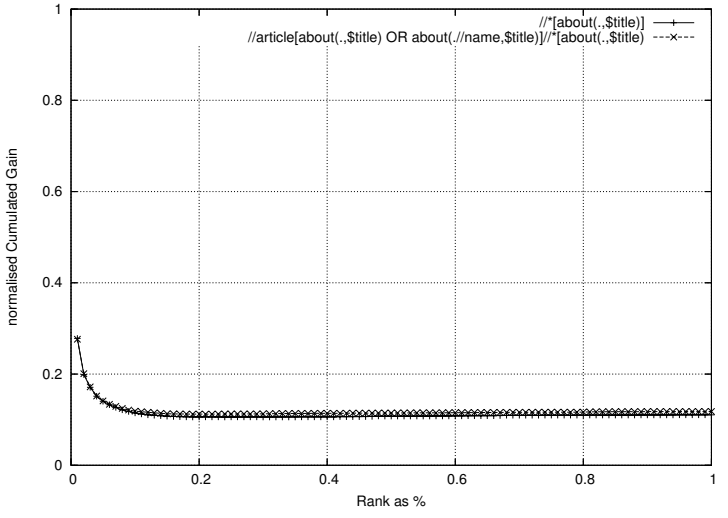


Fig. 4. Normalised Cumulative Gain for //* runs with and without article weighting

collection, selecting the best elements inside an article is still a hard task, and the best strategy is to simply return the entire article. Similar results were found on the BestInContext task:

4 Multimedia Track

The CWI/Utwente participation in the multimedia track is without any image processing; all our submitted runs are purely text based. Still, we adapted our runs to cater for the multimedia information needs. We augmented the Wikipedia collection with metadata, we filtered the results for images and we added some knowledge about the example images to our queries. Below these approaches are discussed in detail.

4.1 Augmenting the Collection with Image Metadata

The MMfragments is similar to the Ad Hoc track in that it asks for fragments from the Wikipedia collection. The main difference is that the information needs in this task have a clear multimedia character. The idea is that a system would return fragments containing relevant images together with relevant text. The PF/Tijah system and the Language Models used are designed for returning relevant (structured) text. To be able to work with images, we tried to get extra textual information in, to help us decide which are the relevant images. We did this by adding text from the image metadata document as available in the Multimedia images (MMimages) collection. Each $< image >$ tag in the collection is augmented with the corresponding metadata from the MMimages collection. We did not try to separate names, users and captions, we simply added the contents of the entire metadata document as text under the image tag.

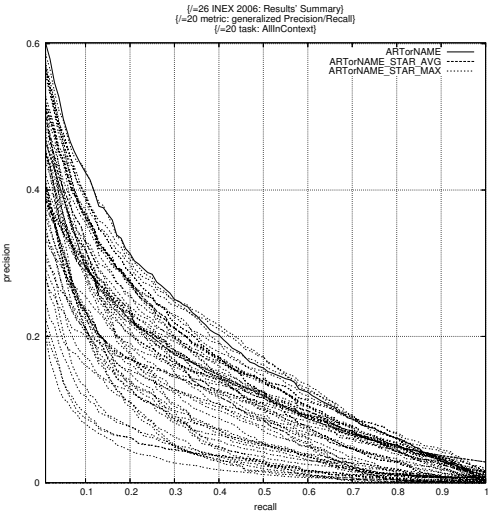


Fig. 5. AllinContext, generalised precision/recall: cwi/utwente compared to the competition

4.2 Filtering Results

Since the MMfragments deals with multimedia information needs, it seems wise to return only fragments that contain images. We made sure this was the case by filtering our results. Not all $< image >$ tags in the Wikipedia correspond to images that are actually part of the INEX multimedia collections; images that are not part of these collections will not be visible to users during assessments. Therefore, we also removed all results that contained references to images that are not in the collection. This way, we made sure all our returned fragments contain at least one *visible* image from the multimedia collections.

4.3 Experiments

We participated in the MMfragments and MMimages tasks. For both tasks, we experimented with relatively simple text only queries, aiming to show that text only queries can be competitive. Below we discuss our experimental results.

MMfragments. For MMfragments we submitted one full article run and three element runs. For the element runs, we did not directly use the given *castitle*, but we experimented with runs of the form: `//*[about(.,X)]`, where X contained some set of terms taken from the topic. We experimented with the following sets of terms:

STAR_TITLE the *title* field

CAS_noMM the terms from the *castitle* field without the visual examples and concepts

CAS_MMtext the terms from the *castitle* field plus the textual terms from the example images metadata documents (again no concepts).

The article run (*ART_TITLE*), was based on the title field of the topic only.

These queries were run against the augmented collection and the results were filtered to make sure all returned fragments contain images from the collection. The results are very disappointing; with mean average effort precision values of around 0.001. The main reason for this is that, like in the Ad Hoc task, we remove overlap from the results. This means we submitted Focused runs, while the results are evaluated using the thorough setting. The results for our thorough runs that still contain overlap show median performance, see Figure 6.

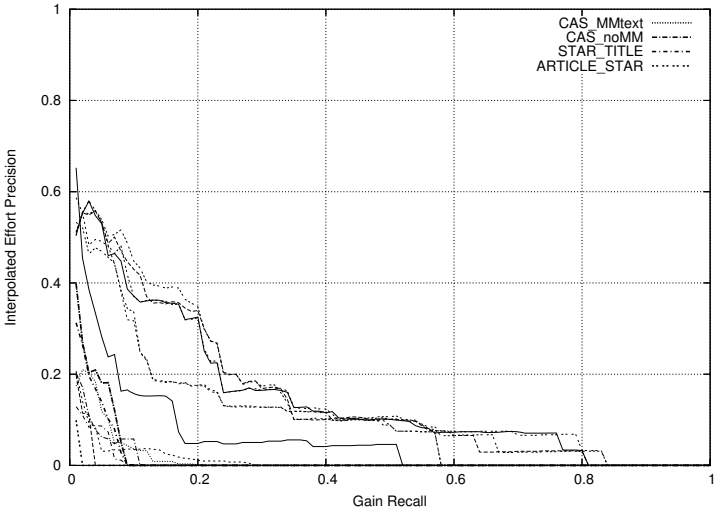


Fig. 6. MMfragments results: cwi/utwente runs compared to competition

MMimages For MMimages, we submitted two very basic runs:

article-title An article run, only using the title field of the topic:
`article[about(.,$title)]`

cas-noMM A *castitle* run ignoring the visual hints, we removed all image examples and concept references from the NEXI query.

Figure 7 shows these basic runs give highly competitive performance on the MMimages task. Clearly it is hard to beat a text only baseline on this task.

5 Conclusion

PF/Tijah was used at INEX for the first time. The flexibility of the system and the ease with which it allows the modification and combination of XML data

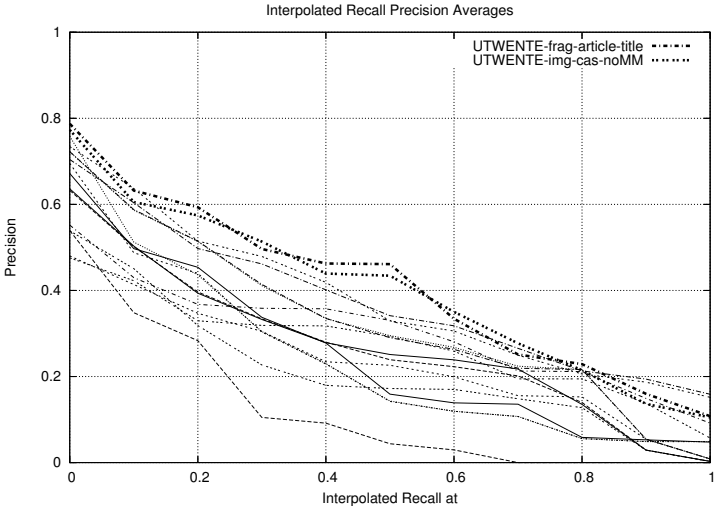


Fig. 7. MMimages results: cwi/utwente runs compared to the competition

makes it a very useful tool for (XML) IR experiments. For short article only queries, the system is reasonably fast; for larger queries it takes a bit longer, but for IR experiments, this is still acceptable.

Without adapting the models we used for the IEEE collection in previous years, the unigram language modelling approach retrieves elements that are often (almost) complete articles or links to them. Moreover, for the AllInContext and BestInContext tasks, the tasks most close to a realistic setting and to the way the assessment procedure is set-up, retrieving complete articles rather than smaller elements appears to be a useful strategy. Still, among the known relevant elements are many smaller elements like paragraphs and sections. Perhaps these smaller relevant elements appear in clusters. Further study is needed to investigate whether this explains the success of the article retrieval strategies.

Our text only approach to multimedia retrieval was very successful on the MMimages task, but less so on the MMfragments task. Perhaps a smarter way of filtering the results is needed to retrieve the appropriate multimedia fragments.

References

1. Arvola, P., Junkkari, M., Kekäläinen, J.: Generalized contextualization method for xml information retrieval. In: CIKM '05. Proceedings of the 14th ACM international conference on Information and knowledge management, New York, NY, pp. 20–27. ACM Press, New York, NY (2005)
2. Boncz, P., Grust, T., van Keulen, M., Manegold, S., Rittinger, J., Teubner, J.: Monetdb/xquery: a fast xquery processor powered by a relational engine. In: SIGMOD '06. Proceedings of the 2006 ACM SIGMOD international conference on Management of data, New York, NY, pp. 479–490. ACM Press, New York, NY (2006)

3. Hiemstra, D., Rode, H., van Os, R., Flokstra, J.: Pftijah: text search in an XML databases system. In: Proceedings of the 2nd International Workshop on Open Source Information Retrieval (OSIR) (2006)
4. List, J., Mihajlovic, V., Ramirez, G., de Vries, A., Hiemstra, D., Blok, H.: Tijah: Embracing ir methods in xml database. Information Retrieval 8(4), 547–570 (2005)

EXTIRP: Baseline Retrieval from Wikipedia

Miro Lehtonen[1] and Antoine Doucet[1,2]

[1] Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI–00014 University of Helsinki
Finland
{Miro.Lehtonen,Antoine.Doucet}@cs.helsinki.fi
[2] IRISA-INRIA
Campus de Beaulieu
F-35042 Rennes Cedex
France
Antoine.Doucet@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 which 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 [1] — 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 to 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 in that there was no tuning one way or another, we did not analyse the Wiki documents before they were indexed and queried for the official submissions. We also have left out many of the characteristic features

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 115–120, 2007.

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.

This article is organised as follows. The anatomy of EXTIRP is described in Section 2. The selection of the index units is explained in Section 3. The results are analysed in Section 4 and finally conclusions are drawn in Section 5.

2 Background

EXTIRP scans through the document collection and selects disjoint fragments of XML to be indexed as atomic units. Typical fragments include XML elements marking sections, subsections, and paragraphs. Examples and more details about the selection algorithm are included in Section 3. The disjoint fragments are treated as traditional documents which are independent of each other. The pros include that the traditional IR methods apply, so we use the vector space model with a weighting scheme based on the tf*idf. The biggest of the cons is that the size of the indexed fragments is static, and if bigger or smaller answers are more appropriate for some query, the fragments have to be either divided further or combined into bigger fragments.

Two separate inverted indices are built for the fragments. A *word index* is created after punctuation and stopwords are removed and the remaining words are stemmed with the Porter algorithm [2]. The *phrase index* is based on Maximal Frequent Sequences (MFS) [3]. Maximal phrases of two or more words are stored in the phrase index if they occur in seven or more fragments. The threshold of seven comes from the computational complexity of the algorithm. Although lower values for the threshold produce more MFSs, the computation itself would take too long to be practical. More details concerning the configuration of the phrase index are included in the PhD thesis of Antoine Doucet [4].

When processing the queries, we compute the cosine similarity between the document and the base term vectors which results in a `Word_RSV` value. In a similar fashion, each fragment vector gets a similarity score `MFS_RSV` for phrase similarity. These two scores are aggregated into a single RSV so that the aggregated RSV = α * `Word_RSV` + β * `MFS_RSV`, where α is the number of distinct query terms and β is the number of distinct query terms in the query phrases.

3 Selective Indexing

The selection of indexed fragments is based on two parameters: fragment size (min and max) and the proportion of Text and Element nodes (T/E measure) [5]. The algorithm starts from the document root and traverses the document tree in document order. The following steps are then iterated:

1. If the element is too big, move on to the next node and start over (from 1).
2. If the content looks like structured data (T/E<1.0), move on to the next node and start from 1.

3. If the element is too small, skip the subtree, move on to the next node and start from 1.
4. Index the element as an atomic unit, skip the subtree, move on to the next node and start from 1.

The resulting fragment collection does not cover the whole document collection. For example, parts of the documents that consist mostly of elements are discarded. Previous experiments on IEEE articles have shown that the algorithm works: it reduces the index size and improves retrieval precision. When tested with the article collection, bibliographic and other data were successfully excluded from the full-text index [6]. Therefore, the Wikipedia XML documents were an interesting challenge for our algorithm.

Figure 1 shows the document with the lowest T/E value in the Wikipedia XML collection. The nested `cadre` elements are there either because of a faulty conversion from the Wiki format into XML or because of inconsistency in the source data. Because of the extra elements, the text content of this document was not included in the full-text index of EXTIRP, and thus it could not be retrieved, regardless of the query. However, the nested structures created with the proliferating XML elements are highly artificial. Therefore, it is questionable to exclude text content from the full-text index because of such artificial structures.

Our observations raise an interesting question: What is the validity of this evaluation at INEX, where the test documents can only be used in the evaluation because the structure is completely useless elsewhere?

4 Results

The results from INEX 2005 showed that the official evaluation metrics [7] do not favour systems like EXTIRP because there is no reward for returning "too small" answers. The 2005 version of EXTIRP could not adjust the granularity of the answers according to the query, but the granularity came directly from the indexed document fragments [8]. The 2006 version of EXTIRP comes with the same drawback even though some "near misses" are rewarded.

In 2006, we only submitted one run for the CO.Focused task. The fragment size in the index was limited to the range of 150–7,000 characters of text content. Only the title and the keyword part of the queries were considered. The overall poor results according to the official metrics are shown in Table 1.

The poor overall performance has several possible explanations. First, all answers shorter than 500 bytes of XML markup were discarded because of the assumption that short answers are not worth retrieving. This assumption no longer holds as the official metrics reward short answers, too, as long as they are relevant to the query. We also observe that the rankings with the filtered assessments are systematically better than those with the original assessments that include very short relevant answers.

Second, EXTIRP has always had a better performance with the strict quantisation of the assessments than with the generalised one which was the only

```
            </cadre>
           </cadre>
          </cadre>
         </cadre>
        </cadre>
       </cadre>
      </cadre>
     </cadre>
    </cadre>
   </cadre>
  </cadre>
 </cadre>
</cadre>
1987 "Recent Acquisitions", Tel Aviv Museum of Art
<cadre>
 <cadre>
  <cadre>
   <cadre>
    <cadre>
     <cadre>
      <cadre>
       <cadre>
        <cadre>
         <cadre>
          <cadre>
           <cadre>
            <cadre>
             <cadre>
              <cadre>
               <cadre>
                <cadre>
                 <cadre>
                  <cadre>
                   <cadre>
"Recent Acquisitions", Isreal Museum, Jerusalem, Isreal "20 Years of Occupation",
Bograshov Gallery, Tel Aviv, Israel (catelogue) "New Israeli Realism", traveling
exhibition, Omanut La'am (catelogue) "Artisrael, the 1980's", Leonard Perlson Gallery,
New YorkCity, "Haifa, Portrait of the City", Haifa Museum, Haifa Israel (catelogue)
"Fresh Paint", Tel Aviv Museum of Art (catelogue)
                   </cadre>
                  </cadre>
                 </cadre>
                </cadre>
               </cadre>
              </cadre>
             </cadre>
            </cadre>
           </cadre>
          </cadre>
         </cadre>
        </cadre>
       </cadre>
      </cadre>
     </cadre>
    </cadre>
   </cadre>
  </cadre>
 </cadre>
1989 International Biennale of Graphic Art, Ljublijana,
Yugoslavia (catelogue)
<cadre>
 <cadre>
  <cadre>
```

Fig. 1. An excerpt from the document '`3125748.xml`' with a T/E value of 0.14

Table 1. The submission "UHel_Run1" measured with nxCG, generalised quantisation. A total of 85 submissions are included in the ranking.

	Overlap on		Filtered assesments		Overlap off		Filtered assesments	
Cutoff	Rank	Score	Rank	Score	Rank	Score	Rank	Score
MAP5	67	0.2316	65	0.2299	70	0.2371	68	0.2371
MAP10	70	0.1818	69	0.1821	71	0.1898	69	0.1898
MAP25	73	0.1295	71	0.1315	73	0.1358	71	0.1358
MAP50	74	0.0960	73	0.1001	73	0.0990	72	0.0992

quantisation in 2006. Third, what EXTIRP assumes of the quality of XML is based on observations of real XML documents that are usable outside the context of INEX. For example, we assume that the XML structure of the documents is designed before any content is converted into that structure and that the XML documents have a real use case instead of only being test material for researchers.

Despite the relatively poor performance, the results do show some signs of stability in the performance of EXTIRP. Along the lines of the previous INEX results from 2004 and 2005, the relative ranking of EXTIRP decreases as the cutoff value increases. This supports the earlier observation that EXTIRP is more geared towards high precision tasks than high recall ones.

5 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. In the future, we are hoping to have a fully implemented version of our system in order to see where it really stands. We also look forward to experimenting with more realistic document collections in order to increase the validity of the results.

References

1. Doucet, A., Aunimo, L., Lehtonen, M., Petit, R.: Accurate Retrieval of XML Document Fragments using EXTIRP. In: INEX, Workshop Proceedings, Schloss Dagstuhl, Germany, pp. 73–80 (2003)
2. Porter, M.F.: An algorithm for suffix stripping. Program 14, 130–137 (1980)
3. Ahonen-Myka, H.: Finding all frequent maximal sequences in text. In: Mladenic, D., Grobelnik, M., (eds.) Proceedings of the 16th International Conference on Machine Learning ICML-99 Workshop on Machine Learning in Text Data Analysis, Ljubljana, Slovenia, J. Stefan Institute, pp. 11–17 (1999)
4. Doucet, A.: Advanced Document Description, a Sequential Approach. PhD thesis, University of Helsinki (2005)
5. Lehtonen, M.: Preparing heterogeneous XML for full-text search. ACM Trans. Inf. Syst. 24, 455–474 (2006)

6. Lehtonen, M.: Indexing Heterogeneous XML for Full-Text Search. PhD thesis, University of Helsinki (2006)
7. Kazai, G., Lalmas, M.: INEX 2005 Evaluation Measures. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 16–29. Springer, Heidelberg (2006)
8. Lehtonen, M.: When a few highly relevant answers are enough. [9] 296–305
9. Fuhr, N., Lalmas, M., Malik, S., Kazai, G., (eds.): Advances in XML Information Retrieval and Evaluation (Revised Selected Papers). In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, Springer, Heidelberg (2006)

Filtering and Clustering XML Retrieval Results

Jaap Kamps[1,2], Marijn Koolen[1], and Börkur Sigurbjörnsson[2,3]

[1] Archives and Information Science, Faculty of Humanities, University of Amsterdam
[2] ISLA, Faculty of Science, University of Amsterdam
[3] Yahoo! Research, Barcelona

Abstract. As part of the INEX 2006 Adhoc Track, we conducted a range of experiments with filtering and clustering XML element retrieval results. Our basic retrieval engine retrieves arbitrary elements from the collection (corresponding to the Thorough Task). These runs are filtered to remove textual overlap between elements (corresponding to the Focused Task). The resulting runs can be clustered per article (corresponding to the All in Context Task). Finally, we select the "best" element for each article (corresponding to the Best in Context Task). 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 non-overlapping 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 overall motivation for INEX 2006 was 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.

During INEX 2006, we conducted a range of experiments with filtering and clustering XML element retrieval results. Our basic retrieval engine retrieves arbitrary elements from the collection (corresponding to the Thorough Task). These runs are filtered to remove textual overlap between elements (corresponding to the Focused Task). The resulting runs can be clustered per article (corresponding to the All in Context Task). Finally, we select the "best" element for each article (corresponding to the Best in Context Task).

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 topics and assessments of the INEX 2006 Adhoc Track. In four separate sections, we report our experiments

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 121–136, 2007.

Table 1. Wikipedia collection statistics

Description	Statistics
# of articles	659,388
# of elements	52,555,826
# of unique tags	1,241
Avg. # of elements per article	79.69
Average depth	4.82

for the Thorough Task (§5); the Focused Task (§6); the All in Context Task (§7); and the Best in Context Task (§8). Experiments with a mixture language model are discussed in Section 9. Finally, in Section 10, we discuss our findings and draw some 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 1,241 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.

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.

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 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 2. Longest elements in Wikipedia collection

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

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

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.[1] 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 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 containing only the most 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, and we build an index containing just these elements. The frequent element left out is `<collectionlink>`. 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 e according to:

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

[1] 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).

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	289

Table 6. Relevant element statistics

Tag name	Frequency	Avg. length
`<p>`	15,315	450
`<item>`	14,375	91
`<emph2>`	14,322	20
`<cell>`	13,898	19
`<section>`	9,291	2,375
`<emph3>`	5,782	16
`<article>`	5,481	9342
`<body>`	5,479	9322
`<title>`	5,197	21
`<template>`	4,107	87

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

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

where $P_{mle}(\cdot|e)$ is a language model for element e in document d; $P_{mle}(\cdot|d)$ 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). None of our official submissions used the three layered mixture model proper, i.e., we use $\lambda_d = 0$ unless indicated otherwise. The default value of λ_e is 0.15.

Finally, we assign a prior probability to an element e relative to its length in the following manner:

$$P(e) = \frac{|e|^\beta}{\sum_e |e|^\beta}, \qquad (3)$$

where $|e|$ is the size of an element e. The β parameter introduces a length bias which is proportional to the element length with $\beta = 1$ (the default setting). For a more thorough description of our retrieval approach we refer to [4]. For comprehensive experiments on the earlier INEX data, see [7].

4 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

Table 7. Elements containing entire relevant passages

Tag name	Frequency
<p>	2,585
<body>	1,639
<section>	1,326
<item>	937
<article>	724
<normallist>	301
<name>	267
<collectionlink>	208
<row>	180
<caption>	174

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 short. The lengths of the elements are measured in characters (text offset).

Table 6 looks at the judgments from the vista point of elements containing only relevant text. After the workshop it was decided that the links in the collection are too small to be relevant. This affects the elements <collectionlink>, <outsidelink>, <redirectlink>, <unknownlink>, <weblink>, <wikipedialink>. This has quite some effect on the set of relevant elements, for example, in the original qrels there were no less than 78,792 <collectionlink> elements highlighted by the assessors. Although the links are no longer considered exhaustive, there are still quite a number of small elements left, like <emph2>, <cell> and <emph3> (see Table 3). The lengths mentioned are the average lengths of the *relevant* elements of that type. Longer elements containing relevant text are mostly <p>, <item> and <section> elements.

The shorter elements often contain only a few words, and often are only a small part of the entire passage. However, there are still a fair number of elements that encompass an *entire* relevant passage. Table 7 shows the frequency of elements that are the shortest element to contain an entire relevant passage. There are 208 passages that are fully contained within a <collectionlink> element, and 43 more passages contained by other link elements that are considered too small to be relevant. The passages are not considered too small, so retrieving a larger element containing the passage still gives some score. Relevance is often found at the paragraph level. 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 Experiments for the Thorough Task

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

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

Run	MAep	nxCG@5	nxCG@10	nxCG@25	nxCG@50
`thorough_element_lm`	0.0471	0.4120	0.3789	0.3262	0.2790
`thorough_section_lm`	0.0431	0.3948	0.3721	0.2977	0.2503
`thorough_element_lm_cas.seperate`	0.0265	0.2124	0.1761	0.1511	0.1208
`thorough_element_lm_cas.joined`	0.0222	0.1872	0.1642	0.1410	0.1100

`<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.

For all submitted runs we used the default length prior parameter $\beta = 1$, although XML retrieval may require special length normalization [8]. Since we're dealing with a new collection, it is interesting to see whether the different parameter settings of the length prior show the same effect on this new collection. We also experimented with different λ_e and β setting to see what works, and what doesn't.

5.1 Results

We will 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 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

Table 9. Results for the Thorough Task (generalized, off) for different parameter settings

Run	MAep	nxCG@5	nxCG@10	nxCG@25	nxCG@50
element $\beta = 0.8, \lambda_e = 0.15$	0.0460	0.3739	0.3383	0.2901	0.2500
element $\beta = 0.9, \lambda_e = 0.15$	0.0465	0.3820	0.3473	0.2940	0.2502
element $\beta = 1.0, \lambda_e = 0.15$	0.0471	0.3868	0.3519	0.2982	0.2508
element $\beta = 1.1, \lambda_e = 0.15$	0.0462	0.3727	0.3317	0.2973	0.2464
element $\beta = 0.8, \lambda_e = 0.30$	0.0462	0.3884	0.3479	0.2966	0.2525
element $\beta = 0.9, \lambda_e = 0.30$	0.0465	0.3897	0.3494	0.3004	0.2533
element $\beta = 1.0, \lambda_e = 0.30$	0.0469	0.3878	0.3512	0.3005	0.2518
element $\beta = 1.1, \lambda_e = 0.30$	0.0465	0.3867	0.3471	0.2995	0.2501

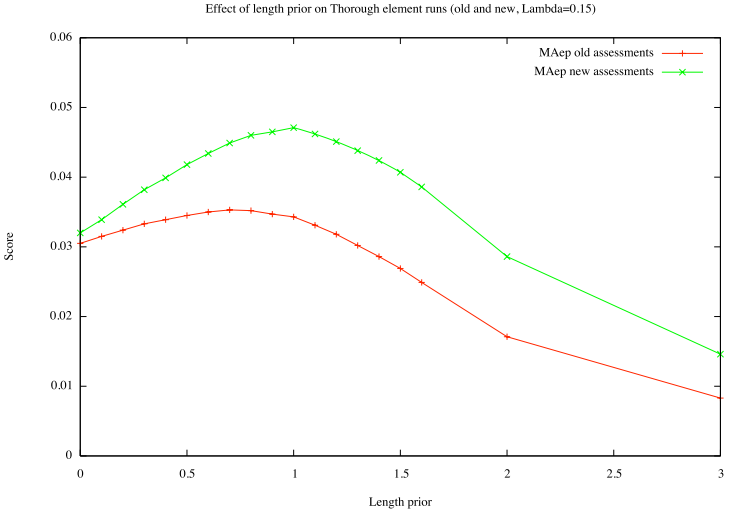


Fig. 1. Effect of the length prior for the Thorough Task using the element index with standard smoothing

filter) performs less good 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.

Table 9 shows the results for non-official runs for the Thorough Task, using different values for the language model parameters. With a higher value of λ_e we see the expected gain of precision and loss of recall (or MAep). Also, with a higher λ_e value, the length prior parameter β has less effect. The scores for the $\lambda_e = 0.30$ runs are very similar. With the standard $\lambda_e = 0.15$ the standard $\beta = 1.0$ gives the best results.

Table 10. Results for the Focused Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
`focused_element_lm`	0.3337	0.2948	0.2252	0.1781	0.0140
`focused_section_lm`	0.3386	0.2868	0.2212	0.1834	0.0152
`focused_element_lm_cas.seperate`	0.2845	0.2288	0.1634	0.1159	0.0080
`focused_element_lm_cas.joined`	0.2400	0.1981	0.1409	0.1010	0.0069

Figure 1 show the effect of different length prior settings on performance in terms of MAep. We clearly see that the default value $\beta = 1.0$ gives optimal performance with the new assessments. If we compare the new assessments with the old assessments (where the links are considered exhaustive), we see that a lower value of β gives better performance. This can be explained by the huge number of relevant `<collectionlink>` elements in the old assessments. Lower β values retrieve more smaller elements, many of which are `<collectionlink>` elements.

6 Experiments for the Focused Task

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 post-processed 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.

We also look at the impact of different levels of smoothing and of length normalization.

6.1 Results

For the Focused Task, none of the retrieved elements was allowed to contain text that overlaps with another retrieved element. Table 10 shows the results

Table 11. Results for the Focused Task (generalized, off) for different parameter settings

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
element $\beta = 0.8, \lambda_e = 0.15$	0.3315	0.2923	0.2305	0.1843	0.0144
element $\beta = 1.0, \lambda_e = 0.15$	0.3337	0.2948	0.2252	0.1781	0.0140
element $\beta = 1.1, \lambda_e = 0.15$	0.3256	0.2913	0.2208	0.1737	0.0135
element $\beta = 0.5, \lambda_e = 0.30$	0.3353	0.3025	0.2354	0.1885	0.0144
element $\beta = 1.0, \lambda_e = 0.30$	0.3410	0.2920	0.2296	0.1790	0.0139
element $\beta = 1.1, \lambda_e = 0.30$	0.3438	0.2913	0.2264	0.1763	0.0139

for the Focused Task. The results for the official measure (nxCG) are listed first (columns 2 through 5). The runs based on the section and element index show almost the same performance for this task. At ranks 5 and 50, the section index run performs better than the element index run. For the two runs using the structured query, we see again that the run using only the tagname of the target element ("separate") scores better. The CAS query runs are less effective than the CO query runs, although the difference in performance is much smaller than for the Thorough Task before.

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. The Thorough measure MAep is listed in column 6 of Table 10 for comparison. Interestingly, the section index run is outperforming the element index run for the Focused Task. When comparing the Focused Task results to the Thorough Task results, we note that, as expected, the scores are substantially lower. There is a moderate decline for the precision scores, but the recall (and mean average precision) drops dramatically.

Table 11 shows the results for non-official runs for the Focused Task, using different values for the language model parameters. As may be expected, a higher value of λ_e leads to a minor increase in precision. However, a lower value of β leads to a small increase at higher ranks and in MAep. This is unexpected since the corresponding Thorough run is actually inferior to the standard length normalization ($\beta = 1.0$). A possible explanation is the non-overlapping nature of the Focused runs: in case a long element is selected, this immediate outlaws a wide range of other elements. A case in point is when the `<article>` element is selected, which effectively exhausts the whole article.

7 Experiments for the All in Context Task

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:

Table 12. Results for the All in Context Task (generalized precision)

Run	MAgP	gP@5	gP@10	gP@25	gP@50
`all_section_lm.highest`	0.1633	0.3014	0.2633	0.1966	0.1579
`all_section_lm.sum`	0.1062	0.1592	0.1351	0.1283	0.1136

Table 13. Results for the All in Context Task (generalized, off)

Run	MAep	nxCG@5	nxCG@10	nxCG@25	nxCG@50
`all_section_lm.highest`	0.0157	0.3357	0.3082	0.2291	0.1898
`all_section_lm.sum`	0.0112	0.2454	0.2135	0.1805	0.1477

Table 14. Results for the All In Context Task (generalized, off) for different parameter settings

Run	MAgP	gP@5	gP@10	gP@25	gP@50
element $\beta = 1.0, \lambda_e = 0.15$	0.1509	0.2743	0.2450	0.1858	0.1449
element $\beta = 1.2, \lambda_e = 0.15$	0.1552	0.2895	0.2611	0.1940	0.1468
element $\beta = 1.4, \lambda_e = 0.15$	0.1556	0.3031	0.2628	0.1947	0.1472
element $\beta = 1.5, \lambda_e = 0.15$	0.1532	0.3046	0.2566	0.1934	0.1452
element $\beta = 1.0, \lambda_e = 0.30$	0.1535	0.2881	0.2489	0.1920	0.1486
element $\beta = 1.2, \lambda_e = 0.30$	0.1564	0.3054	0.2590	0.1997	0.1518
element $\beta = 1.4, \lambda_e = 0.30$	0.1579	0.3065	0.2669	0.1993	0.1515
element $\beta = 1.5, \lambda_e = 0.30$	0.1567	0.3088	0.2650	0.1961	0.1508

`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.

7.1 Results

For the All in Context Task, there is the further restriction that retrieved elements must be grouped per article (and still may not overlap). Table 12 shows the results for the All in Context Task. The official measure is the mean average generalized precision in column 2. 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.

Table 13 also evaluates the runs using the same measures as the Thorough and Focused Task above. When comparing the All in Context Task results to the Focused Task results, we see that the clustering by article improves performance for all measures. That is, clustering the elements per article is more effective than ranking them on their own similarity score. This is an unexpected result

Table 15. Results for the Best in Context Task (generalized, off)

Run	A=0.01	A=0.1	A=1	A=10	A=100
best_section_lm.highest_score	0.1237	0.2103	0.3437	0.5365	0.7369
best_section_lm.first	0.1237	0.2103	0.3437	0.5365	0.7369
best_section_lm.article	0.0754	0.1761	0.3027	0.4853	0.7021

Table 16. Results for the Best in Context Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
best_section_lm.highest_score	0.3290	0.2796	0.2083	0.1678	0.0131
best_section_lm.first	0.3290	0.2796	0.2083	0.1678	0.0131
best_section_lm.article	0.2451	0.1983	0.1424	0.1109	0.0085

that may be related to the organization of information in an encyclopedia like Wikipedia.

Table 14 shows the results for non-official runs for the All in Context Task, using different values for the language model parameters. The runs on the element index are all inferior to the official run on the section index. Again, as may be expected, a higher value of λ_e leads to a small increase in precision. In fact, it also results in an increase of MAep. Moreover, a higher value of β also leads to an increase of both precision and MAep. A possible explanation is the generalized precision measure that treats "retrieved articles" as ranks, and the fact that there is usually only a small number of articles with relevance in the Wikipedia collection. This leads to an incentive to ensure that at least those articles are retrieved.

8 Experiments for the Best in Context Task

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 post-process 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.

8.1 Results

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. With high A values the position of the Best-Entry-Point

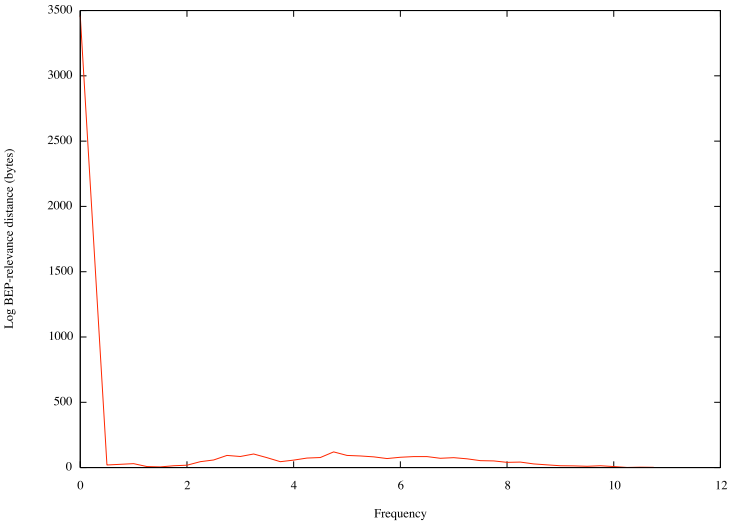


Fig. 2. Distance between BEP and first highlighted character

Table 17. Mixture model results for the Thorough Task (generalized, off)

Run	MAep	nxCG@5	nxCG@10	nxCG@25	nxCG@50
element $\lambda_e = 0.05, \lambda_d = 0.10$	0.0433	0.3429	0.3176	0.2776	0.2324
element $\lambda_e = 0.10, \lambda_d = 0.05$	0.0447	0.3640	0.3358	0.2918	0.2457
element $\lambda_e = 0.10, \lambda_d = 0.10$	0.0447	0.3693	0.3361	0.2939	0.2464
element $\lambda_e = 0.20, \lambda_d = 0.10$	0.0452	0.3895	0.3477	0.2976	0.2506
element $\lambda_e = 0.30, \lambda_d = 0.10$	0.0452	0.3900	0.3481	0.3041	0.2513

in the article has very little impact, leading to scores of 1 for a Best-Entry-Point anywhere in the document. Table 15 shows 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. As mentioned above, with higher A values the position of the BEP has less effect, which can be seen by the convergence of scores for the different methods. Our Best in Context runs where based on postprocessing the All in Context run, resulting 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.

In Table 16, we also evaluate runs here using the same measures as the Thorough Task above, so optimal performance will result in still grossly imperfect scores. 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.

This result can be explained if we look at the position of the Best-Entry-Point relative to the first highlighted character in relevant articles. Figure 2 shows the distribution of absolute distance in bytes between the Best-Entry-Point and the position of the first highlighted character (i.e., the position at which the first passage starts) over articles with relevance. Clearly, assessors predominantly judge the first highlighted character to be the best point to start reading.

9 Mixture Model Runs

In this section we look at the performance of the multinomial language model that takes the context of an element into account. We experimented with various λ_e and λ_d values of the mixture language model and compare performance with the runs mentioned above.

Table 17 shows that less smoothing on the element model improves performance, not only for the official MAep measure, but also for precision at all levels. The difference between $\lambda_e = 0.20$ and $\lambda_e = 0.30$ is very small. If we compare the results with runs based on the element index and the section index (see Table 8), performance of the mixture model is comparable to that of the section index run. Also, setting the article model smoothing parameter λ_d higher also slightly improves precision scores. In sum, the mixture language model has a slight positive effect on precision, but does not improve overall performance.

Table 18 shows the results for the Focused Task. Here, the effects of the two smoothing parameters are very similar to the Thorough Task. When compared to the official element and section based runs (Table 10), we see no gain in performance.

Table 19 shows the results for the All In Context Task, where we restrict our attention to the article ordering based on the highest scoring element. The results show the same effect of the smoothing parameters as for the other tasks. A higher λ_e leads to better performance. The difference between $\lambda_e = 0.20$ and $\lambda_e = 0.30$ is very small again. The mixture model can lead to improvement of precision and MAgP, however the gain is inferior to the increase of length normalization (see Table 14).

The mixture language model proved far less effective for the Wikipedia collection, than earlier research on the IEEE collection [4]. An obvious source of explanation is in the relative length of Wikipedia articles: recall that the average

Table 18. Mixture model results for the Focused Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
element $\lambda_e = 0.05, \lambda_d = 0.10$	0.2987	0.2662	0.1997	0.1542	0.0116
element $\lambda_e = 0.10, \lambda_d = 0.05$	0.3219	0.2786	0.2134	0.1639	0.0125
element $\lambda_e = 0.10, \lambda_d = 0.10$	0.3220	0.2817	0.2136	0.1639	0.0124
element $\lambda_e = 0.20, \lambda_d = 0.10$	0.3334	0.2847	0.2174	0.1688	0.0127
element $\lambda_e = 0.30, \lambda_d = 0.10$	0.3335	0.2833	0.2214	0.1696	0.0130

Table 19. Mixture model results for the All In Context Task (generalized, off, highest scoring element)

Run	MAgP	gP@5	gP@10	gP@25	gP@50
element $\lambda_e = 0.05, \lambda_d = 0.10$	0.1403	0.2603	0.2254	0.1719	0.1319
element $\lambda_e = 0.10, \lambda_d = 0.05$	0.1486	0.2786	0.2389	0.1840	0.1397
element $\lambda_e = 0.10, \lambda_d = 0.10$	0.1489	0.2804	0.2416	0.1847	0.1405
element $\lambda_e = 0.20, \lambda_d = 0.10$	0.1530	0.2957	0.2517	0.1924	0.1472
element $\lambda_e = 0.30, \lambda_d = 0.10$	0.1538	0.2969	0.2540	0.1943	0.1480

article contains only 415 words. This makes the article context a less powerful indicator of relevance, and perhaps the model should be extended to incorporate the broader of the particular wikipedia page (for example by considering the rich, semantic link structure).

10 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, and conducted a range of experiments with filtering and clustering XML element retrieval results. Our basic retrieval engine retrieves arbitrary elements from the collection (corresponding to the Thorough Task). These runs are filtered to remove textual overlap between elements (corresponding to the Focused Task). The resulting runs can be clustered per article (corresponding to the All in Context Task). Finally, we select the "best" element for each article (corresponding to the Best in Context Task).

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 surprisingly—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, 64–69 (2006)
2. Wikipedia: The free encyclopedia (2006) http://en.wikipedia.org/
3. Sigurbjörnsson, B., Kamps, J., de Rijke, M.: An Element-Based Approch to XML Retrieval. In: INEX, Workshop Proceedings, pp. 19–26 (2003)
4. Sigurbjörnsson, B., Kamps, J., de Rijke, M.: Mixture models, overlap, and structural hints in XML element retreival. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 196–210. Springer, Heidelberg (2005)
5. Sigurbjörnsson, B., Kamps, J.: The effect of structured queries and selective indexing on XML retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 104–118. Springer, Heidelberg (2006)
6. Hiemstra, D.: Using Language Models for Information Retrieval. PhD thesis, University of Twente (2001)
7. Sigurbjörnsson, B.: Focused Information Access using XML Element Retrieval. SIKS dissertation series 2006-28, University of Amsterdam (2006)
8. Kamps, J., de Rijke, M., Sigurbjörnsson, B.: The importance of length normalization for XML retrieval. Information Retrieval 8, 631–654 (2005)

GPX - Gardens Point XML IR at INEX 2006

Shlomo Geva

Faculty of Information Technology
Queensland University of Technology
Queensland 4001 Australia
s.geva@qut.edu.au

Abstract. The INEX 2006 evaluation was based on the Wikipedia collection in XML format. It consisted of several tasks that required different approaches to element selection. In this paper we describe the approach that we adopted in an attempt to satisfy the requirements of all the tasks, Thorough, Focused, Relevant in Context, and Best in Context. We have used the same underlying system to approach all tasks. The retrieval strategy is based on the construction of a collection sub-tree, consisting of all nodes that contain one or more of the search terms. Nodes containing search terms were then assigned a score using the GPX ranking scheme which incorporates TF-IDF or BM25 variants, but extends them. Scores are recursively propagated to ancestors in the document XML tree, and finally all scoring XML elements are ranked. We present results that demonstrate that the approach is versatile and produces consistently good performance. We also provide empirical analysis of the GPX ranking scheme and compare its performance against a baseline TF-IDF and a BM25 scoring scheme..

Keywords: XML IR Information Retrieval GPX INEX Evaluation.

1 Introduction

The INEX 2006 Ad-hoc track consisted of 4 tasks, namely Thorough, Focused, All in Context and Best in Context retrieval. These tasks are described elsewhere in the proceedings. We have used the 2005 GPX search engine algorithms with some minor modifications [1,3]. The software was ported in 2006 from C# and MS-Access to Java and the Apache Derby relational database. This was done to achieve speedup in searching, but the basic system remained almost unchanged. We implemented extended support for more complex queries [2], lifting some of the limitations of NEXI. GPX thus represents an evolving system that started in 2004 and its evolution documented in the annual workshop proceedings. The reader is invited to read the 2004 and 2005 descriptions of GPX for more detail [1,3]. In this paper we describe the approaches we took to the various tasks in 2006 and discuss the results. Finally, we analyze the performance of the GPX scoring scheme by comparing it against TF-IDF and BM25 [4]. In the following sections we have also attempted to provide a comprehensive enough description of GPX so that it can be reproduced on the back of any XML index system that supports the retrieval of XPath inverted lists.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 137–150, 2007.

2 The GPX Search Engine

For the sake of completeness we provide a brief description of GPX. 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 identifying a specific XML element, and the term position within the element. The actual data structure is designed for efficient storage and retrieval of the inverted lists which are considerably less concise by comparison with basic text retrieval inverted lists.

2.1 Inverted List Representation

We have chosen to implement GPX using a relational database as backbone architecture; however, the system is in fact based on a traditional inverted list which was extended to support XML IR. The choice is motivated by the extensive off the shelf functionality of a DBMS and ease of programming of all I/O operations. We do not however use any of the expensive recovery and concurrency mechanisms that the DBMS supports by using minimal footprint embedded mode executables. The software that we used is the Apache Derby[1] open source DBMS, freely available under Apache License, Version 2.0. Derby's footprint is small -- about 2 megabytes for the base engine and embedded JDBC driver.

In principle, before optimising the database schema, a suitable inverted list for our purposes consists of a single table with the following structure:

Term-Context = { **Term**, File-Name, XPath, Position }

This structure is sufficient to allow us, given a term, to retrieve all contexts in which the term appears. The Position column allows us to support phrase searches or proximity operators. The collection contains approximately 140 million postings hence each byte in a posting contributes 140MB to the size of the inverted list (ignoring other overheads). It is obvious that this structure is exceedingly redundant and the representation of postings in the list can become very expensive. We describe several ways by which we kept the inverted lists index size under control.

There are approximately 2 million unique terms in the collection. It is not necessary to store the Term column in the table because it could be stored in an auxiliary table, recording the Term, Start position, and End position in the inverted list table. However, this makes it more difficult to manage a dynamic collection where insertions and deletions are allowed. We have chosen not to adopt this approach. However, we do map a term to Term-ID. The Term-ID is only a 4 byte integer and an auxiliary table provides the mapping of Term to Term-ID. Terms usually exceed 4 bytes, particularly since there is always an overhead of several bytes if variable length strings are used. Similarly, file names are very long and so a simple normalization is performed to map a File-Name to a File-ID. With about 660,000 files in the collection 4 bytes are sufficient. The Position column can be safely stored on 2 bytes since text nodes do not exceed 64KB.

[1] http://db.apache.org/derby/

The representation of the XPath is more problematic. The tag names can be rather long, and XPath expressions can contain numerous nodes and be very long. We could have encoded tag names - there are less than 256 meaningful tags so one byte could suffice to represent a tag. We have chosen not to do so because that would render the lists unreadable without decoding and this was inconvenient. With XPath lengths varying from 10 bytes to over 300 in this collection, the overhead of storing the explicit XPath is significant even with coding. We observe that each unique XPath is repeated in the inverted list for each term in the same node, and the XPaths themselves are repeated in many files. For instance, almost every article in the collection has a node /article[1]/body[1]/p[1]. There is significant redundancy here already, but furthermore, many paths which are not identical share a common sub-path. This suggests a compression scheme like LZW might be effective. We considered this to be unnecessary, particularly given the processing overheads and so we have adopted the following simple yet effective compression scheme.

Consider the XPath:

/article[1]/bdy[1]/sec[5]/p[3]

This could be represented by two expressions, a Tag-set and an Index-set:

Tag-set: article/bdy/sec/p

Index-Set: 1/1/5/3

The original XPath can be reconstructed from the tag-set and the index-set. It turns out that there are over 48,000 unique tag-sets, and about 500,000 unique index-sets in the collection. We assign to each tag set and each index-set a hash code and create auxiliary database tables mapping the hash-codes to the corresponding tag-set and index-set entries. These hash tables are small enough to be held in memory and so run-time decoding is efficient. We have used Java's inbuilt hash code function. It is only a 32 bit code, but given the number of elements the risk of hash collisions is minimal and we take it.

Finally, in order to implement BM25 it is necessary to record node sizes. So an XPath table is maintained where the XPath is represented by a hash code computed from the concatenation of the FileName and the XPath columns. We use a 63 bit MD5 hash code because there are about 25 million distinct nodes in the collection and the probability of a collision is too high with Java's 32 bit inbuilt hash function. We use 63 bits although MD5 provides a 64bit code since Java's *Long* integers are limited to the constant 0x7FFFFFFFFFFFFFFF.

Finally, the database schema consists of the following tables:

Term-Context ={ **Term-ID**, File-ID, XPath-Tag-ID, XPath-IDX-ID, Position }
Terms = { **Term**, Term-ID }
Files = { **File-Name**, File-ID }
TagSet = { **XPath-Tag-ID**, Tag-Set }
IndexSet = { **XPath-IDX-ID**, Index-Set }
XPathSize = { **XPath-ID**, Node-Size }

The size of the database is 15GB and this represents an overhead of about 3:1 over the source documents (uncompressed). This is quite acceptable with current disk costs and capacities even if much more efficient representations are possible.

Some performance figures of this database are as follows. The time it takes to parse and load the 659,388 files on a 3GHz PC with 2 GB RAM is 9 hours. When the search engine is started, the Files, TagSet, IndexSet, and XPathSize tables are loaded into RAM – this takes about 15 seconds. The only required I/O operations during a search are then on the Terms table and the Term-Context inverted list table. The average time to evaluate a topic is 7.2 seconds, but a few topics take more than 30 seconds to evaluate on account of having more terms and longer inverted lists that correspond to common terms.

2.2 The GPX Ranking Scheme

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. Several steps are followed in evaluating a query and these are described in the following sub-sections.

2.2.1 Calculation of Text Nodes Score

Equation 1: Calculation of element relevance score from its content

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

Here n is the count of unique query terms contained within the element, and 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. This heuristic of rewarding the appearance of multiple distinct terms can conversely be viewed as taking more strongly into account the absence of query terms in a document. Here it is done by rewarding elements that do contain more distinct query terms. The system is not sensitive to the value of K as demonstrated in the results section and a value of $k=5$ is adequate. The summation is performed over all n terms that are found within the element where t_i is the frequency of the ith query term in the element and f_i is the frequency of the ith query term in the collection. Similar results are obtained if we use the *TF-IDF* to compute the sum, but it does not lead to significantly different results in our experience. We describe the results of experiments with *TF-IDF* and with *BM25* in a later section. At INEX 2006 we used the term inverse collection frequency and refer to it as *TF-ICF* (as distinct from other *TF-IDF* variants.)

Finally, phrases are weighted more heavily than individual terms (phrase weight are multiplied by 10) and nodes that contain query terms that are preceded by a minus sign (undesirable) are not returned at all.

2.2.2 The GPX NEXI Interpretation

The GPX search engine supports an extended set of functionalities which are a superset of NEXI. These are described in more detail by Geva et al in [3]. Here we limit the discussion to the details which are relevant to the INEX 2006 evaluation.

The evaluation of a NEXI expression always starts by converting the query expression from postfix to infix for sequential evaluation. For example, consider the query –

```
//article[about(.,Albert Einstein)]/body[about(.//figure,Copenhagen)  OR
  about(.//section,Bohr)]
```

The query is converted to the following stack oriented evaluation specification –

1) PUSH(//article[about(.,Albert Einstein)])
2) PUSH(//article/body[about(.//figure,Copenhagen)])
3) PUSH(//article/body[about(.//section,Bohr)]
4) PUSH(OR(POP, POP))
5) PUSH(SUPPORT(POP,POP))

This set of operations is evaluated by using an inverted lists stack. The first 3 steps evaluate the 3 distinct filters that appear in the NEXI expression. Each list of elements satisfies its respective *about* clause. In step 4 the top two lists are ORed and the resulting list pushed back onto the stack. This effectively evaluates the OR operator on the body element. In the final step the SUPPORT operator is applied to take into account the filter on the article node which "supports" the selection of the body node on account of content which is not necessarily contained in the body. The implementation of OR, AND, and SUPPORT is now explained.

The OR operator computes the union of two inverted lists, X and Y. The call to OR(X,Y) returns a new list. Elements in the lists identify XML result elements by file-id, full XPath expression, and relevance score. The OR operator performs a set union whereby elements that appear in both lists are merged and their scores added together. Other elements that appear in either list keep their original scores.

The AND operator computes the intersection of two inverted lists, X and Y. In GPX this operator is not necessarily a strict set intersection but can be loosely interpreted in one of three ways. The default option is to simply implement it as OR(X,Y). However, in some queries the user really means AND in a strict sense; therefore, a second option is to implement it as a strict set intersection - only XML elements that appear in both X and Y are kept, and their scores are added together. This option is too restrictive because sometimes the lists contain overlapping elements and then the relationship with respect to AND is unclear. By insisting on a strict match many relevant results are lost. The third implementation keeps overlapping nodes, combines the scores, but keeps only the largest node (oldest common ancestor). In the experiments that we report in the next section, we used the first (default) option. This seems to work quite well in most instances, and works better on average.

The SUPPORT operator does not have an equivalent set operator and is specific to our interpretation of NEXI. In NEXI, we refer to support elements and target elements. Target elements are those elements that appear at the tail of the NEXI expression with a filter, while support elements are internal elements with filters that appear along the path to the target elements. The SUPPORT operator takes a list of nodes in X that provide support to the selection of nodes from list Y. For instance, when we look for paragraphs about Americium in articles with abstracts about the

Periodic Table, the target elements are paragraphs about Americium, and paragraphs are supported by abstracts about the Periodic Table. Both the support and target elements must have a common ancestor within the document tree. In the case of the Wikipedia this is the article element. The supporting abstract must appear in the same article as the supported paragraphs. The support operator identifies for each result element in Y, all the support elements in X, and combines the scores. It is important to note that all the elements in Y are returned, regardless of support. However, elements with support have an increased score.

2.2.3 The GPX Derivation of a Full Recall Base

Having computed the scores of all elements in the collection which contain query terms directly as text within the XML node, we must proceed to consider the scores of elements on account of their relevant descendents. The scores of retrieved elements are now recursively propagated upwards in the document XML tree according to the following scheme.

Equation 2: Calculation of a Branch Element Relevance Score

$$R = L_{0+} D(n) \sum_{i=1}^{n} L_i \tag{2}$$

Where:

L_0 = the score of the current node (from Equation 1), zero by default
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 introduction of the term L_0 was necessary when moving from the IEEE articles collection to the Wikipedia since text appeared only in leaf nodes in the IEEE collection, but many Wikipedia nodes contained both direct text and descendents with text. 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. Generally we have 0<=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 its descendent paragraphs.

It is attractive to consider the use of node size directly in score propagation. As we progress upwards through the tree, node specificity tends to decrease, but coverage (recall) can increase when multiple descendents are combined in an ancestor node. Node size can provide some direct information in relation to precision, but we were unable to discover a robust way to incorporate the node size into Equation 2.

Finally, the cost of propagating the scores can be very high. Many of the terms that appear in the topics are rather common. This leads to a very high computational load. In order to reduce the total time it took to generate the results we have imposed two limitations. All terms that occur with a frequency greater than 100,000 in the

collection were treated as stop-words and ignored. This proved to be of little consequence with the topic set on hand. Of course one can imagine situations when this would be a costly decision. This reduced the processing time of the entire set of topics by 50% with very minimal degradation in MAep values (less than 1%). The second limitation that we imposed was more severe. It reduced the processing time by more than 7 fold. The limitation was placed on the number of nodes that were taken forward from the first phase (Equation 1) towards generation of the full recall base. We have taken the 3000 highest scoring nodes at most. In some instances hundreds of thousands of scoring elements were dropped. This meant that incomplete information was used in propagating scores upwards in the document tree. As it turns out this did make a significant difference to precision and recall. This performance cost is discussed later in the experimental section. It must be noted that both limitations can be lifted at the cost of increased processing time which can be reduced by other means without sacrificing precision and recall – for instance, by storing inverted lists for word-grams instead of single terms. With extremely common terms this could reduce the list lengths by several orders of magnitude by reducing the time required to complete the I/O and set operations over the lists.

After the scores of all nodes are computed GPX proceeds to add the score of the Article node in each document to the score of each node in the article (including the article node itself). This heuristic correction is intended to generally push up the scores of elements that appear in documents that are more relevant. Empirically we were able to establish that better results may be obtained in this manner.

Finally, we note that GPX is based on a simple variation of TFIDF. Robertson provides a comprehensive discussion of various theoretical arguments for IDF [5], but in the end IDF is still a heuristic approach, and so is GPX.

3 Experimental Results

In this section we present and discuss the results that were obtained at INEX 2006. We also present the results of an empirical sensitivity analysis of various parameters of the GPX search engine, performed with the Wikipedia collection.

3.1 Thorough Retrieval

All the QUT runs were generated with GPX search engine, starting with thorough retrieval. We have experimented with several settings of the decay factor, with strict and loose interpretation of NEXI expressions, and with various query expansion techniques. The official results of the Thorough Retrieval task are reproduced in Figure 1. The solid line is the GPX submission that was ranked 3rd, with a MAep value of 0.0699. This is just 0.001 below the top submission, but qualitatively there seems to be a difference - the precision of the GPX run is slightly lower at low recall levels, but the overall recall (area under the curve) is higher. This run was obtained with N1=0.11 and N2=0.31 in determining D(n) in Equation 2. Both GPX runs were CO runs. The COS runs did not perform as well. This is the reverse of what was observed with the IEEE CS collection that was used in earlier evaluations. We attribute the difference to the apparent lack of semantic tagging in the Wikipedia.

The solid dash-dot line corresponds to the GPX run which was ranked 9th with MAep value of 0.0620 and was obtained with the values of the decay parameter N1=0.31 and N2=0.71. Qualitatively it is similar to the other runs in the top 10.

The solid dotted line is an unofficial run that corresponds to exactly the same system setting as our best official run, except that we kept 30,000 elements rather than 3,000 in producing the full recall base through score propagation (equation 2). The MAep increased by 10% to 0.077. This is a very significant improvement and it quantifies the cost of more efficient retrieval. Instead of 15 minutes, it took 111 minutes to generate a complete run submission of 125 topics.

Another difference between the two GPX runs is that the better performing run (by MAep) was produced by adding to the CO title element a support filter. The filter was placed over the article name. In this manner, elements that appeared in articles whose name was *about* the same keywords as the title received a boosted score. This was simply achieved by adding the filter to the title. For example,

//article[about(.,X Y Z)] → //article[about(./name,(X Y Z)]//*[about(.,X Y Z)]

It should be noted that in GPX the meaning of the "//*" path specification is "this node or any descendents" rather than "any descendent". The modified expression is evaluated by GPX in the usual manner and supported nodes receive an additional score from the support element – if found in the same article. The heuristic is obvious – if the search terms appear in the Wikipedia article name itself then it is more probable that the article is relevant.

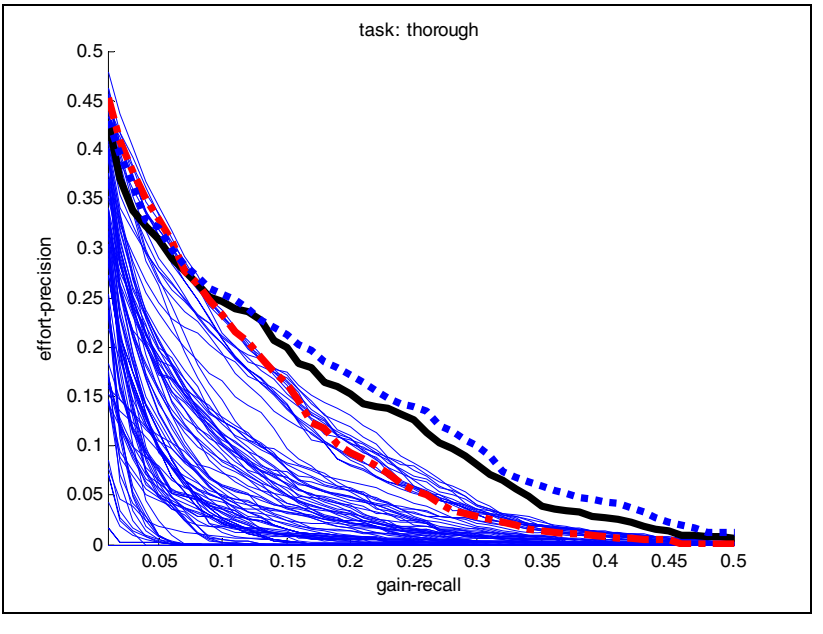


Fig. 1. GPX Thorough Retrieval

3.2 Focused Retrieval

Focused Retrieval starts with the thorough results recall base. The highest scoring elements on a path are selected by keeping only elements that have a higher score than any of their descendents or ancestors. Figures 2 and 3 depict the official plots, with the solid heavy line corresponding to the best official GPX submission. Again, the unofficial dotted line was later produced by running GPX with the same setting as the best official run, but taking the top 30,000 elements rather than the top 3,000 in generating the thorough recall base from which the focused run was produced. The performance difference is again quite large. There is a clear incentive to write a more efficient implementation of GPX that will keep processing low while providing a significant performance improvement.

3.3 Best in Context

We tested a trivial approach here – we simply kept the highest scoring element in each document appearing in the recall base. This simple approach seems to have produced good results with the BEPD metric. The GPX submission was ranked 3rd with the setting A=0.01. This, according to the official BEP metric documentation, means that the system was comparatively successful in pinpointing the BEP. Low A values favor runs that return elements that are very close to the BEP.

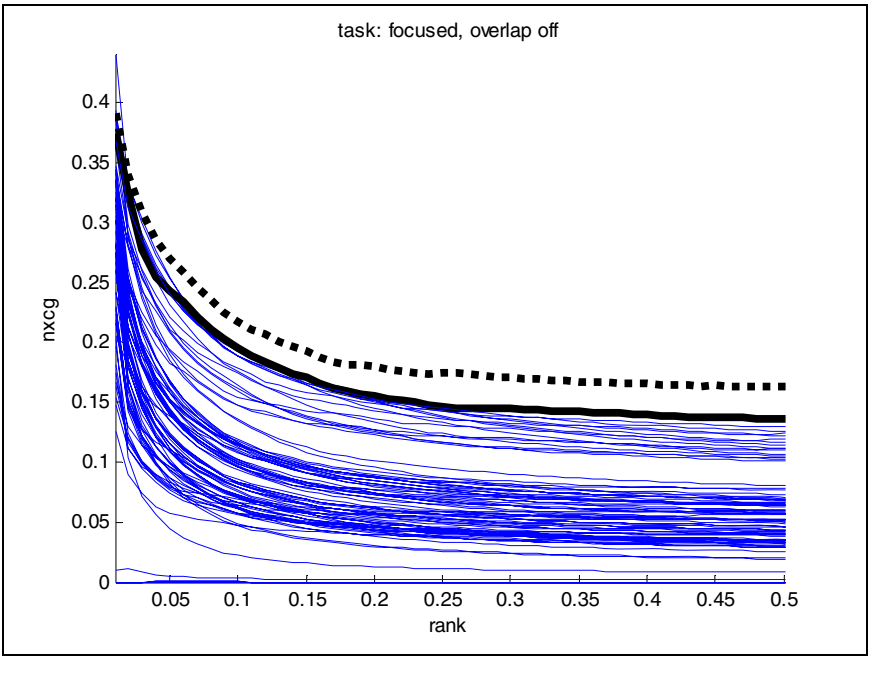


Fig. 2. GPX Focused Retrieval, with overlap OFF

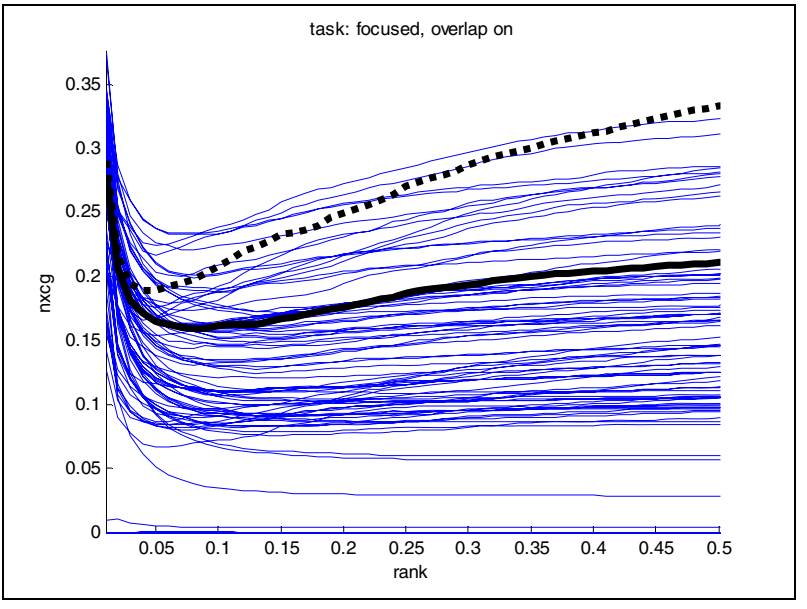


Fig. 3. GPX Focused Retrieval, with overlap ON

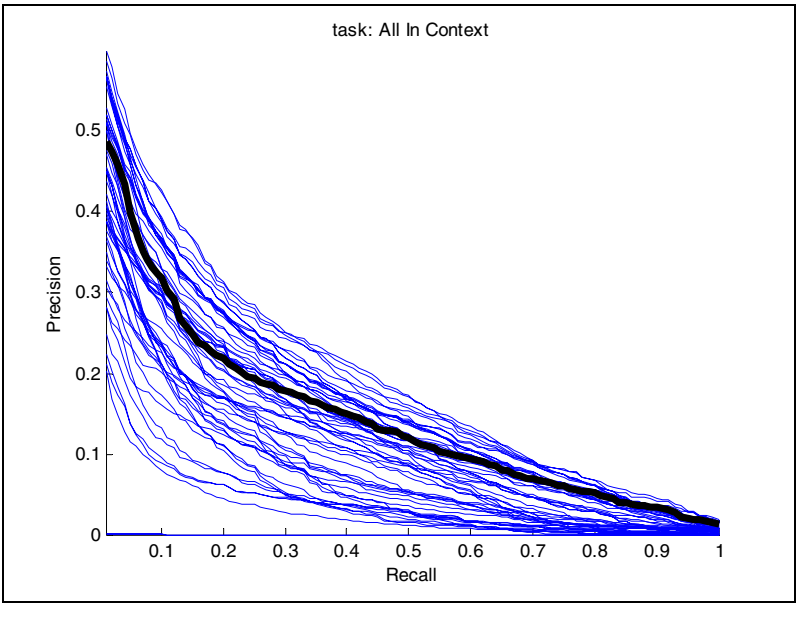


Fig. 4. GPX All in Context Retrieval

3.4 All in Context

The objective of the task was to balance article retrieval and element retrieval. Whole articles are first ranked in descending order of relevance and within each article a set of non-overlapping most focused elements are grouped. We have used the focused results, which were overlap free already, but grouped the elements within articles and sorted the articles by score. The results are reasonable but performance suffers because the focused recall base is not ideal for this task. The focused recall base retains the most relevant elements in the collection, but *out of context*. This means that high recall within article is not assured. Nodes that have a lower score can drop out of the top 1500 elements of the Focused run. This works against All in Context retrieval where the F-Score of an article demands high recall within each article. In addition to this, since from the outset we have only considered the top 3000 elements in generating the thorough recall base, even the thorough recall base was incomplete. This means that elements that might otherwise appear in highly ranked articles were never retained. Figure 4 depicts the official results with the GPX best run depicted as a heavy solid line.

3.5 Empirical Evaluation of GPX Scoring

In this section we present the results of empirical evaluation of the GPX scoring strategy. We study the effect of each of the components in Equation 1 and Equation 2. In order to evaluate the sensitivity of the evaluation to the score propagation constant D(n) in Equation 2, we have fixed all other parameters, and used N1=N2=N, and varied the value of N in small steps from zero to one. The results are summarized in Table 1.

Table 1. The impact of choice D(n)

D(n)	Thorough MAep	Focussed nxCG@50 overlap OFF	Focussed nxCG@50 overlap ON	All in Context MAep
0	0.017	0.262	0.219	0.082
0.1	0.039	0.280	0.231	0.117
0.2	0.049	0.289	0.232	0.125
0.3	0.056	0.296	0.237	0.131
0.4	0.062	0.298	0.233	0.135
0.5	0.066	0.287	0.234	0.142
0.6	0.068	0.268	0.235	0.148
0.7	0.070	0.248	0.230	0.153
0.8	0.069	0.235	0.223	0.156
0.9	0.069	0.218	0.211	0.157
1.0	0.068	0.154	0.158	0.100

The best performance is achieved at different D(n) values for the different tasks. In the Thorough and All in Context higher values (0.8) produce better results since higher D(n) values lead to exhaustive selection within article. In the Focused task lower values (0.3) produce better results since precision is favoured by lower D(n) values. This is also confirmed in our official run results. Importantly however, the performance is not extremely sensitive to the selected value and there is no catastrophic degradation of performance for some values.

In order to ascertain that the heuristic motivation to Equation 1 is indeed sound, we have conducted experiments where we evaluated the individual components in isolation. Figures 5 and 6 depict the performance of runs when the score calculation is based on the number of unique query terms alone (K^{n-1}), on a TFIDF variation alone (TFICF or BM25), and on the combination of the two as in Equation 1. It is clear that when both components are used the performance is much improved (dotted line). Furthermore, there is no advantage to using BM25 over the simpler TFICF variation. This is not surprising since most text elements are small and BM25 is indeed expected to make little difference for small documents [4]. We have used BM25 with the default values K1=2 and b=0.75, but other values produced similar results.

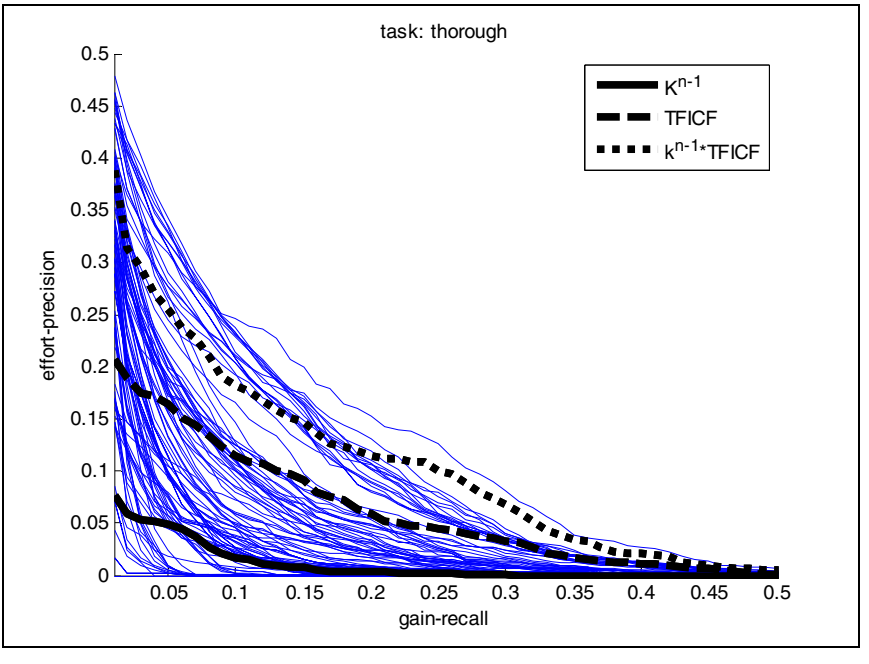


Fig. 5. GPX with TFICF

Finally, we tested several variations for the value of K in Equation 2. While holding all other parameters constant we have varied the value of K from 1 to 50. Figure 7 depicts the results. There is an improvement as K values increase to about 5 and then for values of between 5 and 50 there is no further improvement and the

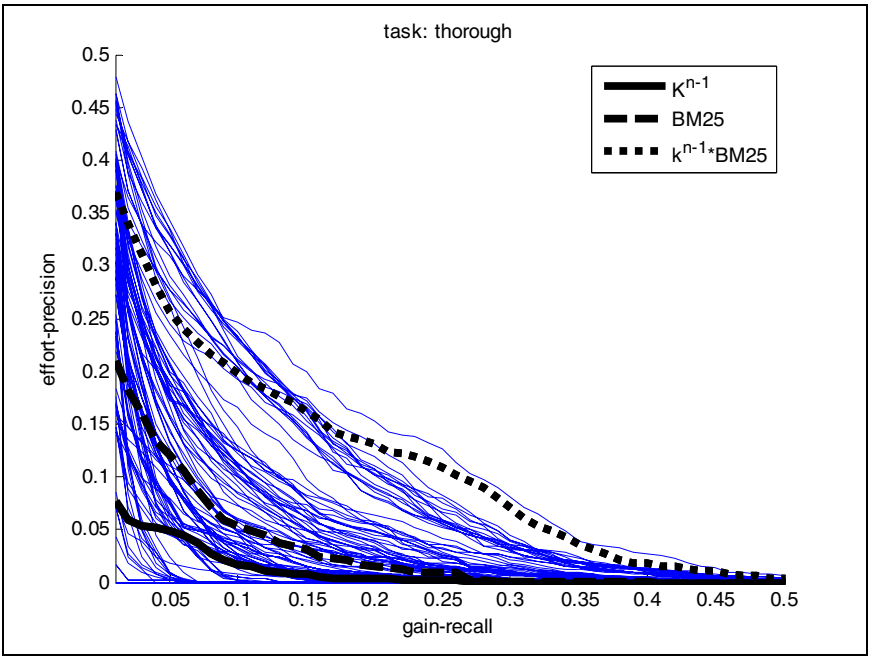


Fig. 6. GPX with BM25

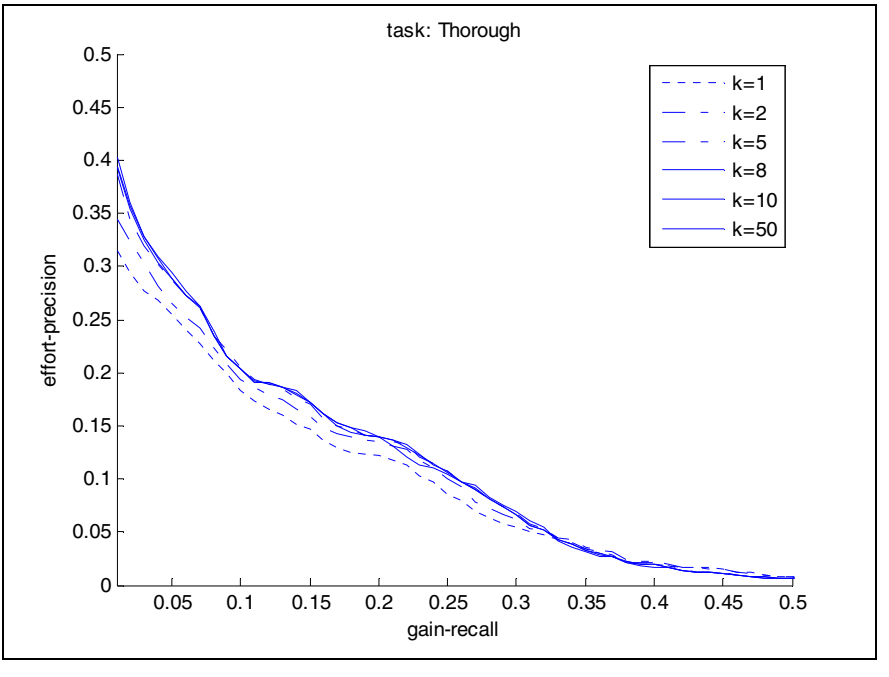


Fig. 7. GPX with varying K values

performance is stabilized. This can be understood as follows. Equation 1 heavily rewards elements that contain more distinct terms, through K^{n-1}. The TFIDF component in Equation 1 moderates that score. Once the value of K is large enough the moderation that is contributed by the TFIDF component is no longer sufficient to moderate the rank order of elements with a very different number of distinct query terms.

4 Conclusions

GPX performed rather well on the Wikipedia in most tasks. This result demonstrates that the method is quite robust since GPX was designed and implemented with the IEEE collection, but evaluated with the Wikipedia. The best relative performance was achieved in the Thorough, Focused (overlap off) , and BEP tasks. The performance in the All in Context task and Focused (overlap on) was not quite as good, but respectable nevertheless. Future work will focus on ranking strategies that take node size and structure into account in an explicit manner, to try and capture the intuitively appealing F-Score calculation which was used in the evaluation of the All in Context task. More work is also required on improving search efficiency. List processing is extensive and the current implementation is CPU bound rather than I/O bound. A response time of 7 seconds per topic is inadequate for implementing a high throughput online system. We are unable to compare the efficiency of our system with that of other systems at this stage because the INEX evaluation does not formally support a systematic comparison of this aspect.

References

1. Geva, S.: GPX - Gardens Point XML Information Retrieval INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S., Szlavik, Z. (eds.) Advances in XML Information Retrieval. Third International Workshop of the Initiative for the Evaluation of XML. LNCS, pp. 211–223. Springer, Heidelberg (2005)
2. Geva, S.: GPX - Gardens Point XML IR at INEX 2005, INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) Advances in XML Information Retrieval. Fourth International Workshop of the Initiative for the Evaluation of XML. LNCS, pp. 240–253. Springer, Heidelberg (2006)
3. Geva, S., Tannier, X., Hassler, M.: XOR - XML Oriented Retrieval Language, SIGIR 2006, Workshop on XML Element Retrieval Methodology, Proceedings online at: http:/www.cs.otago.ac.nz/sigirmw/Proceedings.pdf
4. Robertson, S.E., Sparck Jones, K.: Simple, proven approaches to text retrieval, University of Cambridge Technical Report UCAM-CL-TR-356, ISSN 1476-2986, December 1994, last updated February 2006. http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-356.pdf
5. Robertson, S.: Understanding Inverse Document Frequency: On theoretical arguments for IDF. Journal of Documentation 60(5), 503–520 (2004)

IBM HRL at INEX 06

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 nested XML elements to different indices. This worked fine for the IEEE collection which has a small number of potential component types that can be returned as query results. 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 some preliminary work we did to exploit the connectivity of the Wikipedia collection to improve ranking.

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 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 XML retrieval is how to adapt ranking methods from classical IR that retrieve and 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 [6, 7]

In previous INEX workshops we described a component ranking algorithm [6, 7] 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, since 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 have any assumptions on a pre defined set of potential component types, but still uses the same idea of a small number of separate indices with no nested

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 151–159, 2007.

elements. We further show our attempts to exploit the rich connectivity of the Wikipedia collection to improve component ranking by looking at <collectionlink> and at their anchor text.

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 <collectionlink> and anchor eext to the ranking algorithm. 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 Algorithm

The basic idea in the Component ranking algorithm [6, 7] 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 [6, 7]. 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[4] or similar (step 2) on each result set. Then in step 3, the scores of elements in each result set are normalized to 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.

Search(Q):

For each index i

1. Compute the result set R_i of running Q on index i
2. [Optionally] Apply AQR algorithm on R_i
3. Normalize scores in R_i to [0,1] by normalizing to score(Q,Q)
4. [Optionally] Scale each score by its containing article score from R_0

Merge all R_i's to a single result set R composed of all components sorted by their score

Fig. 1. Component ranking algorithm

The above algorithm requires some prior knowledge on the collection such as the element types to index in each of the 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 as query results so the method of selecting a small number (e.g. 7) of the most informative elements does not scale for such collections.

Table 1. Distribution by element type size

	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

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 an 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 $Index_0 \ldots Index_{n-1}$ where n = numIndices. For each such minimal element we extract all its ancestors and if its depth is less than the number of indices then each ancestor is indexed in a separate index. For example assume 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 no element whose direct size greater than minCompSize then we only index the document itself (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 each index are not nested. This is because for each document, the elements indexed in each $Index_i$ ($i < mumIndices$) are of the same length, therefore can not be 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 Exploiting <collectionlink> and Anchor Text

The Wikipedia collection is highly connected through internal <collectionlink> links. For example document 10013.xml contains the link:

```
<collectionlink xlink:type="simple" xlink:href="18957.xml">
          medical
</collectionlink>
```

which creates a link from document 10013.xml to document 18957.xml. Each such link is further accompanied by a child text that is actually the way that the pointing document describes the pointed document. We call this text the "anchor text" of the link. In the above example the anchor text is "medical".

It would be interesting to try a PageRank [1] like algorithm to see if it can improve recall/precision. The PageRank algorithm assigns a value called PageRank to a document as a function of the number of incoming links to that document and as a function of the PageRank of the pointing documents. A document with a higher PageRank is deemed to be more relevant for a topic than a similar document with a lower PageRank.

In our submissions we tried a much simpler approach. We just count for each document the number of incoming links and use this as the document's weight. We refer to this weight as the document's static score. We further add to each document, the anchor text of all incoming links, but we mark those terms with a special flag so that later in the ranking algorithm we can assign different weight to them.

Since we do XML retrieval it could be beneficial to assign an individual static score and anchor text to each XML element. However, since the links in Wikipedia point to full documents and not to internal XML elements, our indexing schema infers for each document, the document's static score and anchor text to all its contained elements.

A modified component ranking algorithm that takes into account the static score is depicted in Figure 2 below –

Search(Q, SSWeight)

For each index i

1. Compute the result set R_i of running Q on index i
2. [Optionally] Apply AQR algorithm on R_i
3. Normalize scores in R_i to [0,1] by normalizing to score(Q,Q)
4. [Optionally] Weight scores by their static score
5. [Optionally] Scale each score by its containing document's score from R_0

Merge all R_i's to a single result set R composed of all components sorted by their score

Fig. 2. Component ranking algorithm with static score

The algorithm gets a new parameter SSWeight (static score weight) which is the weight we give to the static score in the final element's score. The new step 4 assigns scores to elements according to the following formula –

$$Score(e) = SSWeight * e_{staticscore} + (1 - SSWeight) * Score(e)$$

Where $e_{staticscore}$ is the element's static score.

4 Runs and Results

We describe our submissions and results for the CO and COS thorough and BestIn-Context runs. The runs were submitted using the Juru[3] search engine. In all runs we applied term stemming and we ignored phrase boundaries (namely phrase terms were treated as simple terms). We treat "+" terms as regular terms and use the "+" only to boost score of those terms. We treat "-" terms strictly, namely we never return a result which has a "-" term. For all runs we picked minCompSize=20 and numIndices=7 and we added the anchor text to the pointed documents with same weight as regular terms.

From the 5 steps in the modified component ranking algorithm (see figure 2) we skipped the optional step 2 namely we didn't apply Automatic Query Refinement. In step 5 we used doc pivot 0.5 namely elements scores were composed from 50% of their original score in their index and 50% of their full document score.

Out official runs are described in the following subsections.

Thorough Runs

We submitted 3 CO runs and 3 COS runs. For the COS runs we translated the topics to XML Fragments[2, 5] and applied a variant of the component ranking algorithm (see Figure 2) by using the tags in the first index (the index of the full documents) to filter out elements from the final result set based on the topic structure. The detailed algorithm for COS is described in [6].

The results of our thorough runs using the filtered assessment pool[1] with Metric:ep-gr, Quantization: gen, overlap=off are summarized in Table 2 below.

Table 2. Results for the thorough run

Rank	Run type	Run description	MAep
7	CO	SSWeight=0.1, title + on topic keywords	0.0653
10	CO	SSWeight=0.1, title only	0.0619
11	CO	SSWeight=0, title only	0.0616
32	COS	SSWeight=0.2, with synonyms	0.0323
34	COS	SSWeight=0.1, with synonyms	0.0319
47	COS	SSWeight=0, no synonyms	0.0242

So we tried several static score weights and for the COS runs we tried runs were we treat the structure strictly (row 6 – no synonyms) and runs in which we allowed a more vague structure interpretation (rows 4,5 – with synonyms) by defining {section, p, item} as synonyms to each other.

Our observations from the results are as follows:

1. The best result was achieved for a topic that used both the title and the on topic keywords. However we are more interested in the runs which used the title only (rows 2-6).
2. The 3 CO runs were superior to the COS runs. This strengthens our findings from last years that structural hints do not seem to contribute to the results.
3. Our naïve static score implementation didn't change the results much. We can see that the MAep of the 2nd and 3rd rows is almost identical were one was using SSWeight 0.1 and one didn't use static score at all.

[1] The filtered assessment pool is the pool after removing small *link* elements that were inserted as relevant automatically by the assessment tool.

4. The COS runs with strict structure (row 6) was inferior to COS runs that treat structure vaguely.

BestInContext Runs

The BestInContext run was performed by first running a thorough run and then a filtering step on the returned results. In the filtering step we go over the returned ranked list of elements and pick the first element seen from each document as the best entry point for that document. We then remove all other elements from same document. Our results for the metric BEPD with parameter A=0.01 are summarized in table 3 below –

Table 3. Results for the BestInContext runs, metric BEPD, A=0.01

Rank	Run type	Run description	MAep
4	CO	SSWeight=0, title only	0.1621
5	CO	SSWeight=0.1, title only	0.1614
7	CO	SSWeight=0.2, title only	0.1608
26	COS	SSWeight=0.2, with synonyms	0.1213
29	COS	SSWeight=0.1, with synonyms	0.1177
52	COS	SSWeight=0, no synonyms	0.0754

Like in the thorough run we can see that our naïve static score implementation for <collectionlink> did not yield any change and that our CO runs were much superior to the same COS runs.

The metric parameter A measures how much the returned element is close to the optimal BEP (Best entry point). High values of A (e.g. 10) tend to give a score of 1 to any answer in a relevant document; hence the score does not discriminate whether x is near to or far from the BEP.

For A = 10.0 our runs were ranked top which means that we did rank the documents correctly but the entry point we returned was not always close to the optimal BEP.

5 Discussion and Summary

We described a scalable and 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 applying a naïve static score algorithm based on incoming links 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. This can be used as an indication if our indexing schema is robust enough. We should also explore other indexing algorithm that uses combination of direct size and total size for picking minimal elements. And finally we should try to run a real PageRank[1] algorithm and try different weights for the added anchor text terms.

References

1. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. In: Proceedings of the seventh international conference on World Wide Web 7, pp. 107–117 (Section 2.1.1 Description of PageRank Calculation) (1998)
2. Broder, A.Z., Maarek, Y., Mandelbrod, M., Mass, Y.: Using XML to Query XML–From Theory to Practice. In: Proceedings of RIAO'04, Avignon France (April 2004)
3. Carmel, D., Amitay, E., Herscovici, M., Maarek, Y., Petruschka, Y., Soffer, A.: Juru at TREC 10 - Experiments with Index Pruning. In: Proceedings of NIST TREC 10 (November 2001)
4. 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)
5. 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 (August 2003)
6. Mass, Y., Mandelbrod, M.: Retrieving the most relevant XML Component. In: Proceedings of the Second Workshop of the Initiative for The Evaluation of XML Retrieval (INEX), December 15-17, 2003, Dagstuhl, Germany, pp. 53–58 (2003)
7. Mass, Y., Mandelbrod, M.: Component Ranking and Automatic Query Refinement for XML Retrieval, Advances in XML Information Retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 73–84. Springer, Heidelberg (2005)

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

Mathias Géry

Laboratoire Hubert Curien - UMR CNRS 5516
Université Jean Monnet - Saint-Étienne
Mathias.Géry@univ-st-etienne.fr

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 Information Retrieval, 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 quite homogeneous compared to the Web structure. Document structure is an essential constituent of information description. We have 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.

In this paper, we present an Information Retrieval model that takes into account XML structure view as reading paths. The outline of our paper is the following: Section 2 deals with Web Information Retrieval and structure. Section 3 describes the information propagation that allows to index XML documents as reading paths. Finally, Section 4 presents some preliminary results at INEX 2006.

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, [4]), that is not suitable for the heterogeneous Web. Two

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 160–164, 2007.

other kind of approaches use the "global link information" (i.e. using the whole structure independently of the query) for the indexing phase of an IRS: "information propagation" and "popularity propagation". Another kind of approach, "relevance propagation", uses the "local link information", i.e. using structure considering the query. This kind of propagation considers a subset of the graph, such as the set of nodes relevants to the query. 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 of 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 top to bottom [7]. 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 as 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 links to many good authorities, and a good authority is linked from many good hubs*" [3]).

These link-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 generally disappointing. Several papers in TREC conference (Web track) have showed that there is no significant improvement of IR quality using the links network [8] [2]. 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 the meaning of a link, and how to fully consider it for IR.

We propose a Web model more structured than the uniform "triple-bag", in order to improve IR using links-based techniques. This model aims to describe the information as the authors have thought it, and index the information 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 of 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 of 659.388 XML **articles** (Wikipedia entries). XML documents are composed of sections, sub-sections, paragraphs, etc. In these experiments, we consider only articles a_i and sections s_i.

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

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

INEX Ad-hoc collection test includes 125 queries. Each query is composed of 5 fields : title, castitle, description, narrative, ontopics keywords. In these experiments, we have used only the "title" field.

3.2 VSM Atomic 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, paragraphs, etc., or any other XML element. All the text that appears outside a section is considered only at article level.

Our system is based on the Vector Space Model (VSM) [6] 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:

$\overrightarrow{a_i} = (w_{i1}, w_{i2} ... w_{ij} ... w_{in})$

$\overrightarrow{s_k} = (w_{k1}, w_{k2} ... w_{kj} ... w_{kn})$

n is the number of terms in the collection.

3.3 Information Propagation: Reading Paths for IR

We assume that a reader will generally 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 $s_1, s_2, ...s_{i-1}$ 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. Thus, 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 :
$\boldsymbol{a_i} =, (w_{i1},\ w_{i2}\ ...\ w_{ip}\ ...\ w_{in})$
(b) $\forall\, a_i \in collection, \forall\, s_j \in a_i$, BM25 produces vectors :
$\boldsymbol{s_j} =, (w_{j1},\ w_{j2}\ ...\ w_{jp}\ ...\ w_{jn})$
(c) $\forall a_i \in collection$, for $j = 1$ *to* $sizeof(a_i)$
$\forall s_k, k \in [j+1..sizeof(a_i)]$
if (k - j) <= d_max then
$\boldsymbol{s_k} += \alpha.\frac{\boldsymbol{s_j}}{(k-j)}$
endif

Fig. 1. Information propagation

The function $sizeof(a_i)$ returns the number of sections in a_i. The distance $k - j$ is the distance between two nodes s_k and s_j, i.e. the number of edges between s_k and s_j. The parameters α and d_{max} (maximum distance) have been fixed to 0.16 and 3 in our experiments.

4 Preliminary Results

Wikipedia collection at INEX 2006 is composed of 659.388 XML articles. Our algorithm splits these articles in 1.909.598 first level sections. We use the BM25 weighting function [5], 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 classical indexation (baseline run): without information propagation, i.e. steps a) and b) of our algorithm.
- Sections level 1, BM25 classical indexation with information propagation, i.e. steps a), b) and also step c) of our algorithm.

Our baseline run gives a score of 0.0186, ranked 64th/106. Our second run (simple information propagation), is not conclusive, as the score obtained is below our baseline: 0.0170, ranked 70th/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, tuning the parameters and especially taking into account various document granularities, i.e. various XML elements instead of section level only.

5 Conclusion and Future Work

Our model considers the Web as a set of reading paths, instead of a set of flat, atomic and independent Web pages. Our first experiments at INEX 2006 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 take into account the node order ? To accumulate information along reading path ? To combine reading path indexation and more classical hierarchical indexation ? To index a a reading path, i.e. a document composed of several nodes extracted from distinct parts of a XML document ? How can we extract these "documents" ? How can we identify reading links ?

References

1. Brin, S., Page, L.: The anatomy of a large-scale Hypertextual Web Search Engine. In: 7th World Wide Web Conference (WWW'98), Brisbane, Australia (April 1998)
2. Craswell, N., Hawking, D., Wilkinson, R., Wu, M.: Overview of the trec 2003 web track. In: 12th Text REtrieval Conference, pp. 78–92 (2003)
3. Kleinberg, J.: Authoritative Sources in a Hyperlinked Environnement. Journal of the ACM 46, 604–632 (1999)
4. Mendelzon, A., Mihaila, G.A., Milo, T.: Querying the World Wide Web. Journal of Digital Libraries 1, 68–88 (1997)
5. Robertson, S., Zaragoza, H., Taylor, M.: Simple bm25 extension to multiple weighted fields. In: CIKM 2004, Washington, DC, November 2004, pp. 42–49 (2004)
6. Salton, G., McGill, M.: Introduction to Modern Information Retrieval, Janvier 1983. McGraw-Hill, New York (1983)
7. Sauvagnat, K., Boughanem, M.: Propositions pour la pondération des termes et l'évaluation de la pertinence des éléments en recherche d'information structurée. In: 3th COnférence en Recherche d'Informations et Applications (CORIA 2006), Lyon, France (March 2006)
8. Savoy et, J., Rasolofo, Y.: Report on the TREC-9 Experiment: Link-based Retrieval and Distributed Collections. In: 9th Text REtrieval Conference, Gaithersburg, Maryland, United States (November 2000)

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 exposes the results of our participation in INEX'06. Two runs were submitted to the Ad Hoc Thorough track obtained with Garnata, our Information Retrieval system for structured documents. We have implemented 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, in this edition it is the first time that their members submit a run to the official tasks. Until now, our contribution to INEX had been the design of several topics and the assessments of relevance judgements, but now we have participated with a new experimental platform to perform structured retrieval using Probabilistic Graphical Models.

We have participated in the Ad Hoc Track (the Thorough subtask) with the results given by two models based on Influence Diagrams [5]: the Simple Influence Diagram Model (SID) and the Context-based Influence Diagram Model (CID) [2,3]. They 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]. As fas as we know, these models are the first attempts to apply Influence Diagrams to structured retrieval.

It also should be pointed out that the results of this first participation are not good, and in fact clearly disappointing. In fact, we are in the last positions of the ranking in the Thorough task. Perhaps, this bad behaviour could be due to a wrong implementation of the algorithm found in Garnata after studying the poor performance obtained by the SID and CID models, once the official results were published. However, after sharping the implementation fact, the results are

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 165–177, 2007.

still not good enough. Therefore, as future works we propose a list of possible improvements in the algorithms trying to identify the problems in them.

Another reason to get such a classification in the ranking could be the free unspecified parameters of the models. Since they are collection-dependent, those that we used in the experimentation were not the best, because no Wikipedia assessments were disposable at that time. We think that the behaviour of both models could be clearly improved with a more systematic experimentation finding an optimal configuration of the parameters.

In order to describe the models and the software that we have used, this paper is organised as follows: the next section will introduce Influence Diagrams to the reader (the formalism used in the models). 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 unexpected performance of the models. This paper will finish with the conclusions and future works for our system.

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. They also have 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 *utility* 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 networks[1]. 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, they express the profit or the preference degree of the consequences derived from the decision process.

[1] In fact, the subset of an Influence Diagram that consists only of chance nodes is a Bayesian network. Thus, an Influence Diagram can also be viewed as a Bayesian network enlarged with decision and utility nodes.

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

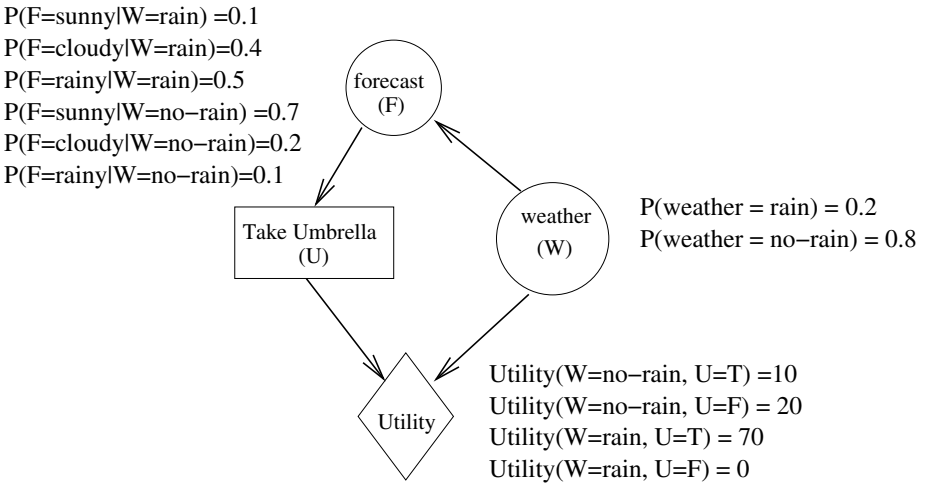


Fig. 1. An example of an Influence Diagram

There are also different types of arcs in an Influence Diagram, which generally represent influence. The arcs between chance nodes represent probabilistic dependences (as it 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, taking 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].

3.1 The Underlying Bayesian Network

In this model, we consider three different kinds of entities which will be represented by the means of three different kinds of random variables. Namely: *index terms*, *basic structural units*, and *complex structural units* (see below for definitions).

Because our models are Influence Diagrams, they are based in an underlying Bayesian networks which represents a structured document set. This Bayesian network will contain two kinds of *nodes*, representing the terms and the structural units. The former will be given by the set $\mathcal{T} = \{T_1, T_2, \ldots, T_l\}$. As stated before, there are two types of structural units: *basic structural units*, those which only contain terms (*leaf nodes* in XML), and *complex structural units* (*non-leaf nodes* in XML), that are composed of other basic or complex units. For those units containing both text and units (appearing often in Wikipedia), we consider them as complex units, and the text of that unit is assigned to a new unit called *virtual unit*[2], a non-retrievable basic unit (see figure 2). The notation for these nodes is $\mathcal{U}_b = \{B_1, B_2, \ldots, B_m\}$ and $\mathcal{U}_c = \{S_1, S_2, \ldots, S_n\}$, 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.

[2] Of course this unit will not appear in the XPath route of its descendants, is only a formalism that allow us using the two different kinds of units explained above.

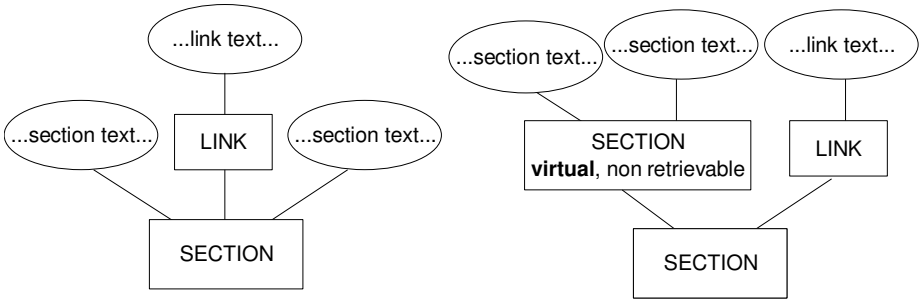


Fig. 2. An example of a virtual unit: after the change, the "section" does not contain text, only other units

Regarding the arcs of the models, there will be an arc from a given node (either term or structural unit) to the particular structural unit the node belongs to. It express 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, term nodes have no parents.

Note 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 $Hi(U) = \text{null}$ 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 $pa(X)$. 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:

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

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

where $w(T, B)$ is a weight associated to each term T belonging to the basic unit B and $w(U, S)$ 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 with the only restrictions that

$$w(T, B) \geq 0, \quad w(U, S) \geq 0, \sum_{T \in Pa(B)} w(T, B) \leq 1, \text{ and } \sum_{U \in Pa(S)} w(U, S) \leq 1.$$

3.2 Constructing the Influence Diagram

Once the Bayesian network has been constructed, it is enlarged by including decision and utility nodes, and so 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 $U_i \in \mathcal{U}$, $\forall i = 1, \ldots, |\mathcal{U}|$. V_i will measure the value of utility for the corresponding decision.

We shall also consider a *global utility node* representing the joint utility of the whole model. This node will be denoted by Σ, meaning we are assuming an additive behavior of the model. In addition to the arcs between chance nodes (those 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, $\forall i = 1, \ldots, |\mathcal{U}|$. These arcs mean that the utility function of V_i depends obviously only on the decision made and the relevance value of the structural unit considered. Finally, the utility node Σ has all the utility nodes $V_{i,j}$ 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 (of course, for those units U where $Hi(U) = \text{null}$, this arc does not exist).
   Again, the utility node Σ will have the same set of parents as in the SID model.

Figure 3 shows an example of both Influence Diagram models: the SID (left-hand side) and the CID (right-hand side).

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 numerical value for the possible combination of the decision node R_i and the chance node

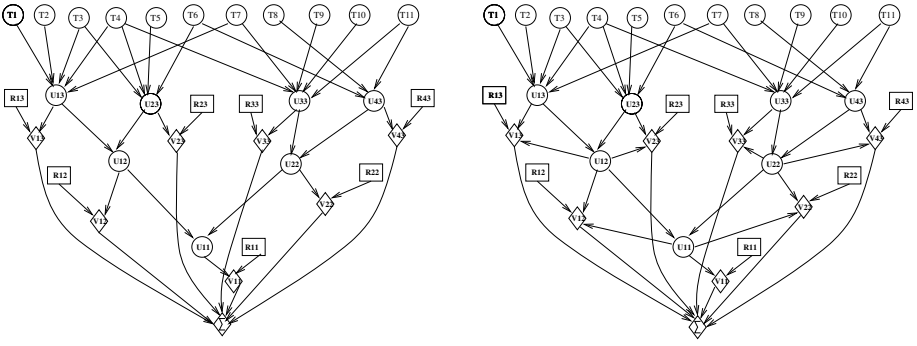


Fig. 3. Influence diagrams for the SID and CID models

representing the structural component U_i. The *four values* are $v(r_i^+|u_i^+)$, $v(r_i^-|u_i^+)$, $v(r_i^+|u_i^-)$ and $v(r_i^-|u_i^-)$.

2. *Utility nodes in CID*: For each utility node V_i we have *eight parameters*, 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 $u_{hi(U_i)} \in \{u_{hi(U_i)}^-, u_{hi(U_i)}^+\}$.

3.3 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 to 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:*

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

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

– *CID Model:*

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

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

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:

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

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

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, the process of 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*. Thus, records of that precomputed weight files are kept in order to provide a fast way to insert one into the index itself in order 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 root unit is found.
- Given a final unit, returning the list of contained terms (using a *direct index*).

The Garnata's indexing subsystem also implements file compression to speed up query processing.

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. The 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 shall have several candidates (those identifiers being the hash inverse of the slot identifier).

For the final units cache, in each slot, we shall store the unit containing greater number of terms (among the candidates). For the complex units cache, in each slot, we shall 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 and technical description of Garnata.

5 Experimental Setting for INEX'06. Analysis of the Results

5.1 Parameter Setting and Official Runs

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

First of all, the weighting scheme used in equations 7 and 8 to compute posterior probabilities has basically been a normalized tf-idf scheme for weights of terms in units. On the other hand, the weights of units included in a complex 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, $\frac{1}{|\mathcal{T}|}$.

Because we are ranking units by the expected utility of retrieving them, $EU(r_i^+ \mid q)$, we only need to assess half of the parameters for each model (those utilities with r_i^+), being two in the SID and four in the CID. 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 $v(r_i^-|u_i^+) = 0$. As stated before, $v(r_i^-|u_i^+)$ and $v(r_i^-|u_i^-)$ are not used. In the context of the CID model, and following the idea of trying to return a unit only when its container is not relevant, we shall fix these two parameters: $v(r^+|u^+, w^-) = 1$ and $v(r^-|u^+, w^-) = 0$, and set the rest to 0. Therefore, the utility values are those in Table 1.

Table 1. Utility configuration for the CID model

v^+_{--}	v^+_{-+}	v^+_{++}	v^+_{+-}
0.0	0.0	0.0	1.0

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, which is the 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.

For the case of the CID model in the Thorough task, this setting is clearly counterproductive reducing the RSV of lots of units whose container is relevant. This fact will be discussed in the next subsection.

We should note that we finally compute RSV as $EU(r_i^+ \mid q) \cdot \mathrm{nIdf}_Q(u_i)$, being $\mathrm{nIdf}_Q(u_i)$ the proportion of the idf of the query terms contained in this unit or in their ancestors, that is

$$\mathrm{nIdf}_Q(u_i) = \frac{\sum_{p \in Q \cap u_i} \mathrm{idf}(p)}{\sum_{t \in Q} \mathrm{idf}(t)}.$$

Thus, $\mathrm{nIdf}_Q(u_i) \in [0, 1]$, and acts as a correcting factor for the utility (the more query terms a units contains, the more interesting for the user it is).

With these experimental settings, according to the results published in the evaluation section of the INEX'06 website [3], considering "Metric:ep-gr, Quantization: gen, Overlap=off", we have obtained a mean of effort-precision of 0.0004 with the runs of obtained by Garnata for the two models, occupying the disappointing 97^{th} and 98^{th} positions in the Thorough task.

5.2 Analysis of the Results

Studying the reasons of this bad behaviour, we discovered several bugs in the software, by which, in some cases, instead of retrieving a certain unit, we were returning a neighbour. Basically, the software returned a completely different unit to that which should have been returned. This has been fixed now.

About possible improvements, on the one hand, we have shown that the set of parameters used for the CID model should not perform well in this task. That set would clearly perform better choosing the best entry point for an XML file, but it is not productive in the Thorough task. We have carried an unofficial run changing the parameters in 1 to $(0, 1, 1, 1)$ and we have obtained a MAep of 0.0015. This is a 375% better than our previous result, but clearly not enough.

On the other hand, using nIdf_Q as a correcting factor sets the RSV of a unit which does not contain query terms to be zero. Since we are using a representation called *virtual unit* for units containing both text and units, the contained units will get a RSV of 0 if they do not contain query terms. However this is not what is supposed in the evaluation procedure, which assigns a positive RSV to a unit without query terms contained into other unit with relevant text. That fact is very common in the Wikipedia collection (assessments contain plenty of "Collectionlinks" without relevant text) but we are not managing that fact correctly.

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

6 Conclusions and Future Works

In this paper, we have presented the SID and CID models, and Garnata (the Information Retrieval System which implements them). These are the retrieval tools that we have used to participate in this our first edition of INEX, in the Thorough task. The results are very disappointing as we are at the bottom of the ranking. We have explained several reasons of this situation that we shall solve in future editions of the INEX workshop.

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 should make some improvement.

Also, 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, using the underlying formalism of Influence Diagrams, are specially designed to make decisions considering the context, and it could perform acceptably.

Also, we are developing new models based in Probabilistic Graphical Models, which improve the performance of the SID and CID models, avoiding the limitations of the current. With these actions, we hope to have a better performance in the future.

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. de Campos, L.M., Fernández-Luna, J.M, Huete, J.F.: The BNR model: foundations and performance of a Bayesian network-based retrieval model. Int. J. Appr. Reason. 34, 265–285 (2003)
2. de Campos, L.M., Fernández-Luna, J.M., Huete, J.F.: Using context information in structured document retrieval: An approach using Influence Diagrams. Inform. Process. Manag. 40(5), 829–847 (2004)
3. de Campos, L.M., Fernández-Luna, J.M., Huete, J.F.: Improving the Context-based Influence Diagram for Structured Retrieval. In: Losada, D.E., Fernández-Luna, J.M. (eds.) ECIR 2005. LNCS, vol. 3408, pp. 215–229. Springer, Heidelberg (2005)
4. de Campos, L.M., Fernández-Luna, J.M., Huete, J.F., Romero, A.E.: Garnata: An Information Retrieval System for Structured Documents based on Probabilistic Graphical Models. In: Proceedings of the Eleventh International Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU), pp. 1024–1031 (2006)
5. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer, Heidelberg (2001)
6. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes. Morgan Kaufmann, San Francisco (1999)

7. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, San Mateo. Morgan Kaufmann, San Francisco (1988)
8. Shachter, R.: Evaluating Influence Diagrams. Oper. Res. 34, 871–882 (1986)
9. Shachter, R.: Probabilistic Inference and Influence Diagrams. Oper. Res. 36(5), 527–550 (1988)

Information Theoretic Retrieval with Structured Queries and Documents

Claudio Carpineto[1], Giovanni Romano[1], and Caterina Caracciolo[2]

[1] Fondazione Ugo Bordoni, Rome, Italy
{carpinet,romano}@fub.it
[2] Food and Agriculture Organization of the UN (FAO), Rome, Italy
caterina.caracciolo@fao.org

Abstract. In this paper we present an extension of information retrieval based on Kullback-Leibler divergence (with backoff smoothing) to support structured queries on structured documents. The proposed method applies to several common retrieval tasks characterized by an implication relationship among texts, including fielded topics and XML documents. We discuss how to choose the method parameters to make the computation of the ranking function efficient. We finally report some experimental results obtained using a loose approximation of the model based on a discriminative selection strategy.

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 types of 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 an approximation of the presented model based on a discriminative selection strategy. A full implementation of the model and a more significant evaluation of its retrieval effectiveness are left for future work.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 178–184, 2007.

2 Information-Theoretic Retrieval

Given a query Q and a document D, the score of D relative to 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:

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

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

3 Structured Information-Theoretic Retrieval

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]. One well known and more closely related approach [5] was based on KL divergence with linear interpolation (rather than backoff smoothing) and used distinct language models and parameters associated with every node of the tree of the document.

We take a different approach to model the implication relationship among documents (we speak of documents rather than elements, meaning the smallest individual text units). 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 the words contained in D itself; i.e., $D^* = \{w \,|\, w \in D_i \backslash D, D \leq D_i\}$. This is a very general assumption that holds for a number of cases of interest characterized by an implication relationship among texts, including XML documents, fielded topics, thesauri, and retrieval feedback.

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.

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

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

$$\sum_{w \in D} \alpha \frac{c(w,D)}{|D|} + \sum_{w \in D^*} \beta \frac{c(w,D^*)}{|D^*|} + \sum_{w \notin D \cup D^*} \mu \, p(w|C) =$$

$$= \alpha + \beta + \sum_{w \notin D \cup D^*} \mu \frac{c(w,C)}{|C|} = 1 \tag{4}$$

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.

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

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

$$\gamma \; + \; \delta \; + \sum_{w \; \notin Q \cup Q^*} \pi \, \frac{c(w,C)}{|C|} = 1 \tag{6}$$

Thus, in all we have six parameters (i.e., $\alpha, \beta, \mu, \gamma, \delta, \pi$) and two equations (i.e., equations 4 and 6). Note that these parameters will, in general, be different for each query and each document. We now discuss how to estimate the parameters in a more compact and elegant way. Then we will 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 query and each document.

To make the computation of probability estimation more manageable, we can assume that the parameters are not independent. One possible choice is to assume that

$$\beta = k_1 \, \alpha, \;\; \mu = k_2 \, \alpha \tag{7}$$

and

$$\delta = k_1 \, \gamma, \;\; \pi = k_2 \, \gamma \tag{8}$$

The intuitive meaning of this assumption is that the the weight assigned to logically related or unseen terms should be a fraction of the weight of original terms, regardless of whether you are considering queries or documents. The values for k_1 and k_2 can be chosen using an inverse function of the average length of $Q, D, Q \cup Q^*$, and $D \cup D^*$, for instance, or using other collection statistics. Alternatively, they can be set experimentally with the goal of optimizing a retrieval performance function, provided that training data are available.

Once the values for k_1 and k_2 have been determined, it is sufficient to substitute such values in equations 4 and 6 and then find α and γ using the collection statistics found at indexing time. The next step is the computation of the KL scores by expression 1, using the values for α and γ found at the earlier step and expressions 7 and 8 to compute the probabilities in expressions 3 and 5 .

In order to compute expression 1, it is convenient to partition the overall set of terms in small subsets whose corresponding KL scores are easier to compute. Consider first that each term can either belong to Q, or Q^*, or $\overline{Q \cup Q^*}$. Also, each term can either belong to D, or D^*, or $\overline{D \cup D^*}$. The whole set of terms can thus be expressed by taking the union of all possible intersections between the Q-partitions and the D-partitions, i.e,

$$T = (Q \cap D) \; \cup \; (Q \cap D^*) \; \cup \; (Q \cap \overline{D \cup D^*}) \; \cup \; (Q^* \cap D) \; \cup \; (Q^* \cap D^*) \; \cup \; (Q^* \cap \overline{D \cup D^*}) \; \cup \; (\overline{Q \cup Q^*} \cap D) \; \cup \; (\overline{Q \cup Q^*} \cap D^*) \; \cup \; (\overline{Q \cup Q^*} \cap \overline{D \cup D^*})$$

The value of $KL(\theta_Q|\theta_D)$ can thus be computed in the following way.

$$KL(\theta_Q|\theta_D) = \sum_{w \in Q \cap D} \gamma \, \frac{c(w,Q)}{|Q|} \; log \frac{\gamma \, \frac{c(w,Q)}{|Q|}}{\alpha \frac{c(w,D)}{|D|}} \; + \sum_{w \in Q \cap D^*} \gamma \, \frac{c(w,Q)}{|Q|} \; .$$

$$log\frac{\gamma\ \frac{c(w,Q)}{|Q|}}{k_1\alpha\frac{c(w,D^*)}{|D^*|}} + \sum_{w\in Q\cap\overline{D\cup D^*}} \gamma\,\frac{c(w,Q)}{|Q|}\ log\frac{\gamma\ \frac{c(w,Q)}{|Q|}}{k_2\alpha\frac{c(w,C)}{|C|}} + \sum_{w\in Q^*\cap D} k_1\gamma\,\frac{c(w,Q^*)}{|Q^*|}\ \cdot$$

$$log\frac{k_1\gamma\ \frac{c(w,Q^*)}{|Q^*|}}{\alpha\frac{c(w,D)}{|D|}} + \sum_{w\in Q^*\cap D} k_1\gamma\,\frac{c(w,Q^*)}{|Q^*|}\ log\frac{k_1\gamma\ \frac{c(w,Q^*)}{|Q^*|}}{\alpha\frac{c(w,D)}{|D|}} +$$

$$\sum_{w\in Q^*\cap D^*} k_1\gamma\,\frac{c(w,Q^*)}{|Q^*|}\ log\frac{k_1\gamma\ \frac{c(w,Q^*)}{|Q^*|}}{k_1\alpha\frac{c(w,D^*)}{|D^*|}} + \sum_{w\in Q^*\cap\overline{D\cup D^*}} k_1\gamma\,\frac{c(w,Q^*)}{|Q^*|}\ \cdot$$

$$log\frac{k_1\gamma\ \frac{c(w,Q^*)}{|Q^*|}}{k_2\alpha\frac{c(w,C)}{|C|}} + \sum_{w\in\overline{Q\cup Q^*}\cap D} k_2\gamma\,\frac{c(w,C)}{|C|}\ log\frac{k_2\gamma\ \frac{c(w,C)}{|C|}}{\alpha\frac{c(w,D)}{|D|}} +$$

$$\sum_{w\in\overline{Q\cup Q^*}\cap D^*} k_2\gamma\,\frac{c(w,C)}{|C|}\ log\frac{k_2\gamma\ \frac{c(w,C)}{|C|}}{k_1\alpha\frac{c(w,D^*)}{|D^*|}} + \sum_{w\in\overline{Q\cup Q^*}\cap\overline{D\cup D^*}} k_2\gamma\,\frac{c(w,C)}{|C|}\ \cdot$$

$$log\frac{k_2\gamma\ \frac{c(w,C)}{|C|}}{k_2\alpha\frac{c(w,C)}{|C|}}$$

Note that the the first eight summands only involve terms that are present in Q, or Q^*, or D or D^*, and that the last summand can be efficiently computed by considering the complement of the sum of the probabilities of the terms contained in $Q \cup Q^* \cup D \cup D^*$. Thus, in order to compute $KL(\theta_Q|\theta_D)$, it is necessary to take into account only the probabilities of the seen terms and the probabilities of the terms that are logically related to them.

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 loose approximation of it.

For each query, we first collected the first 200 results retrieved by a plain commercial search engine from the online Wikipedia collection in response to the query title. Then we matched the titles of the retrieved documents against the titles of the documents contained in the INEX 2006 collection to extract a set of potentially relevant documents. Clearly, this process was prone to errors and omissions that may have hurt the final retrieval effectiveness, partly because the contents of Wikipedia documents usually change over time while the INEX

collection is not updated, and partly because we took into account only a small fraction of the documents in the INEX collection.

We then performed an element level analysis for each INEX 2006 retrieved document to choose the best element(s) according to a straightforward application of the KL divergence. Considering only the terms in the query title and observing that the probability of terms in the query title was constant for each query, the determination of $score(Q, D)$ reduced to:

$$score(Q, D) = -p(w|\theta_Q) \sum_{w \in V} (logp(w|\theta_Q) - logp(w|\theta_D)) \cong \sum_{w \in V} logp(w|\theta_D)$$

In practice, we considered only the four most general XML elements in the document tree and used the maximum likelihood estimate to compute $p(w|\theta_D)$. We finally reordered the elements in each document according to their KL scores and used the combined document-element ranks to submit runs in three tasks: focused, all in context, and best in context.

The first stage amounted 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, a larger set of features was used (essentially intra-document elements) to perform fine selection/reordering of the results retrieved in the first stage.

It is conceivable that this two-step strategy may be useful not only to increase efficiency but also effectiveness, because optimized tools for retrieving full documents are already available and this may guarantee a high quality pre-filtering of elements. In the second stage, the full set of features (e.g., elements, document structure, query structure) may be brought to bear to maximize retrieval performance, although we were unfortunately unable to fully experiment with this approach.

The retrieval performance of our runs 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 at least represent an indication that:

- The KL divergence may be effectively used to perform intra-document element ranking.

- The discriminative selection strategy may be useful independently of how the documents are next processed.

5 Conclusions

We presented an extension of KL divergence-based information retrieval that supports retrieval from structured documents with structured queries. We discussed parameter estimation and efficiency issues. We also reported some preliminary effectiveness results using a loose approximation of the model based on a discriminative selection strategy. The next steps of this research are (1) to implement and experiment with the full model and (2) to test the hypothesis that a discriminative selection strategy is indeed useful for XML retrieval.

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, New Orleans, LA, USA, pp. 172–180 (2001)
3. Katz, S.: Estimation of probabilities from sparses 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, New Orleans, LA, USA, pp. 111–119 (2001)
5. Ogilvie, P., Callan, J.: Language Models and Structured Document Retrieval. In: Proceedings of the INEX 2002 Worksop, Schloss Dagsthul, Germany, pp. 33–40 (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)

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

Eugen Popovici, Gildas Ménier, and Pierre-François Marteau

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

Abstract. In this paper we report on the retrieval approach taken by the VALORIA laboratory of the University of South-Brittany while participating at INEX 2006 ad-hoc track with the SIRIUS XML IR system. SIRIUS retrieves relevant XML elements by approximate matching both the content and the structure of the XML documents. A 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 content of the retrieved elements. We briefly describe the approach and the extensions made to the SIRIUS XML IR system to address each of the four subtasks of the INEX 2006 ad-hoc track. Finally we present and analyze the SIRIUS retrieval evaluation results. SIRIUS runs were ranked on the 1st position out of 77 submitted runs for the Best In Context task and obtained several top ten results for both the Focused and All In Context tasks.

1 Introduction

This 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 [1] 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, a new set of topics and new tasks, ii) the implementation of the approximate search and indexing process using a distributed inverted file architecture; and iii) the use of selective indexing profiles defining how the structure and the content of XML tags should be indexed. 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, All In Context and Best In Context tasks. Finally, in Section 5 we conclude the paper.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 185–199, 2007.

2 SIRIUS XML IR System

SIRIUS [2, 3] 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*. The matching process takes into account mismatched errors both on the attributes and on the XML elements. It 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.

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:

$$p(n)=<n_0, A(n_0)> <n_1, A(n_1)> \ldots <n, A(n_n)> . \quad (1)$$

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 $p(n)$ <*ctxtId*>.

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 [4]) 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 [5]. 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 [6]. 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 [7] function for this purpose:

$$v_i(e_i, CO) = \xi \cdot \sum_k \lambda_k . \tau_k \ . \tag{2}$$

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 $\xi = 1 / \Sigma_k (\lambda_k)$;

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:

$$\sigma(p^R, p_i^D) = 1/(1 + \delta_L (p^R, p_i^D)) \ . \tag{3}$$

where δ_L is a dedicated editing distance (see [8]).

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 [9]. 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 [10] 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 **[1]** 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: $C_{subst}(n, n')=\{ \xi$ *if* $(n \neq n'); \frac{1}{2}\cdot\xi$ *if* $(n = n') \ \& \ (\neg attCond(n')); 0$ *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 [11] computes the best $Seq(p_i^D, p^R)$ (i.e. minimizes $GC()$ *cost)* with a complexity of $O(length(p_i^D) * length(p^R))$ as stated earlier. Let

$$\delta_L(p^R, p_i^D,) = Min_k\, GC(Seq_k(p^R, p_i^D)) . \tag{4}$$

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:

$$v'_i = \beta \cdot \sigma(p^R, p_i^D) + (1 - \beta) \cdot v_i . \tag{5}$$

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 [10] 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

SIRIUS has 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* [12], 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 [13]. The index model (Section 2.1) is implemented on top of the Berkeley DB[1] library using a combination of BTrees and Hashtables structures. The inverted file index is constructed in parallel by using a *Physical Document Partitioning* approach [14]. 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 (see eq. 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.

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 [10]. 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 (see eq. 3) involving specific heuristics for attributes and attributes values [1]. Finally we merge the content and structural match scores using a weighted linear aggregation method (see eq. 5).

[1] http://www.sleepycat.com/

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 *//A//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 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 a 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 $v'_i = (v_i + Max_j (v_j)) / 2$ where $\exists e_j^D \in \{ (e_j, v_j) \}$.

Using this approach, the target elements without 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, All In Context and Best In Context [15]. 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 [10] to SIRIUS recursive query language as described in [1].

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 eq. 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, eq. 2 and 3) .

– We use weighted linear aggregation for content and structure matching scores. (see eq. 5) 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. β=0.1 biases the ranking towards the textual content while β=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 and Table 2 the evaluation curves for the ep/gr evaluation metric and the ranks obtained by all the submitted runs. The results may contain overlapping elements (i.e. Overlap=off). Details of the evaluation metrics can be found in [16].

Table 2. Task: Thorough, Metric:ep-gr, Quantization: gen, Overlap=off, R: rank/106 runs

			filtered assessments	
RunId	*MAep*	*R*	*MAep*	*R*
IDF_BOOL_noSEQ	**0.0158**	**42**	**0.0296**	**37**
IDF_BOOL	0.0151	45	0.0287	42
casEDsW0_1_IDF_BOOL_noSEQ	0.0146	48	0.0274	43
casEDsW0_5_IDF_BOOL_noSEQ	0.0134	50	0.0253	45
casEDsW0_5_IDF_BOOL	0.0130	51	0.0242	48

We obtained average rankings for the Thorough task. 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 quality of the retrieved results. Rather the opposite. 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 and respectively 0.0296 when evaluated against the *filtered assessments*[2].

4.2 Focused Task

The aim of the Focused 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. For the Thorough task we considered 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.

[2] "element links" (i.e. collectionlink, wikipedialink, redirectlink, unknownlink, outsidelink and weblink) in the assessments have been given an exhaustive value of ? (corresponding to "too small" in INEX 2005 relevance definition).

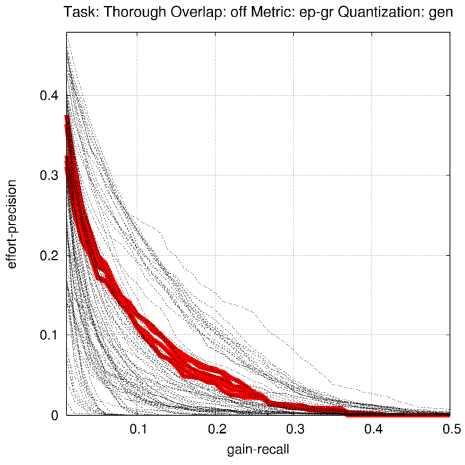


Fig. 2. Task: Thorough, Metric:ep/gr , Quantization: gen, Overlap: off, Filtered assessments

We implemented a two steps post filtering process to remove the overlapping elements from the results list [1]: i) we recalculate the relevance of the elements in the answer list in order to reflect the relevance of their descendants elements (if any); and ii) we select non overlapping elements from the list.

The weights are calculated in a bottom-up manner from the 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 [1]. 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.

We report here the nxCG values @5, @10, @25 and @50 (see [16] for metric descriptions) for all the submitted focused runs, along with their official ranks in the INEX06 campaign. The runs are evaluated on both the original (see Tables 3 and 5) and filtered assessments (see Tables 4, 6 and Fig. 3).

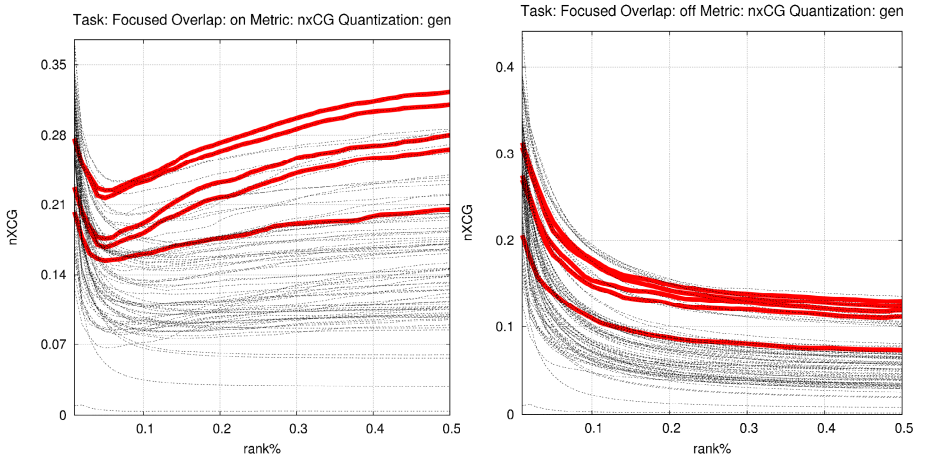


Fig. 3. Task: Focused Metric:nxCG, Quantization: generalised, Filtered assessments, Overlap= on (left) ; and Overlap=off (right)

Table 3. Task: Focused, Metric: nxCG, Quantization: generalised, Overlap=on, R: rank/85 runs

RunId	*nxCG@5*	*R*	*nxCG@10*	*R*	*nxCG@25*	*R*	*nxCG@50*	*R*
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=on, Filtered assessments, R: rank/85 runs

RunId	*nxCG@5*	*R*	*nxCG@10*	*R*	*nxCG@25*	*R*	*nxCG@50*	*R*
IDF_BOOL_noSEQ_MAX_MRD	0.2832	48	**0.2752**	**23**	**0.2475**	**9**	**0.2289**	**4**
IDF_BOOL_MAX_MRD	**0.2840**	**46**	0.2679	28	0.2469	10	0.2211	7
casEDsW0_5_IDF_BOOL_noSEQ_MAX_MRD	0.2297	66	0.2218	59	0.2039	31	0.1782	23
casEDsW0_5_IDF_BOOL_MAX_MRD	0.2301	64	0.2196	61	0.2004	33	0.1709	32
casEDsW0_1_IDF_BOOL_noSEQ_Foc_MAX_HA	0.2043	73	0.2012	63	0.1757	47	0.1562	42

Table 5. Task: Focused, Metric: nxCG, Quantization: generalised, Overlap=off, R: rank/85 runs

RunId	*nxCG@5*	*R*	*nxCG@10*	*R*	*nxCG@25*	*R*	*nxCG@50*	*R*
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

Table 6. Task: Focused, Metric: nxCG, Quantization: generalised, Overlap=off, Filtered assessments, R: rank/85 runs

RunId	*nxCG@5*	*R*	*nxCG@10*	*R*	*nxCG@25*	*R*	*nxCG@50*	*R*
IDF_BOOL_noSEQ_MAX_MRD	**0.3227**	**37**	**0.3238**	**14**	**0.2805**	**12**	**0.2440**	**9**
IDF_BOOL_MAX_MRD	0.3180	39	0.3084	20	0.2766	14	0.2353	12
casEDsW0_5_IDF_BOOL_noSEQ_MAX_MRD	0.2770	59	0.2829	33	0.2477	18	0.2082	17
casEDsW0_5_IDF_BOOL_MAX_MRD	0.2719	61	0.2726	39	0.2416	23	0.2011	18
casEDsW0_1_IDF_BOOL_noSEQ_Foc_MAX_HA	0.2073	73	0.2063	64	0.1780	54	0.1483	43

The MAX_MRD method for overlap removal retrieves more focused elements and seems to be more adequate for the focused task than its MAX_HA competitor. This may be considered with care as the results are influenced with different degrees by the structural matching process.

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 several results in the best top ten runs using the nxCG@25 and nxCG@50 metrics (see Tables 3, 4, 5 and 6; top ten results are highlighted, best results are in bold characters).

By analyzing the overall comportment of nxCG curves of Fig. 3 we observe that SIRIUS runs have a good recall. We also observe a slightly decrease in the system retrieval performance for the first ranked results. 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. This hypothesis is sustained by the slight increase in the SIRIUS performance when evaluating the runs against the filtered assessments. The differences are not major as the indexing profiles are not an exact match of the rules used to obtain the filtered assessments. The indexing profile eliminated only a part of the tags defined as "too small". When evaluating the runs against the filtered assessments the remaining element links (i.e. *redirectlink*, *wikipedialink* and *weblink*) as well as the eliminated tags but that were considered relevant by the assessors (*emph2*, *emph3*, *title*, *name*, and *caption*) [17] penalize the results. We observe that as for the Thorough task, the runs involving structural conditions performed worse than their content only pairs.

4.3 All in Context Task

For the INEX 2006 All In Context task, the systems have 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 clustered the non overlapping results by file and ranked them according to their relevance inside each file. We set the article score equal to the most relevant element occurring inside each file. The files are ranked by their relevance. 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 official and additional SIRIUS evaluation results for this task are given in Tables 7, 8, 9 and Fig. 4 (left) (top ten results are highlighted, best results are in bold characters).

Table 7. Task: All In Context (Article level), Metric: hixeval-article, R: rank/62 runs

RunId	*F[5]*	*R*	*F[10]*	*R*	*F[25]*	*R*	*F[50]*	*R*	*MAP*	*R*
IDF_BOOL_MAX_MRD_10	**0.1028**	**44**	**0.1150**	**40**	0.1181	39	0.1055	33	0.0752	39
IDF_BOOL_noSEQ_MAX_MRD_10	0.1027	45	0.1132	41	**0.1203**	**38**	**0.1079**	**32**	**0.0759**	**38**
casEDsW0.5_IDF_BOOL_noSEQ_MAX_HA_5	0.0881	50	0.0871	53	0.0966	48	0.0864	45	0.0623	48
casEDsW0.5_IDF_BOOL_MAX_MRD_5	0.0867	52	0.0910	52	0.0956	49	0.0864	46	0.0613	49
casEDsW0.1_IDF_BOOL_noSEQ_AVG_HA_5	0.0373	57	0.0457	57	0.0549	57	0.0533	57	0.0299	58

Table 8. Task: All In Context (Element level), Metric: hixeval-element, Overlap=off, R: rank/57 runs

RunId	*hixeval-element-intersection*		*hixeval-element-union*	
	F-avg	*R*	*F-avg*	*R*
casEDsW0.1_IDF_BOOL_noSEQ_AVG_HA_5	**0.4695**	**2**	0.2845	24
casEDsW0.5_IDF_BOOL_noSEQ_MAX_HA_5	0.4677	3	0.3306	15
IDF_BOOL_noSEQ_MAX_MRD_10	0.4658	5	**0.3492**	**8**
IDF_BOOL_MAX_MRD_10	0.4650	6	0.3464	9
casEDsW0.5_IDF_BOOL_MAX_MRD_5	0.3948	32	0.2752	31

Table 9. Task: All In Context (combining Article and Element levels scores), Metric: generalized Precision/Recall, R: rank/56 runs

RunId	*gP[5]*	*R*	*gP[10]*	*R*	*gP[25]*	*R*	*gP[50]*	*R*	*MAgP*	*R*
IDF_BOOL_MAX_MRD_10	**0.2245**	**32**	**0.2164**	**24**	**0.1801**	**15**	0.1386	13	**0.1414**	**15**
IDF_BOOL_noSEQ_MAX_MRD_10	0.2231	33	0.2117	25	0.1779	16	**0.1417**	**11**	0.1408	16
casEDsW0.5_IDF_BOOL_noSEQ_MAX_HA_5	0.1881	43	0.1654	41	0.1414	33	0.1141	29	0.1133	29
casEDsW0.5_IDF_BOOL_MAX_MRD_5	0.1642	45	0.1553	44	0.1325	35	0.1059	33	0.1021	34
casEDsW0.1_IDF_BOOL_noSEQ_AVG_HA_5	0.1586	47	0.1448	46	0.1245	40	0.0988	34	0.0868	38

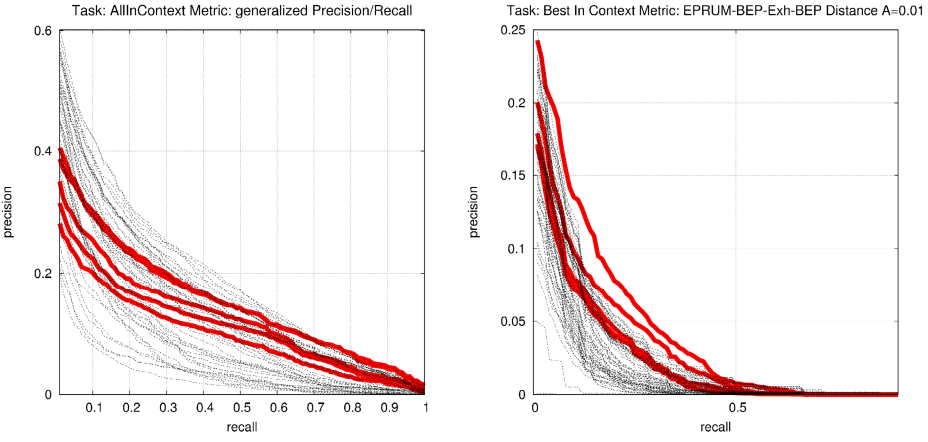


Fig. 4. Task: All In Context (combining Article and Element levels scores), Metric: generalized Precision/Recall (left) ; Task: Best In Context. Metric:EPRUM-BEP-Exh-BEPDistance, A=0.01 (right).

SIRIUS obtained relatively good rankings for the All In Context task (see Table 9). We may get an insight at the SIRIUS retrieval performance by analyzing All In Context Task additional scores for the article-level (Table 7) and element-level (Table 8). The element-level scores show that SIRIUS is able to detect and extract the amount of retrievable relevant information within an article with very good results. Unfortunately, SIRIUS retrieval performance highly degrades when evaluated at article-level. A possible way to improve the retrieval performances of the system is to rank the files using a global relevance value computed at article level.

4.4 Best in Context Task

For the Best In Context task we had to retrive a ranked list of articles. For each article, we must return 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 All In Context Task with N set to 1. The official results evaluated with BEP-D (see Table 10) and EPRUM-BEP-Exh-BEPDistance [18] (see Table 11) were ranked several times in the top ten positions out of 77 submitted runs (see Fig. 4 – right). The top ten results are highlighted while the best obtained values are in bold characters.

Table 10. Task: Best In Context. Metric: BEPD, R: rank/77 runs.

RunId	*A=0.01*	*R*	*A=0.1*	*R*	*A=1*	*R*	*A=10*	*R*	*A=100*	*R*
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 11. Task: Best In Context. Metric:EPRUM-BEP-Exh-BEPDistance, R: rank/77 runs.

RunId	*A=0.01*	*R*	*A=0.1*	*R*	*A=1*	*R*	*A=10*	*R*	*A=100*	*R*
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 10.).

5 Conclusions

This year, at INEX 2006, we have pursuit the evaluation of the retrieval performances of the SIRIUS XML IR system [2, 3] started last year within the INEX 2005 campaign [1]. 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 content of the retrieved elements. A number of extensions were brought to the system in order to cope with the requirements of the Thorough, Focused, All 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, All In Context and Best In Context tasks. SIRIUS obtained average rankings for the Thorough task and top ten ranked results in the range of the 50 first retrieved answers for the Focused and All In Context task. For Best In

Context task the results were quite encouraging as the system was ranked on the 1st place out of 77 submissions for both BEPD and EPRUM metrics with A=0.01[3]. (see Tables 10, 11).

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 relaxed variants.

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 analyzing the use of structural hints within the XML IR requests are necessary to better understand the reasons for this behaviour.

References

1. Popovici, E., Ménier, G., Marteau, P.-F.: SIRIUS: A Lightweight XML Indexing and Approximate Search System at INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 321–335. Springer, Heidelberg (2006)
2. Ménier G., Marteau P.F.: Information retrieval in heterogeneous XML knowledge bases, IPMU, July 1-5, 2002, Annecy, France (2002)
3. Ménier, G., Marteau, P.F.: PARTAGE: Software prototype for dynamic management of documents and data. In: ICSSEA, 29 November-1 December, 2005, Paris, France (2005)
4. Clark, J., DeRose, S.: XML Path Language (XPath) Version 1.0, W3C Recommendation, November 16 (1999) http://www.w3.org/TR/xpath.html
5. Carmel, D., Maarek, Y.S., Mandelbrod, M., Mass, Y., Soffer, A.: Searching XML documents via XML fragments, SIGIR 2003, Toronto, Canada, pp. 151–158 (2003)
6. Amer-Yahia, S., Koudas, N., Marian, A., Srivastava, D., Toman, D.: Structure and Content Scoring for XML, VLDB, Trondheim, Norway, pp. 361–372 (2005)
7. Salton, G., Buckeley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management 24, 513–523 (1988)
8. Levenshtein, A.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Sov.Phy. Dohl. 10, 707–710 (1966)
9. Mignet, L., Barbosa, D., Veltri, P.: The XML Web: A First Study, WWW 2003, May 20–24, Budapest, Hungary (2003)
10. Trotman, A., Sigurbjörnsson, B.: Narrowed Extended XPath I (NEXI). In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 16–40. Springer, Heidelberg (2005)
11. Wagner, R., Fisher, M.: The String-to-String Correction Problem. Journal of the Association for Computing Machinery 12(1), 168–173 (1974)
12. Tannier, X., Girardot, J.-J., Mathieu, M.: Classifying XML Tags through Reading Contexts. In: DocEng, Bristol, United Kingdom, pp. 143–145 (2005)
13. Porter, M.F.: An algorithm for suffix stripping, Program. Program 14(3), 130–137 (1980)

[3] Note that high values of A (e.g. 10) does not discriminate whether the answer is near to or far from the BEP. Whereas, low values of A (e.g. 0,1) favour runs that return elements very close to a BEP.

14. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press. Addison-Wesley, New York (1999)
15. Clarke C., Kamps J., Lalmas M.: INEX 2006 Retrieval Task and Result Submission Specification. In: INEX 2006 Workshop Pre-Proceedings, Dagstuhl, Germany, December 18–20, 2006, pp. 381–388 (2006)
16. Kazai, G., Lalmas, M.: INEX 2005 Evaluation Metrics. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 16–29. Springer, Heidelberg (2006)
17. Kamps, J., Koolen, M., Sigurbjörnsson, B.: The University of Amsterdam at INEX 2006. In: INEX 2006 Workshop Pre-Proceedings, Dagstuhl, Germany, December 18–20, 2006, pp. 88–99 (2006)
18. Piwowarski, B., Dupret, G.: Evaluation in (XML) information retrieval: expected precision-recall with user modelling (EPRUM). In: SIGIR 2006, pp. 260–267 (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

Abstract. Our experiments in the 2006 INEX ad'hoc track were based on the use of the proximity of the query terms in the documents to rank them. More precisely we define around each occurence of a query term an influence function. For an occurence appearing in the text itself, this influence function is linearly decreasing from 1 to 0 depending on the distance to the occurence. When a query term happens to appear in a title of a structured document its influence is uniformly 1 from the beginning to the end of the (sub-)section. We use boolean queries and these influence functions are combined according to the tree of a query using fuzzy logic. The score of any part of a document is the summation of the resulting influence function at the root of the query tree on the range of this part. We present and comment the results.

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 occurences 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 refered 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 developped 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

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 200–212, 2007.

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 developped 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 occurences, 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.* [1] and Hawking *and al.* [2] developped 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 occurence of any of the keywords, the shortest interval that contains all the keywords is selected. So if there are two successive occurences of the same keyword without any occurences 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.* [3]. 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 occurences of two query terms contribute to this additional score.

2.2 Fuzzy Influence Function Model

Beigbeder *et al.* [4] developped a retrieval model based on *the fuzzy proximity of the keywords*. More precisely each occurence of a keyword has a fuzzy influence on its neighbouring. This influence reaches its maximum value *one* at the keyword occurence 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 way to define a 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 occurence.

Given a term the influences of its occurences 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 occurence 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 occurence 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 as lists and emphasis for instance. As a particular case, we consider that a document is the highest level in the sectionning hierarchy.

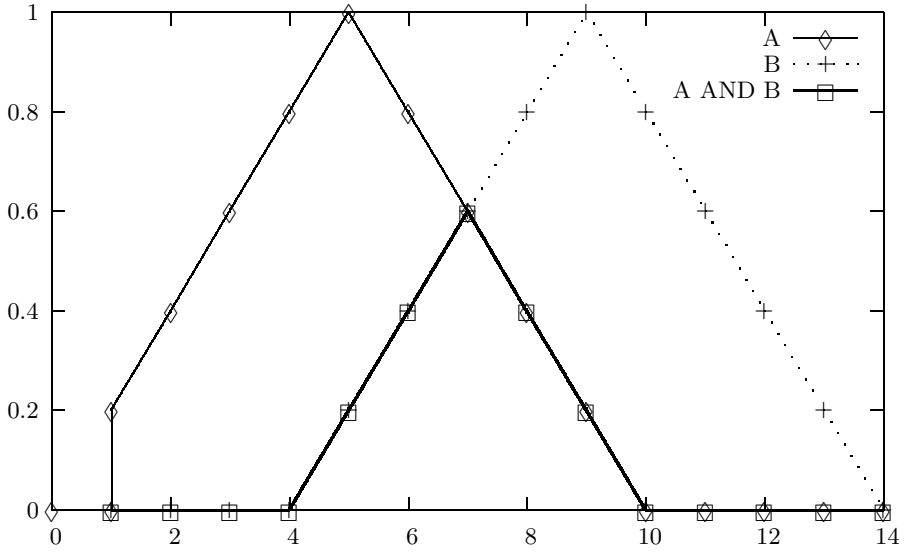


Fig. 1. Proximities to the terms A and B and their combination for the query A AND B: *x-axis:* position of words in the text, *y-axis:* fuzzy proximities

Another point is that sectionning and titles are tightly related so that in the LaTeX 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 LaTeX styles with the sectionning commands only. Here is an example:

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

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

For a term occurence which appears in the *section_text* parts, the basis is the same as in linear text: A decreasing value of the distance to the occurence. But we add another constraint, the influence is limited to the *section_text* part in which the occurence 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 occurence 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 occurences 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 occurence of the term A is in the *title_text* part of the section. Figure 2 shows the propagation of the influence of the occurence 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.

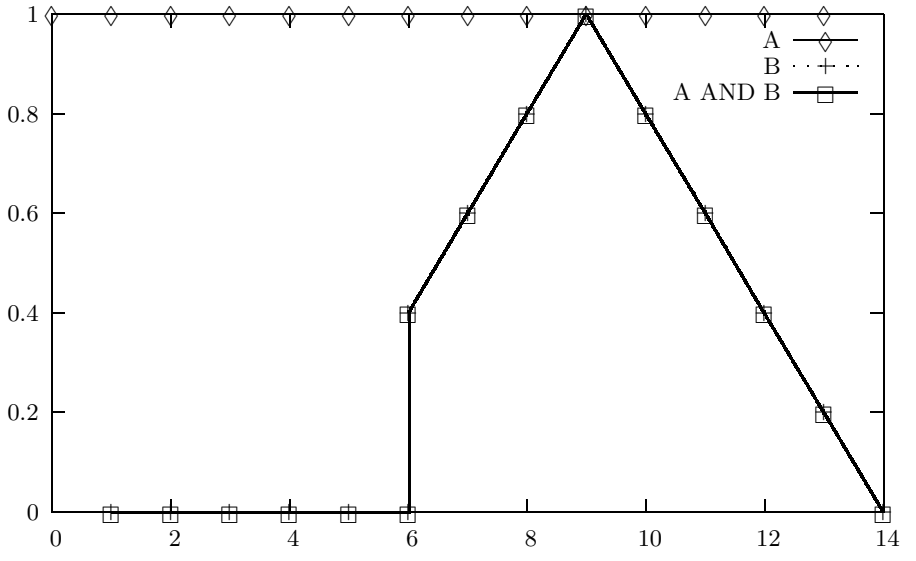


Fig. 2. Limitation of the influence of an occurence to the *section_text* part in which it appears (proximity to the term B); propagation of a title term (proximity to the term A): *x-axis:* position in the text, *y-axis:* fuzzy proximities

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.

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 highligth (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[1] in version 0.5.4. Though it is an outdated version that is now replaced by different versions of ZETTAIR[2], 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 occurence 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 the 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

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

ones (plural vs. singular) or derivationnal 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 conjonction, 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)
```

We built two sets of queries, the short ones where we used only the title keywords with very minor expansion, and the enhanced queries for which we used terms from any field with expansions.

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. With our two query sets we used two values of k, 50 and 200. As only three runs could be submitted, we sent as official runs these combinations except the one with enhanced queries and $k = 50$.

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.

Finaly 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.

6 Results

Whatever the task, our best results were obtained with our set of enhanced queries. With the set of short queries, our two sets of results with $k = 50$ and $k = 200$ are very close one to each other and quite worse than with enhanced

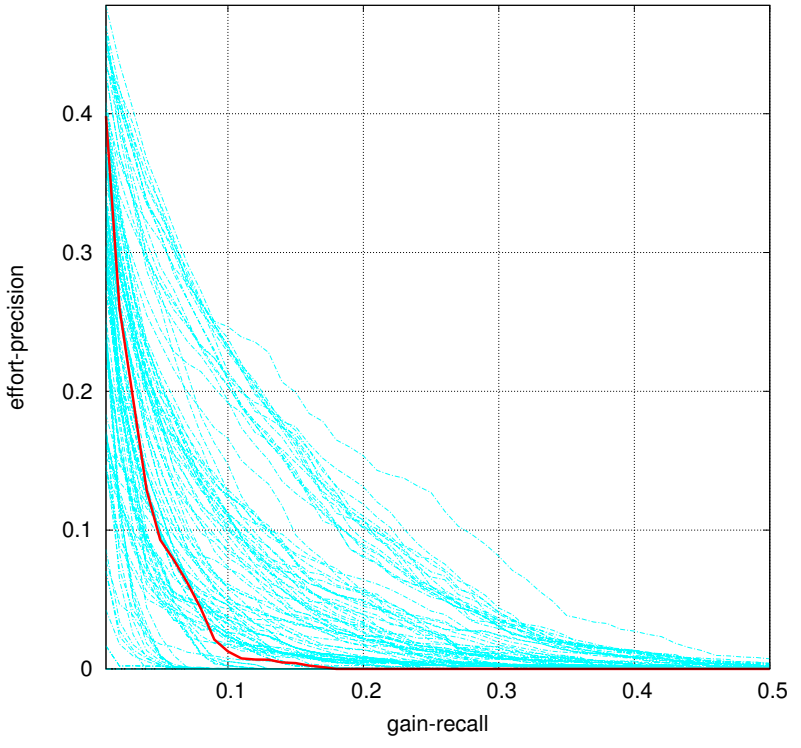


Fig. 3. INEX 2006: Results' Summary, Metric: ep-gr, Quantization: gen, Task: thorough, Run: title_Q_Prox200NT02

queries. This shows that the expansion mechanism is mandatory. More experiments should be done on the value of k to draw some conclusion. In the following all the results are related to the set of enhanced queries.

Figure 3 shows our results summarized with the ep-gr metric for the set of enhanced queries and $k = 200$. At the first level, the results are quite good but there is a rapid decrease in quality and it is very near from zero after 0.1. The same behaviour is seen in the results for the focused task either with overlap On (Fig. 4) or Off (Fig. 5). Moreover these figures display the result measure at 5 documents. When looking further in the list of results, our method compare less and less favourably to other methods. We develop some explanations about this remark in the following.

Our queries are conjunctions of the terms that a (part of a) document has to contain to have a non zero score. Moreover the occurences of the different terms must be close one to each other. As a result this method is highly selective. Table 1 displays the distribution of queries in regards to the length of the result list built by our method. This highlights the fact that our list of results are quite short. The length of the lists are longer as k increases — which is trivial — and longer also with the enhanced queries as most of them relax the constraints on

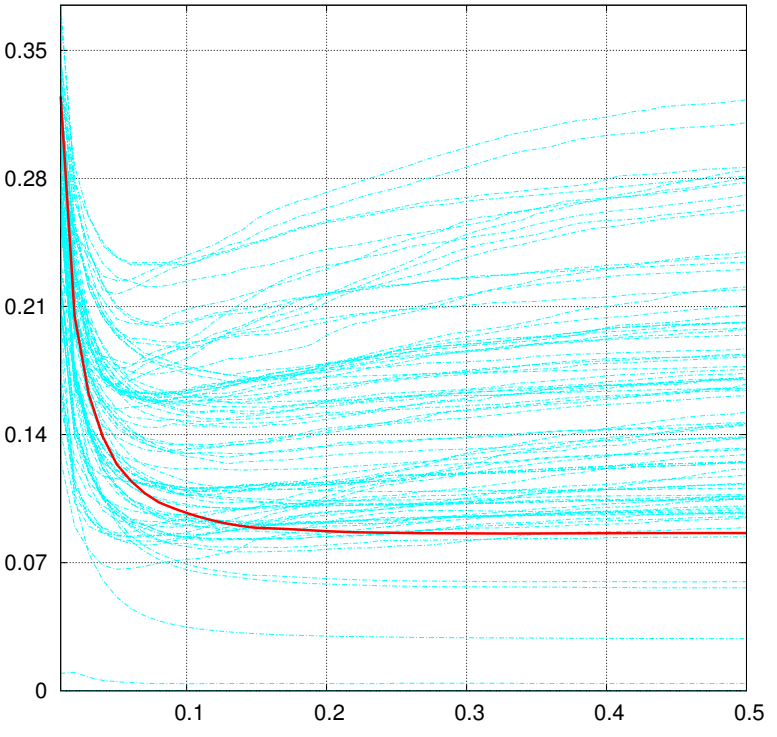


Fig. 4. INEX 2006: Results' Summary, Metric: nxCG[5], Quantization: gen, Task: focused, Overlap=On, Run: title_Q_Prox200NF02

the retrieved documents. Also it can be seen that the very large majority of our result lists are much shorter than the limit of 1500 imposed by INEX.

It seems though that we obtain a quite good precision. Our summarized results were also disavantaged because of our simplification of the structure. In the thorough task we only returned elements tagged as sections and did not add higher level elements when one section was retrieved.

Finally our propagation mechanism was disturbed by strange documents. For instance document number 192509 contains many text in lists between the `<title>` and `</title>` tags. This results with our structure simplification with a very long title whose terms are propagated to the whole title and its associated section. If it happens that all the query terms appear in this erroneous title the section reach a maximal score of 1. Besides this "document bug", it highlights a drawback of our method.

7 Conclusion

We presented in this paper the ideas used for our participation to the INEX 2006 campaign in the ad'hoc retrieval task. Our method is based on the proximity of

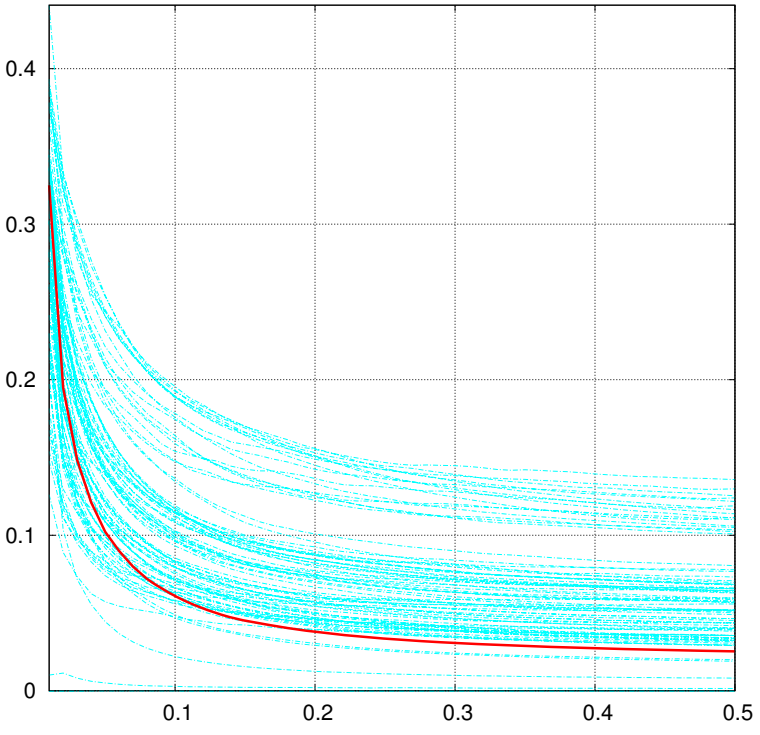


Fig. 5. INEX 2006: Results' Summary, Metric: nxCG[5], Quantization: gen, Task: focused, Overlap=Off, Run: title_Q_Prox200NF02

Table 1. Distribution of queries as a function of their result list length

	No result	1 to 10	10 to 100	100 to 1000	more than 1000
Thorough k=50 short	13	21	50	37	4
Thorough k=200 short	7	20	46	44	8
Thorough k=200 enhanced		7	53	55	10
Foc./Best k=50 short	13	40	48	22	2
Foc./Best k=200 short	7	30	60	26	2
Foc./Best k=200 enhanced		15	75	32	3

the keywords in the document and the propagation of the title words to the whole associated section. We obtained quite good precision results. The summarization done by the official measure was at our disadvantage because our result lists were very short and because we simplified the structure so we did not return every kind of elements. We also found that this method needs query expansion mechanism. Further works are to be done to study the best setting for the parameter k which controls the range of influence of a term in the text. Also some work has to be done to improve the propagation mechanism of title words which is very crude at the time.

References

1. Clarke, C.L.A., Cormack, G.V., Burkowski, F.J.: Shortest substring ranking (multitext experiments for TREC-4). [5]
2. Hawking, D., Thistlewaite, P.: Proximity operators - so near and yet so far. [5]
3. Rasolofo, Y., Savoy, J.: Term proximity scoring for keyword-based retrieval systems. In: Sebastiani, F. (ed.) ECIR 2003. LNCS, vol. 2633, pp. 207–218. Springer, Heidelberg (2003)
4. Beigbeder, M., Mercier, A.: An information retrieval model using the fuzzy proximity degree of term occurences. In: Haddad, H., Liebrock, L.M., Omicini, A., Wainwright, R.L. (eds.) SAC, pp. 1018–1022. ACM Press, New York (2005)
5. Harman, D.K. (ed.): In: Harman, D.K., (ed.) The Fourth Text REtrieval Conference (TREC-4), Department of Commerce, National Institute of Standards and Technology (1995)

Set of Enhanced Queries

```
289 Napoleon & (Polish | Poland)
290 genetic & (algorithm | algorithms) &
(history | algorithm | function | (data &
(structure | structures)) | implementation)
291 (Olympian | Olympie) & (god | goddess)
292 Italian & Flemish & painting & Renaissance
293 wifi & security & encryption
294 user & interface & (design | usability)
295 software & ((intellectual & property) |
(patent & license))
296 (Borussia & Dortmund)
297 cool & jazz & West & coast & (musician |
musicians)
298 ((George & Orwell) | (Eric & Arthur &
Blair))
299 software & development & process &
iterative
300 Airbus & A380 & (order | orders | ordered
| (air & (compagny | compagnies)))
301 (Vector & Space & Model) | (Latent &
Semantic & Indexing) | (salton & smart) |
(extended & Boolean & model)
302 (web & services & security)
303 fractal & (applications | application)
304 (allergy | allergies) & (treatment |
treatments)
305 revision & control & (system | systems)
306 (genre & (theory | classification))
| (structuralist & approach) | ((Plato |
Aristotle) & (form | forms))
307 (Islam | Islamic) & faith
308 wedding & (traditions | customs) & (day &
ceremony)
309 Ken & Doherty & (finals | final)
310 Novikov
311 global & warming
312 recessive & (gene | genes) & ((hereditary
& (disease | diseases)) | (genetic &
(disorder | disorders)))
313 kant & philosophy
314 ((food & additive) | (E & number)) &
(toxin | carcinogen)
315 (spider | spiders) & hunting & (insect |
insects)
316 gymnastics & sport & ((discipline |
disciplines) | (movement | movements))
317 (tourism | visit) & paris & ((museum |
museums) | (cathedral | cathedrals))
318 atlantic & ocean & islands & (slave |
slaves)
319 ((northern & lights) | (polar & lights) |
(aurora & borealis))
320 paris & (Gare & de & Lyon) & (Gare & du &
Nord)
321 Antoni & Gaudi & Barcelona
322 (castles | castle | kasteel) & netherlands
323 ikea & founder
324 composition & planet & rings
325 (Cirque & du & Soleil)
326 Scotland & tourism
327 (clone | clones | cloning) & ((United &
States & of & America) | USA)
328 NBA & European & basketball & player
329 (national & (clothing | dress)) &
(Scottish | Scotland)
330 (nobel & prize) & laureate & physics &
(dutch | Netherlands)
331 (tulips | tulip) & (figure | figures)
332 (NCAA & basketball & tournament) | (march
& madness)
333 (steve & wozniak) & (steve & jobs)
334 (Silk & Road) & China
335 acorn & (eat | eating)
336 (species & monotreme)
337 (security & (algorithms | algorithm))
338 high & blood & pressure & (effect |
effects)
339 (Toy & Story)
340 (Reinforcement & Learning) | (Q &
Learning)
341 microkernel & operating & (system |
systems)
342 (birthday & party) & (nick & cave)
343 Goodkind & (novel | novels)
344 XML & database
345 (Sex & Pistols) & Manchester
```

346 unrealscript
347 (state & machine) & (Moore | Mealy)
348 drinking & water & germany
349 protocol & wireless & security
350 animal & flight
351 (Chinese | China) & wedding & (custom | customs | tradition)
352 faster & than & light & travel
353 (in & place) & (sort | sorting) & (algorithm | algorithms)
354 (novel | novels) & (adaptation | adaptations) & (science & fiction & (film | films))
355 (best & actress) & (academy & (awards | award)) & winner
356 (natural & language & processing) & (information & retrieval)
357 (babylonia | babylonian | assyriology) & culture
358 (information & retrieval) & ((semantic & indexing) | (ontologies | ontology))
359 (shortest & path) & (problem | algorithm | algorithms)
360 solar & energy & (electricity | heating)
361 Europe & after & (second & world & war)
362 (effect | effects) & nuclear & power & plant & accident
363 (Bob & Dylan) & (Eric & Clapton)
364 mushroom & (poisonous | poisoning)
365 Peru & international & investment
366 Fourier & transform & (applications | application)
367 (true & story & films) & ((best & director) | (movie & award))
368 Hymenoptera & Apocrita
369 Pillars & Hercules
370 sport & (offside | (off & side)) & (rule | rules)
371 William & Buckley
372 voodoo & (rituals | ritual)
373 Australia & Echelon & spy & network
374 2004 & Tsunami & (aid | aids)
375 (states | countries) & (nuclear & (proliferation | nonproliferation) & treaty) | npt
376 ((diabetes & mellitus) | (type & 2 & diabetes)) & (symptoms | symptom)
377 (malvasia & grape) & (vinification | wine)
378 indoor & (sports | sport) & ball
379 embargo & Cuba
380 headache & fatigue & nausea & symptoms
381 ubiquitous & computing & (application | applications)
382 Aphrodite
383 Lyon
384 (albert & einstein) & (politics | political)
385 arnold & schwarzenegger & ((co & (starring | star)) | cast | casting)
386 fencing & (weapon | weapons)
387 bridge & types
388 rhinoplasty
389 (cryptography | encryption) & key & (algorithm | algorithms)
390 Insomnia & (cause | causes)
391 (rule | rules | play | playing) & cricket
392 Australian & aboriginals & stolen & generation
393 wireless & (devices | device) & (Health & Hazards)
394 global & warming & (effect | effects)
395 September & 11 & conspiracy & (theory | theories)
396 2004 & Tsunami & Indian & Ocean & Earthquake
397 SUSE & Linux
398 ringo & starr & (musicians | musician)
399 mobile & phone & UMTS & (country | countries)
400 (violent) & revolution & (country | countries)
401 (award | awards) & ((eddie & murphy) | (jim & carrey) | (robin & williams))
402 (countries | country) & (europe | european) & (capital | capitals)
403 color & television & analog & (standard | standards)
404 (french | france) & (singer | singers)
405 The & Old & Man & and & the & Sea
406 architecture & (book | books)
407 (Football & World & Cup) & (Miracle & of & Bern)
408 (electroconvulsive & therapy) & depression
409 (Hybrid & Vehicles) & ((fuel & (efficiency | sources)) | types)
410 (Routers | Router | Switches | Switch) & (computer & (network | networks))
411 (GSM & CDMA) & (standards | standard | coverage | roaming | price | prices)
412 (NT | linux | windows) & (stability | price | prices | security)
413 ((capital & cities) | (capitals)) & Europe & (coordinates | population | latitude | longitude)

Supervised and Semi-supervised Machine Learning Ranking

Jean-Noël Vittaut and Patrick Gallinari

Laboratoire d'Informatique de Paris 6
104, avenue du Président-Kennedy, F-75016 Paris, France
{vittaut,gallinari}@poleia.lip6.fr

Abstract. We present a Semi-supervised Machine Learning based ranking model which can automatically learn its parameters using a training set of a few 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-Thourough 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 [2] [3] [8], passage classification, automatic summarization [1] 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 examples. This is why most real data applications of ranking are based on two-classes problems. Nevertheless, some linear methods has been proposed [8] [1] and under some conditions, fast rates of convergence are achieved with this class of methods [4].

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 first ideas to combine labeled and unlabeled data come from the statistician community at the end of the 70's . Most of these methods use a mixture of gaussian model and try to estimate its parameters by maximizing the joint likelihood of labeled and unlabeled data [10]. There is in general a one to one relation between classes and mixture components [9]. As it is usually impossible

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 213–222, 2007.

to perform this estimation directly, so they use a class of iterative algorithms called EM for Expectation Maximization [6]. The semi-supervised paradigm has also been used in IR [13]. It has showed its efficiency on a class of tasks where there are only a few labeled examples available. This is the case in Structured Information Retrieval (SIR) and more generally in Information Retrieval (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 examples are labeled like in 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. We also show how we learn using both labeled and unlabeled examples. In section 4 we comment the results reached by our model and compare it to a baseline Okapi method adapted for SIR.

2 Ranking Model

We present in this section a general model of ranking which can be adapted to IR or SIR. The idea of the ranking algorithms proposed in the machine learning community is to learn a total order on a set $\mathcal{X}$, which allows to compare any element pair in this set. Given this total order, we are able to order any subset of $\mathcal{X}$ in a ranking list. For instance in IR, $\mathcal{X}$ can be the set of couples (document, query), and the total order is the natural order on the document scores.

As for any machine learning technique, one needs a training set of labeled examples in order to learn how to rank. This training set will consist in ordered pairs of examples. This will provide a partial order on the elements of $\mathcal{X}$. The ranking algorithm will use this information to learn a total order on the elements of $\mathcal{X}$ and after that will allow to rank new elements. For plain IR, the partial ordering may be provided by human assessments on different documents for a given query.

2.1 Notations

Let $\mathcal{X}$ be a set of elements with a partial order $\prec$ defined on it. This means that some of the element pairs in $\mathcal{X}$ may be compared according to the $\prec$ relation. For Structured Information retrieval $\mathcal{X}$ will be the set of couples (doxel, query) for all doxels and queries in the document collection. This set is partially ordered according to the existing relevance judgments for each query.

2.2 Ranking

Let f be a function from $\mathcal{X}$ to the set of real numbers. We can associate a total order $\prec_T$ 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 vector of real numbers:

$$x = (x_1, x_2, ..., x_d).$$

In our case, the features will be local scores computed on different contextual elements of a doxel. In the following, f will be a linear combination of x's features:

$$f_\omega(x) = \sum_{j=1}^{d} \omega_j x_j \tag{2}$$

where $\omega = (\omega_1, \omega_2, ..., \omega_d)$ are the parameters of the combination to be learned.

Ranking loss. f_ω is said to respect $x \prec x'$ if $f_\omega(x) < f_\omega(x')$. In this case, couple (x, x') is said to be well ordered by f_ω. The ranking loss [8] measures how much f_ω respects $\prec$.

By definition, the ranking loss measures the number of mis-ordered couples in $\mathcal{X}^2$:

$$R(\mathcal{X}, \omega) = \sum_{\substack{(x,x') \in \mathcal{X}^2 \\ x \prec x'}} \chi(x, x') \tag{3}$$

where $\chi(x, x') = 1$ if $f_\omega(x) > f_\omega(x')$ and 0 otherwise.

Ranking aims at learning ω for minimizing (3).

Exponential loss. In practice, this expression is not very useful since χ is not differentiable, ranking algorithms use to optimize another loss criterion called the exponential loss:

$$R_e(\mathcal{X}, \omega) = \sum_{\substack{(x,x') \in \mathcal{X}^2 \\ x \prec x'}} e^{f_\omega(x) - f_\omega(x')}. \tag{4}$$

If is straightforward that $R(\mathcal{X}, \omega) \leq R_e(\mathcal{X}, \omega)$. (4) is differentiable and convex, and then can be minimized using standard optimization techniques. Minimizing (4) will allow to minimize $R(\mathcal{X}, \omega)$.

We can compute a gradient descent. The components of the gradient of R_e are:

$$\frac{\partial R_e}{\partial \omega_k}(\mathcal{X}, \omega) = \sum_{\substack{(x,x') \in \mathcal{X}^2 \\ x \prec x'}} (x_k - x'_k) e^{f_\omega(x) - f_\omega(x')}. \tag{5}$$

With no more hypothesis, the computation of (5) is in $O(|\mathcal{X}|^2)$.

3 Application to CO Tasks

3.1 Definitions

Let's denote $\mathcal{D}$ as 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 Combination of Preference Relations

We suppose each element $x \in \mathcal{X}$ can be ordered according to several preference relations $pref_i$. Let denote $\mathcal{T}$ 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:

$$f_\omega(x) = \sum_{t \in \mathcal{T}} 1_{[node\_type(x)=t]} \cdot \omega_t^\emptyset \cdot \left(1 + \sum_i \omega_t^i \cdot pref_i(x)\right)$$

where t is the node type of x and $pref_i$ is a preference relation among doxels based on textual content. In our experiments, we used a SIR adapted Okapi model [11] described in [12]. 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. For each doxel, this Okapi model was computed on its textual content, on its parent textual content and also on the whole document containing it.

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.

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):

$$R_e(\mathcal{X}, \omega) = \sum_{q \in \mathcal{Q}} \left\{ \sum_{\substack{(x,x') \in \mathcal{X}_q \times \mathcal{X}_q \\ x \prec x'}} e^{f_\omega(x)} e^{-f_\omega(x')} \right\}. \tag{6}$$

Assessments. For each subset $\mathcal{X}_q$, we suppose we have information indicating preferences among doxels:

- an information of exhaustivity, which measures how much a doxel answers the totality of an information need
- an information of specificity, which measures how much a doxel answers only the information need

For doxels which are equally exhaustive or specific, there is no preference.

An assessment is a couple (exhaustivity, specificity). Let denote $\mathcal{A}$ the set of assessments and $A(x)$ the assessment of element x. We can define a partition $\mathcal{X}_q = \bigcup_{a \in \mathcal{A}} \mathcal{X}_q^a$, where

$$\mathcal{X}_q^a = \{x \in \mathcal{X}_q / A(x) = a\}.$$

We can rewrite (6):

$$R_e(\mathcal{X}, \omega) = \sum_{q \in \mathcal{Q}} \sum_{a \in \mathcal{A}} \left\{ \left(\sum_{x \in \mathcal{X}_q^a} e^{f_\omega(x)} \right) \left(\sum_{\substack{b \in \mathcal{A} \\ \mathcal{X}_q^b \prec \mathcal{X}_q^a}} \sum_{x \in \mathcal{X}_q^b} e^{-f_\omega(x)} \right) \right\}. \tag{7}$$

where $\mathcal{X}_q^b \prec \mathcal{X}_q^a$ means that the assessments of the elements of $\mathcal{X}_q^a$ are better than those of $\mathcal{X}_q^b$.

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 $K = |\mathcal{X}|$.

3.4 Gradient Descent

Since (7) is convex, we can use a gradient descent technique to minimize it. The components of the gradient has the following form:

$$\begin{aligned} \frac{\partial R_e}{\partial \omega_k}(\mathcal{X}, \omega) = \sum_{q \in \mathcal{Q}} \sum_{a \in \mathcal{A}} \Bigg\{ & \left(\sum_{x \in \mathcal{X}_q^a} x_k e^{f_\omega(x)} \right) \left(\sum_{\substack{b \in \mathcal{A} \\ \mathcal{X}_q^b \prec \mathcal{X}_q^a}} \sum_{x \in \mathcal{X}_q^b} e^{-f_\omega(x)} \right) \\ & + \left(\sum_{x \in \mathcal{X}_q^a} e^{f_\omega(x)} \right) \left(\sum_{\substack{b \in \mathcal{A} \\ \mathcal{X}_q^b \prec \mathcal{X}_q^a}} \sum_{x \in \mathcal{X}_q^b} -x_k e^{-f_\omega(x)} \right) \Bigg\}. \end{aligned} \tag{8}$$

The complexity for computing the gradient is the same ($O(K \cdot |\mathcal{Q}| \cdot |\mathcal{X}|)$) as that of (7).

3.5 Incorporation of Unlabeled Examples

We have described the Ranking model. We defines now the Semi-supervised model which is an extension using unlabeled examples. The natural way to incorporate an unlabeled example y would be to compute all probabilities $P(y \prec x)$ for x in $\mathcal{X}$. But this method would be computationnaly costly because we could not use property 7 to reduce the complexity since y is not in a particular $\mathcal{X}_q^a$.

A better method is to affect y to only one $\mathcal{X}_q^a$, using a certain probability of belonging. We say that a example belongs to the group with which it has the maximum probability of indifference:

$$P(y \in \mathcal{X}_q^a) = P(\{y\} \perp \mathcal{X}_q^a) = \prod_{x \in \mathcal{X}_q^a} P(y \perp x) = \prod_{x \in \mathcal{X}_q^a} P(y \prec x) P(x \prec y)$$

where $P(y \perp x)$ is the probability that there is no preference between x and y.

If we use the exponential ranking loss, we will choose the group $\mathcal{X}_q^a$ which minimize the ranking loss:

$$\exp(f_\omega(y)) \sum_{x \in \mathcal{X}_q^a} \exp(-f_\omega(x)) + \exp(f_\omega(-y)) \sum_{x \in \mathcal{X}_q^a} \exp(f_\omega(x)).$$

Note this is not ordinal regression because there is some groups ($\mathcal{X}_q$'s) which are each other indifferent and if we could know a priori if an unlabeled example belongs to one of this group. Even if our model produce a total order on $\mathcal{X}$, there is some comparisons without sense (for instance, we see in section 3.3 that comparing search results from different queries is none sense).

We sum up the semi-supervised model in the following algorithm:

Algorithm 1

1. *Minimize the ranking loss on labeled examples.*
2. *Repeat until convergence:*
   3. *Affect each unlabeled example to a group* $\mathcal{X}_q^a$ *according to the minimum ranking loss:*
      $\exp(f_\omega(y)) \sum_{x \in \mathcal{X}_q^a} \exp(-f_\omega(x)) + \exp(f_\omega(-y)) \sum_{x \in \mathcal{X}_q^a} \exp(f_\omega(x))$.
   4. *Minimize the ranking loss on labeled and unlabeled examples.*

4 Experiments

4.1 Learning Base

The Wikipedia collection [7] has been used with a small set of three queries. Then we made 300 assessments for these three queries to be used as a learning base. We collected these assessments using the following method :

1. the system is initialized with equal preferences
   between node types and using only one
   preference relation on doxels' content

2. Repeat several times:
   3. compute the results
   4. re-order 25 not assessed results
   5. learn with the new partial order provided by user assessments

4.2 Filtering

In CO-Focussed task, overlapping doxels were not allowed. In order to suppress all overlapping elements from the lists computed by the ranking algorithm, we used a strategy which consists in removing all elements which are overlapping with an element ranked higher in the list.

As for Okapi model, we used the same strategy exept that biggest doxels like articles or bdy's were not allowed in the final ranking list to reach better performance.

4.3 Results

We comment here the results obtained with the nxCG official metric with generalized quantization which is more related to the ranking loss criterion and the different levels of assessment we have used in our model.

CO-Focussed. We have plotted in figures 1 and 2 the evaluation of the lists produced by the ranking algorithms and by the modified Okapi for CO-Focussed task. We evaluate the model taking overlap into account (overlap on) or not (overlap off). We can see that the ranking algorithms perform better than Okapi, but semi-supervised model has not been able to increase the performance of the ranking algorithm.

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 begining of the list comparing to other INEX participan approaches.

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

CO-Thorough. Figure 3 show the evaluation of the lists produced by the ranking algorithm and modified Okapi where overlap was not removed. We can see that the ranking algorithms performs clearly better than Okapi but the semi-supervised model do not perform better than the supervised model.

Table 3 shows that the ranking models are always better than their baseline Okapi model, and that is quite good to retrieve the most informative doxels in

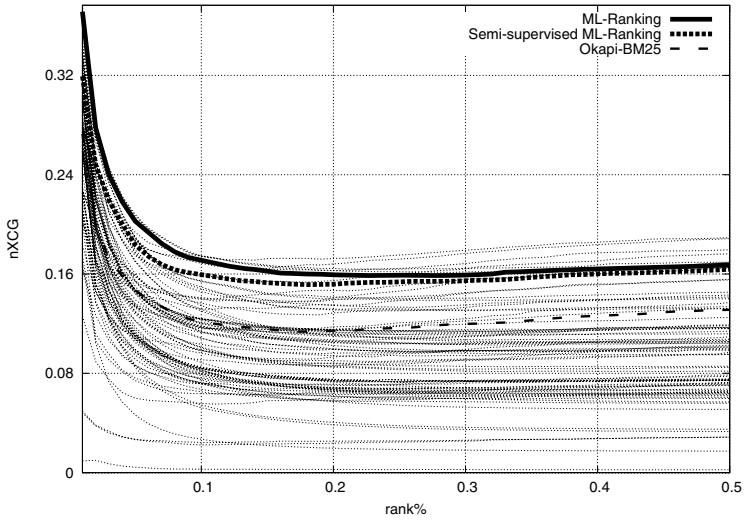


Fig. 1. Performance of ranking and Okapi models for CO-Focussed task evaluated with the cumulated gain based metric ncXG with overlap on

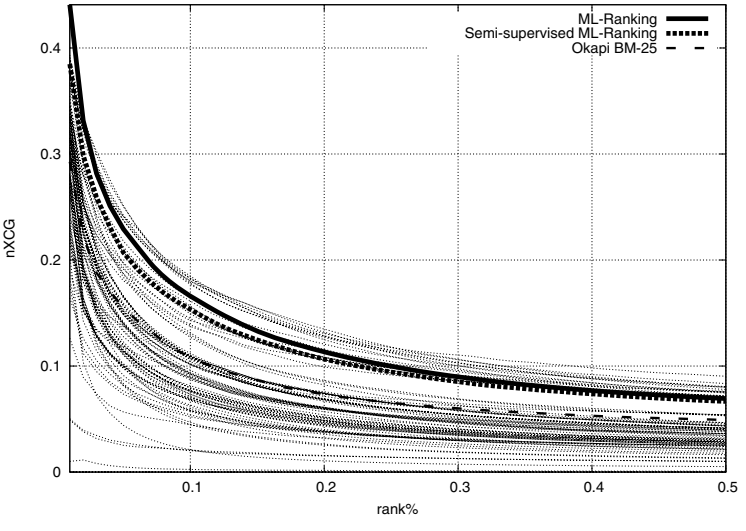


Fig. 2. Performance of ranking and Okapi models for CO-Focussed task evaluated with the cumulated gain based metric ncXG with overlap off

the begining of the list. This can be explained by the expression of the ranking loss which highly penalize an irrelevant doxel when it is located in the begining of the list.

For all experiments, the ranking algorithm has been able to increase the performance of the baseline Okapi. Ranking methods thus appear as a promising

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

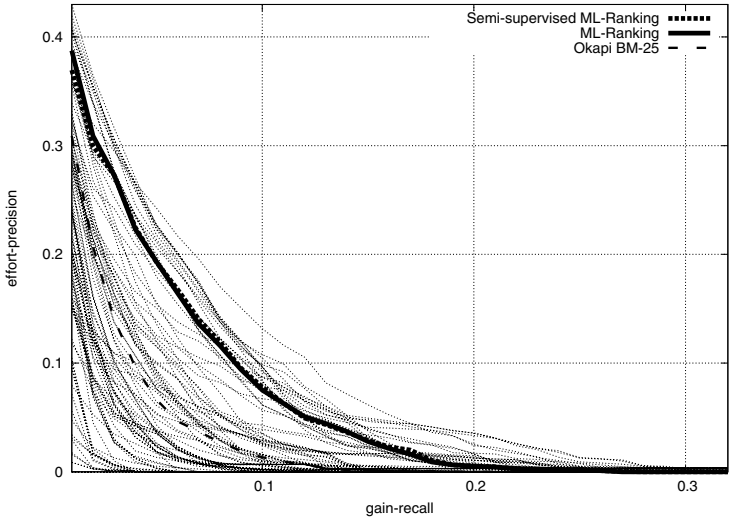


Fig. 3. Performance of ranking and Okapi models for CO-Thorough task evaluated with the cumulated gain based metric ncXG

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

direction for improving SIR search engine performance. It remains to perform tests with additional features (for example the scores of additional IR systems).

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).

Eventually, there is also work in progress to make unlabeled data useful for Ranking algorithms.

References

1. Amini, M.R., Usunier, N., Gallinari, P.: Automatic text summarization based on word-clusters and ranking algorithms. In: ECIR pp. 142–156 (2005)
2. 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, vol. 10, The MIT Press, Cambridge, MA (1998)
3. Bartell, B.T., Cottrell, G.W., Belew, R.K.: Automatic combination of multiple ranked retrieval systems. In: Research and Development in Information Retrieval, pp. 173–181 (1994)
4. Clémençon, S., Lugosi, G., Vayatis, N.: Ranking and scoring using empirical risk minimization. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS (LNAI), vol. 3559, pp. 1–15. Springer, Heidelberg (2005)
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, ACM Press, New York (2005)
6. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via EM algorithm. Journal of the Royal Statistical Society B(39), 1–38 (1977)
7. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus SIGIR Forum (2006)
8. 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)
9. Miller, D., Uyar, H.: A Mixture of Experts classifier with learning based on both labeled and unlabeled data. Advances in Neural Information Processing Systems 9, 571–577 (1996)
10. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text Classification from Labeled and Unlabeled Documents using EM. In: Proceedings of National Conference on Artificial Intel-ligence (1998)
11. Robertson, S.E., Walker, S., Hancock-Beaulieu, M., Gull, A., Lau, M.: Okapi at TREC. In: Text REtrieval Conference, pp. 21–30 (1992)
12. 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)
13. Vittaut, J.N., Amini, M.R., Gallinari, P.: Learning Classification with Both Labeled and Unlabeled Data. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) ECML 2002. LNCS (LNAI), vol. 2430, Springer, Heidelberg (2002)

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 particular, automatic detection of titles – 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 them 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 last year.)

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-duisburg.de/

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 223–232, 2007.

of the overlapping results are more suitable for 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 [2,4]; we repeat the description here for completeness. We use the Apache Lucene information retrieval engine[2] 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.

⟨sec⟩Hello, ⟨b⟩world!⟨/b⟩
How ⟨i⟩are⟨/i⟩ you?
⟨/sec⟩

XPath	Indexed contents
/sec[1]	Hello, world! How are you?
/sec[1]/b[1]	world!
/sec[1]/i[1]	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 Deriving Non-thorough Results

We base our results for the Focused, All in Context, and Best in Context tasks on our thorough results; this is simply another post-processing step added to the end of step 3. The approaches are rather naïve:

[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

Focused: We repeatedly add the element with the highest score from the result tree to the results and remove all its descendants and ancestors.

All in Context: We take our Focused results and group them by document; the documents are then sorted according to the highest-scoring element in each document.

Best in Context: We simply choose the element with the highest result.

The method for getting Focused results proved to be reasonably effective last year, so we used it again. The "in Context" methods were last-minute additions that do little more than ensure that the task requirements are fulfilled; we did not expect them to be very successful.

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

- Query terms with the "`-`" qualifier are discarded (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//atl`, we prefer all `atl` 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 XML retrieval engine. The postprocessing steps we present in this section aim at making use of the structure of the documents to improve retrieval quality [3].

Although they are not viable retrieval results on their own, elements with little text can still provide valuable retrieval hints: For example, the title of a section is typically less than twenty words long and not helpful to the searcher, but a well-chosen title gives a strong hint about the contents of the section.

One way to make use of this is to manually specify which elements types are titles, but this would obviously depend on the schema and require manual inspection of the schema. 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. Figure 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 parent element (in figure 3, the root element `/sec[1]`) is the reference for the scale of the horizontal axis.

⟨sec⟩
Hello, ⟨b⟩world!⟨/b⟩
How ⟨i⟩are⟨/i⟩ you?
⟨/sec⟩

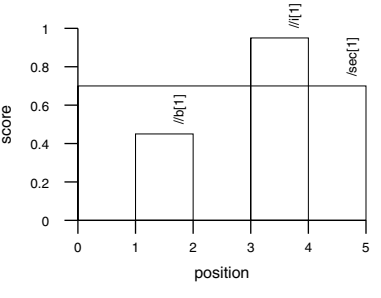


Fig. 3. XML text and corresponding context diagram. The horizontal axis denotes the positions and lengths of the text fragments, and the vertical axis shows the score (in this case random numbers).

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 figure 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 5,000 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.

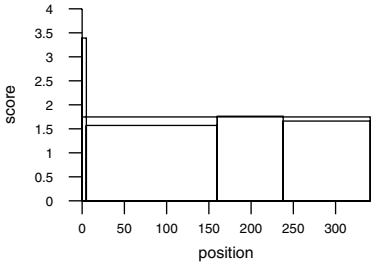


Fig. 4. Example context graph for the title pattern. The short peak at the left is the section title.

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.

Once we have detected the presence of a match in a title element, we need to modify the scores of the involved elements. For a match in a title, an appropriate action is to increase the parent's score (because the match indicates that the enclosing section is highly relevant) and decrease the first child's score (because the title itself contains too little information to be of any use).

Last year, we used several other patterns in addition to the title pattern, but it turned out that only the title pattern had a positive effect on retrieval quality. Because of this, our submitted runs only make use of the title pattern.

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.

One simple space-saving method is normalization of tag names: The vocabulary of tag names is limited, so the search engine assigns a unique number to each one. The numbers assigned depends on the order the tag names are encountered during indexing; it would save even more space to assign lower numbers to the most frequent tags, but this would require two passes of parsing the document collection. In any event, all tag names can be encoded in three or fewer bytes, which is shorter than the mean length of theirs texts.

The structure of the original documents is mirrored using a command–argument structure; it would be wasteful to use a whole byte for each command, since there are only three basic commands (start/end element, text), so the search engine uses the condensed format as shown in Fig. 5. Each command is encoded in an 8-bit byte which consists of one or more *command bits*, and any remaining bits are used as *value bits*.

7	6	5	4	3	2	1	0	bit number
1	0	tag ID						start element
1	1	text length						end element (preceded by text)
0	text length							short text
0	1	1	1	1	1	1	1	long text; followed by variable-length integer

Fig. 5. Basic structure of document digest commands

The following commands are necessary for our purposes:

- Start tag $\#n$.
- End tag it.
- Text fragment of length n (measured in tokens).

The end tag needs no argument, always applies to the last unclosed tag, so we can use the value bits of *end tag* commands for other purposes. In many cases,

closing tags are preceded by text (in the Wikipedia collection, this is the case for 70 % of all closing tags), so we can use the unneeded six lower bits for storing the length of this text.

For the length of long texts and tag IDs, we use a variable-length encoding which enables us to store small numbers in a small number of bytes without placing a fixed limit on the maximum number that can be stored.

For the start tag command, the value bits encode the start of the tag number in variable-length encoding: If the tag ID is less than 64, bit 5 is set to 0, and the tag ID is stored in the lower five bits; otherwise, bit 5 is set to 1, the tag ID modulo 64 is stored in the lower five bits, and the tag ID divided by 64 is stored in the following bytes in the regular variable-length integer encoding.

For the end tag command, the value bits contain the length of the text fragment preceding the closing tag (if this text fragment is too long to be stored in six bits, a text command is inserted before the end tag and the value bits of end tag are set to zero).

Most text fragments are short, so it is reasonable to accept a small penalty for longer texts in order to store short texts efficiently. Thus, for text lengths less than 127 ($111\,1111_2$), the text's length is stored in the value bits of the command byte, otherwise all value bits are set, and the text length is encoded in the following variable-length integer. The other option would be to use a regular variable-length encoding for the numbers (that is, use bit 6 as a marker bit), but this would have increased the number of bytes used for texts that are between 64 and 126 tokens long.

Table 1. Space statistics for digests

Number of bytes needed for ...	Mean	Median	Encoded in one byte
... start tags	1.1735	1	83 %
... text fragments*	1.0001	1	100 %

* counting text embedded in end tags as one byte.

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

10.000000 ⟨sec⟩
0.0000001 one text token
10.000001 ⟨b⟩
11.000001 one text token, ⟨/b⟩
0.0000001 one text token
10.100000 ⟨i⟩
0.0100001
11.000001 one text token, ⟨/i⟩
11.000001 one text token, ⟨/sec⟩

Tag ID	Name
0	sec
1	b
65	i

Fig. 6. Example digest for the document from Fig. 1

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.

Originally, the results from the first retrieval step contain elements from 58,000 documents on average; after discarding the low-scoring documents, 5,600 documents remain (this number is higher than 5,000 because we also keep all documents with exactly the same score as the 5,000th document). The mean time needed to process a topic drops from 13.7 seconds to 4.2 seconds, and as we will see in the following section, retrieval quality is not compromised.

5 Evaluation and Discussion

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 results[3] are incomplete, three tentative conclusions can be drawn at this point (see Table 2 and Fig. 7 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.

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

Table 2. INEX 2006 results. CO.Thorough had 106 submissions, CO.Focused (with overlap=on) had 85, some of them with overlaps.

	CO.Thorough		CO.Focused	
Run	MAep	Rank	nxCG@25	Rank
Fuzzy patterns	0.0336	31	0.1685	51
Crisp patterns	0.0291	41	0.1719	49
Crisp patterns (best 5,000 documents kept)	0.0293	40	0.1724	48

Although it is unreasonable to expect the same level of performance on a different document collection and topic set, the differences to last year's results are too large to be ignored: At INEX 2005, our search engine consistently ranked among the top five for cut-off points up to 50. It is unclear why the results are so much worse this year; the characteristics of the Wikipedia collection are not much different from that of the IEEE collection.

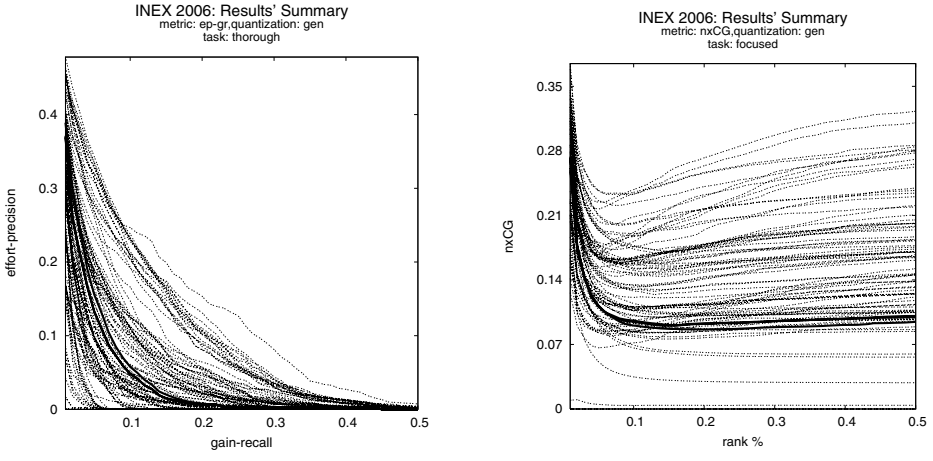


Fig. 7. Metrics graphs. Our runs are drawn with thicker lines.

As expected, our naïve approaches to the "in context" tasks were not effective. For All in Context, our best run is 35th of 56 submissions (mean average generalized precision/recall), and roughly 35th to 40th out of 77 for Best in Context for the various parameter values of the evaluation measure.

6 Summary

We have described how we adapted the retrieval system we used in 2005. The most significant change concerns handling the larger document collection well

(the old system did not scale). Our basic retrieval approach is mostly unchanged, but the quality of the retrieval results has dropped considerably, which we currently cannot explain.

References

1. Denoyer, L., Gallinari, P.: Ludovic Denoyer and Patrick Gallinari. SIGIR Forum 40(1), 64–69 (2006)
2. Dopichaj, P.: The University of Kaiserslautern at INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, Springer, Heidelberg (2006)
3. Philipp Dopichaj. Improving content-oriented XML retrieval by applying structural patterns. In: ICEIS, – Proceedings of the Ninth International Conference on Enterprise Information Systems, 2007,To appear (2007)
4. Eger, B.: Entwurf und Implementierung einer XML-Volltext-Suchmaschine. Master's thesis, University of Kaiserslautern (2005)
5. Kazai, G., Lalmas, M.: Notes on what to measure in INEX. In: Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology (2005)
6. Michalewicz, Z., Fogel, D.B.: How to Solve It: Modern Heuristics, 2nd edn., chapter 13, pp. 367–388. Springer, Heidelberg (2004)
7. Trotman, A., Sigurbjörnsson, B.: Narrowed extended XPath I (NEXI). In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, Springer, Heidelberg (2005)

TopX – AdHoc Track and Feedback Task

Martin Theobald, Andreas Broschart, Ralf Schenkel, Silvana Solomon, and Gerhard Weikum

Max-Planck-Institut für Informatik
Saarbrücken, Germany
{mtb,abrosch,schenkel,solomon,weikum}@mpi-inf.mpg.de
http://www.mpi-inf.mpg.de/departments/d5/

Abstract. This paper describes the setup and results of the Max-Planck-Institut für Informatik's contributions for the INEX 2006 AdHoc Track and Feedback task. The runs were produced with the TopX system, which is a top-k retrieval engine for text and XML data that uses a combination of BM25-based content and structural scores.

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:

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 233–242, 2007.

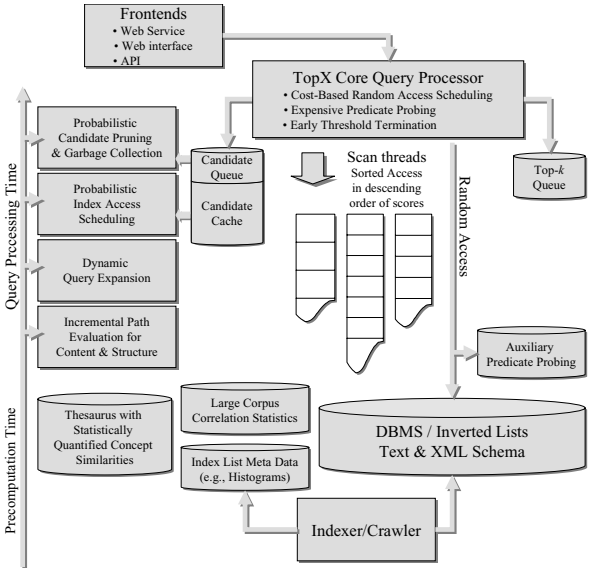


Fig. 1. TopX architecture

- 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 `*[about(., t)]` is then computed by the following BM25-inspired formula:

$$score(e, \texttt{*[about(., t)]}) = \frac{(k_1+1)\ ftf(t,e)}{K + ftf(t,n)} \cdot \log\left(\frac{N_A - ef_A(t) + 0.5}{ef_A(t) + 0.5}\right) \quad (1)$$

$$\text{with } K = k_1 \left((1-b) + b \frac{\sum_{s \in \ full\ content\ of\ e} ftf(s,e)}{avg\{\sum_{s'} ftf(s',e') \mid e' \ with\ tag\ A\}}\right)$$

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.:

$$score(e, \texttt{*[about(., } t_1 \ldots t_m \texttt{)]}) = \sum_{i=1}^{m} score(e, \texttt{*[about(., } t_i \texttt{)]}) \quad (2)$$

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 satisfies 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 specifitiy. 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 bidimensional 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.

Table 1. Results for the Thorough Task

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

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.

Table 2. Results for the Focused Task with the nxCG metric at different cutoffs (ranks are in parentheses), with overlap=on

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 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-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 4. Results for the BestInContext Task with the EPRUM-BEP-Exh-BEPDistance 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)

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 Official Results

Due to time constraints, we submitted only a single official run where we used only content-based feedback from the top-20 elements of our baseline run

`TOPX_CO_Thorough_all`. In the official evaluation, this setting performed well, with an absolute improvement of 0.0064 in MAep over the baseline run for the *Generalised* quantisation, which corresponds to a relative improvement of about 25%. Both the t-test and the Wilcoxon signed-rank test show that the improvement is significant (with $p \approx 0$).

5.3 Additional Results

Besides our official run that used only content-based feedback, we tried different combinations of candidate classes. We measured only MAP and precision at different cutoffs. 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. We tested the significance of our results with the t-test and the Wilcoxon signed-rank test [6].

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 ($p < 0.01$).

evaluation	baseline	C	D	C+D	A	AD	A+AD
plain	0.0188	***0.0364***	***0.0328***	***0.0344***	0.0187	0.0164	***0.0228***
freezeTop	0.0188	***0.0284***	***0.0248***	***0.0256***	**0.0189**	0.0181	***0.0216***
resColl-result	0.0108	***0.0264***	***0.0212***	***0.0218***	0.0106	0.0106	**0.0171**
resColl-anc	0.0101	***0.0246***	***0.0194***	***0.0201***	0.0102	0.0102	**0.0169**
resColl-desc	0.0049	***0.0087***	0.0078	***0.0081***	0.0048	0.0045	0.0056
resColl-doc	0.0041	***0.0077***	**0.0067**	***0.0072***	0.0040	0.0038	0.0048
resColl-path	0.0046	***0.0085***	0.0072	***0.0079***	0.0044	0.0043	0.0056

Table 5 shows the results for our additional runs. 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 unidimensional 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.

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. Bast, H., Majumdar, D., Theobald, M., Schenkel, R., Weikum, G.: IO-Top-k: Index-optimized top-k query processing. In: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006), pp. 475–486 (2006)
2. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
3. Pehcevski, J., Thom, J.A.: HiXEval: Highlighting XML retrieval evaluation. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 43–57. Springer, Heidelberg (2006)
4. Robertson, S.E.: On term selection for query expansion. J. Doc. 46, 359–364 (1990)
5. Robertson, S.E., Sparck-Jones, K.: Relevance weighting of search terms. J. Am. Soc. Inform. Sci. 27, 129–146 (1976)
6. Savoy, J.: Statistical inference in retrieval effectiveness evaluation. Inform. Process. Manag. 33(4), 495–512 (1997)
7. Schenkel, R., Theobald, M.: Relevance feedback for structural query expansion. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 344–357. Springer, Heidelberg (2006)
8. Schenkel, R., Theobald, M.: Structural feedback for keyword-based XML retrieval. In: Proceedings of the 28th European Conference on IR Research (ECIR 2006), pp. 326–337 (2006)
9. Theobald, M., Schenkel, R., Weikum, G.: Efficient and self-tuning incremental query expansion for top-k query processing. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 242–249 (2005)
10. Theobald, M., Schenkel, R., Weikum, G.: An efficient and versatile query engine for TopX search. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005), pp. 625–636 (2005)
11. Theobald, M., Schenkel, R., Weikum, G.: TopX & XXL INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 282–295. Springer, Heidelberg (2006)
12. Theobald, M., Weikum, G., Schenkel, R.: Top-k query evaluation with probabilistic guarantees. In: Proceedings of the 30th International Conference on Very Large Data Bases (VLDB 2004), pp. 648–659 (2004)

Tuning and Evolving Retrieval Engine by Training on Previous INEX Testbeds

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 changes between INEX'2005 and INEX'2006 notably the corpus change. 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 results of the official experiments carried out at INEX'2006 and additional results.

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 information retrieval systems based on probabilistic models [8][11][12][15], information retrieval systems based on vector space models [2][4][6][9] and systems based on databases [3][10].

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][9], probabilistic methods such as [11][15] and database systems such as [10] 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][13] 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 official and additional results. Section 5 concludes this paper.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 243–252, 2007.

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, a new task has been defined i.e. Best in Context. In addition, runs using queries on content only and runs using queries on content and structure were merged for evaluation. This new processing offers a mean to evaluate whether treating structural hints in our method improves results or not,
- 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 INEX.

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 [6][7][14]. 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 the 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(bdy, 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 3

$gc(Q,E)$ 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 definition 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 gc(Q,E)=0.0 for elements that do not match the structural constraints, the structural constraints on result are strictly taken into account. When gc(Q,E)=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:

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

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 were 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 $prf(t_i,Q)$ 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 α=0.1 and φ=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 α=0.6 and φ=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 α=0.0 and φ=3000. This combination obtained good results on INEX'2005 testbeds for the subtask Focused notably for strict quantisation. This combination does not include score aggregation and considers strongly global query presence.

4.2 Official and Additional Results

The official results corresponding to different configurations of our method tested for the different are detailed in the following tables. Additional results are included for the Thorough task in Table 1.

Table 1. Results for the subtask Thorough

Task	Thorough							
Metric: ep/gr	*Quantisation: generalised, overlap=off*							
	Official results				*Additional results*			
Run	V2006Cg35CH01x1tho		V2006CASCH01x1tho		*Cg35CH06x50*		*Cg35CH05x1000*	
	Maep	Rank	Maep	Rank	*Maep*	*Rank*	*Maep*	*Rank*
	0.0147	47/106	0.0161	40/106	*0.0232*	*(23)*	*0.0259*	*(20)*

A first observation is that official results are slightly over average. The configuration results from experiments on strict quantisation with INEX'2005 testbeds and gave weaker results for generalised quantisation. Without official results for strict quantisation final conclusions cannot be established. The INEX'2006 official results tend to show same behaviours of configurations for generalised quantisation. The additional results notably for the run labelled *Cg35CH06x50*, which gave good results on INEX'2005 testbed for the same task and quantisation, lead to better evaluations with INEX'2006 data.

Another observation is that structural conditions seem to improve the results since the run using castitle parts of queries obtain higher average precision than the run with same configuration using titles of queries.

A first observation is that results are average. The results have the same behaviour as with INEX'2005 testbeds for the same task and quantisation. 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 the Focused task than to return intermediate nodes.

Table 2. Results for the subtask Focused

Task	Focused					
Metric: nxCG	*Quantisation: generalised*					
Run	V2006CH0x3000foc		V2006CASCH0x3000foc		V2006CH06x50foc	
overlap=on	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
overlap=off	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

Another observation is that structural conditions do not improve results which can be explained by the fact that only a restricted set of XML elements are treated when score aggregation is not used. Having results for strict quantisation could lead to further conclusions notably with regard to the run V2006CH0x3000foc whose configuration gave good results with the INEX'2005 data.

Table 3. Results for the subtask BestInContext

Task	BestInContext					
Metric: BEP-D						
Run	V2006CH01xp1bic		V2006CH06xp50bic		V2006CASCH06xp50bic	
	BEPD	rank	BEPD	rank	BEPD	rank
At 0.01	0.1175	30/77	0.1088	35/77	0.0015	75/77
At 0.1	0.1826	30/77	0.1702	38/77	0.0039	75/77
At 1.0	0.2958	30/77	0.2907	34/77	0.0079	75/77
At 10.0	0.4729	34/77	0.4832	30/77	0.0122	75/77
At 100.0	0.6430	39/77	0.6711	30/77	0.0185	75/77

A first observation is that configuring our method with weak score aggregation and with no query presence factor or configuring our method with score aggregation and with query presence factor seems to lead to close evaluations. Further investigations at the query level have to be carried out to compare the results of these two runs to determine the proportion of common ranked elements and eventually to consider a possible fusion strategy.

A second observation is related to the weak results given using the castitle part of the queries (i.e. run V2006CASCH06xp50bic). Introducing structural hints seems to move

Table 4. Results for the subtask AllInContext

Task	AllInContext					
Metric: generalized Precision/Recall						
Run	V2006CH01x1ric		V2006CH06x50ric		V2006CASCH06x50ric	
	MAgp	rank	MAgp	rank	MAgp	rank
	0.0441	50/56	0.0835	39/56	0.0887	37/56

away the selected elements from the relevant ones. A too high value fixed for the coefficients associated to structural treatment could be a reason.

The results for this subtask are weaker than for the other subtasks. Additional evaluations with respect to article level and element level seem to indicate that the difficulty seems to exist in the element retrieval rather than in the article retrieval. Further analysis has to be carried out to find explanations and to consider evolutions of our method.

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,
- A new task called 'Best in Context' has been defined 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 results lead to mixed conclusions. The behaviour of the different configurations of our method using INEX'2006 data is similar to the behaviour of the same configurations when learning on INEX'2005 data. However, additional results show that other configurations of our method lead to better results on INEX'2006 data. A wider range of domains included in the INEX'2006 could explain this difference. Furthermore, the results show that a unique configuration of our method does not fit all the subtasks defined. A given configuration seems to be more adapted to a given subtask. This leads to consider a future study to determine how to configure our method to suit a given subtask. A first study [5] has been carried out on INEX'2005 data notably for the Thorough task. This study aims to identify how to configure our method according to different quantisations.

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), Barcelona, 2001, pp. 291–300 (2001)

[2] Crouch, C.J., Khanna, S., Potnis, P., Doddapaneni, N.: The Dynamic Retrieval of XML ElementsAn Approach to Structured Retrieval Based on the Extended Vector Model. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 268–281. Springer, Heidelberg (2006)

[3] Fuhr, N., Großjohann, K.: XIRQL: An XML query language based on information retrieval concepts. ACM Transactions on Information Systems (TOIS) 22(2), 313–356 (2004)

[4] Geva, S.: GPX – Gardens Point XML IR at INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 240–253. Springer, Heidelberg (2006)

[5] Hubert, G., Mothe, J., Englmeier, K.: Tuning Search Engine to Fit XML Retrieval Scenario. 3rd International Conference on WEB Information Systems and Technologies (WEBIST 2007), Barcelona, pp. 228–233 (2007)

[6] Hubert, G.: XML Retrieval Based on Direct Contribution of Query Components. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 172–186. Springer, Heidelberg (2006)

[7] Hubert, G.: A voting method for XML retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 183–196. Springer, Heidelberg (2005)

[8] Larson, R.R.: Probabilistic Retrieval, Component Fusion and Blind Feedback for XML Retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 225–239. Springer, Heidelberg (2006)

[9] Mass, Y., Mandelbrod, M.: Using the INEX Environment as a Test Bed for Various User Models for XML Retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 187–195. Springer, Heidelberg (2006)

[10] 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. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 72–87. Springer, Heidelberg (2006)

[11] Ogilvie, P., Callan, J.: Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 211–224. Springer, Heidelberg (2006)

[12] Sigurbjörnsson, B., Kamps, J., de Rijke, M.: The Effect of Structured Queries and Selective Indexing on XML Retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 104–118. Springer, Heidelberg (2006)

[13] Sauvagnat, K., Hlaoua, L., Boughanem, M.: XFIRM at INEX 2005: Ad-Hoc and Relevance Feedback Tracks. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 88–103. Springer, Heidelberg (2006)

[14] Sauvagnat, K., Hubert, G., Boughanem, M., Mothe, J., IRIT,: at INEX, 2nd INitiative for the Evaluation of XML Retrieval, Dagstuhl, 2003, pp. 142–148 (2003)

[15] Vittaut, J.-N., Piwowarski, B., Gallinari, P.: An Algebra for Structured Queries in Bayesian Networks. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 100–112. Springer, Heidelberg (2005)

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

Abstract. Our submission to the INEX 2006 Ad-hoc retrieval track is described. We study how to utilize the Wikipedia structure (XML documents with hyperlinks) by combining XML and Web retrieval. In particular, we experiment with different combinations of language models and the HITS algorithm. An important feature of our techniques is a filtering phase that identifies the relevant part of the corpus, prior to the processing of the actual XML elements. We analyze the effect of the above techniques based on the results of our runs in INEX 2006.

1 Introduction

The Ad-hoc track of INEX 2006 consists of four *tasks,* namely, *Thorough, Focused, All-in-Context* and *Best-in-Context.* This paper describes our participation in this track. In particular, we describe the different methods and techniques we used and the results of our runs.

In all of the Ad-hoc tasks, the goal is to appropriately estimate the relevance of *elements* in *XML documents.* One may consider this goal as being equivalent to estimating whole documents by considering each element as a stand-alone source. We, however, take a different approach: First, we apply some preliminary ranking to the documents and *filter out* those with low relevancy. We then rank each of the elements of the documents in the remaining corpus. Our estimation, hence, consists of two main components: a *document filter* and an *element ranker.* Furthermore, each of these components is itself associated with some method of estimation. Our runs and experimentations consist of several combinations of filters and rankers. In addition, the configuration of each run contains parameter settings and some additional techniques that we describe later in this section.

In our runs, we used two methods of estimation for both document filtering and element ranking. The first method is a linear interpolation of language models [1], namely, the corpus, document and element (in the case of element ranking). The graph nature of the Wikipedia collection, which contains hyperlinks between documents and elements (in the form of *XPointers* and *XLinks*), led us to believe that combining methods of XML and Web retrieval could be

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

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 253–260, 2007.

a promising approach. Thus, the second method is based on applying the HITS algorithm [2] and is combined with the language-model approach.

We used additional, simple techniques that were highly useful when testing our methods on the results of the Ad-hoc track of INEX 2005. One example is *length cutoff* [3], that is, elimination of very short elements from the results. Another example is the elimination of elements that have relatively high rank, yet contain none of the query term (such cases can occur due to smoothing [4] and the application of HITS).

We obtained several insights on our methods from the results of the Ad-hoc track of INEX 2006. First, our methods generally provide a good estimation of relevancy. In the Thorough and Focused tasks, most of our runs were among the best. In the other two tasks, some of our runs were among the top 10. Second, better results were obtained when using HITS for filtering rather than for element ranking. Finally, the results did not show that either one of the two filtering methods (i.e., language models and HITS) was consistently better than the other one.

2 Document Filters and Element Rankers

The approach that we take for estimating relevancy of XML elements entails two main steps. The first step is to identify a relevant subset of the documents in the corpus, while the second is to rank each of the elements of the documents found in the first step. More particularly, each of our submissions to the Ad-hoc track is produced by the following two components:

1. The *document filter* chooses a set of documents that are relevant to the given query. We call this set the *filtered corpus.*
2. The *element ranker* ranks each of the elements in the documents of the filtered corpus.

Note that this two-step approach is efficient since, typically, it allows us to ignore a huge number of XML elements.

To implement each of the above two components, we used two different ranking methods. Thus, a specific estimation technique in our setting is essentially a combination of two ranking methods: one for filtering out irrelevant documents and the other for ranking elements.

3 Document Filters

The document filters basically apply a preliminary ranking to the documents and choose the top N, where N is a predefined parameter. In all our submissions to the Ad-hoc track, we used $N = 500$. Next, we describe each of the two filters we used.

3.1 Language Model

The first filter, denoted by F_{LM}, is based on statistical language modeling [1] combined with smoothing techniques [4]. More particularly, for a query q, filtering consists of the following three steps.

1. Estimate the likelihood of generating q in each document's language model. This results in a value $P(q|D)$ for each document D.
2. Count the number of terms of q appearing in each document D. Denote this number by $|q \cap D|$.
3. Choose the top N documents, when sorted *lexicographically,* first by $|q \cap \cdot|$ and then by $P(q|\cdot)$.

The estimation of $P(q|D)$ is the standard document-language model [1], smoothed by the corpus model [4]:

$$P(q|D) = \prod_{i=1}^{n} \lambda P(t_i|D) + (1-\lambda)P(t_i|\mathcal{C}).$$

In this formula, q is a list of *terms* $t_1, \ldots, t_n$ and $\mathcal{C}$ denotes the corpus. The smoothing parameter λ was set to 0.9 in all our submissions. We estimate the probability $P(q|X)$ of generating an individual term t_i in the language model of X (which is either D or $\mathcal{C}$) as

$$P(t_i|X) = \frac{tf(t_i, X)}{\sum_{t \in X} tf(t, X)},$$

where $tf(t, X)$ is the total number of occurrences of t in X.

Utilizing Structural Hints. Our CO+S runs take the query structure (formulated in NEXI [5]) into account in the following simple way. We transform a query q into two different CO queries: The *target query,* q_{tgd}, consists of the terms that appear in the target element of q. The *supporting query,* q_{sup}, consists of the rest of the terms, i.e., all the terms that appear in q, except for those appearing in the target element. In the second filtering step described above, the evaluation of $P(q|D)$ is replaced with a linear interpolation of the probabilities associated with the two queries q_{tgd} and q_{sup}, that is,

$$P(q|D) = \alpha P(q_{tgd}|D) + (1-\alpha)P(q_{sup}|D).$$

Here, $P(q_{tgd}|D)$ and $P(q_{sup}|D)$ are evaluated regularly. In our submissions, we used $\alpha = 0.9$.

3.2 HITS

The second filter, denoted by F_{HITS}, is based on an analysis of the links (given as either XPointers or XLinks) among the Wikipedia documents using the HITS

algorithm [2].[1] In particular, the result of this filter consists of the top-N documents w.r.t. the score obtained by applying HITS.

To describe the application of HITS, we essentially need to define the graph over which HITS is applied. So, consider a query q. The nodes of the graph are the documents in the set S that is constructed by the following three-step process:

1. Construct the set S_q of all documents D, such that a link to D contains one or more terms of q.
2. Apply the filter F_{LM} to s_q and let S_q^f be the set of the top-5 documents of S_q.
3. S includes all the documents of S_q^f, every document that points to a document of S_q^f, and every document that is pointed to by a document of S_q^f.

In the graph, there is an edge from document D_1 to document D_2 if D_1 contains a hyperlink to D_2 (i.e., either an XLink to D_2 or an XPointer to an element of D_2).

4 Element Rankers

The element rankers are applied to the elements of the documents in the filtered corpus in order to obtain the final rank. Again, we use two element rankers. The first ranker, R_{LM}, is based on statistical language modeling. The second ranker, R_{HITS}, combines the first ranker with the rank that the document containing the element obtains when applying HITS.

We first describe the ranker R_{LM} in detail. This ranker uses the element model, smoothed by both the document and corpus models. More formally, given a query q and an element E, we define

$$R_{LM}(E) = \prod_{i=1}^{n} \lambda_1 P(t_i|E) + \lambda_2 P(t_i|D) + \lambda_3 P(t_i|\mathcal{C}) \tag{1}$$

where

- D and $\mathcal{C}$ are the document that contains E and the corpus, respectively;
- $t_1, \ldots, t_n$ are the terms of q; and
- $\lambda_1, \lambda_2, \lambda_3$ are nonnegative smoothing parameters with a total sum of 1.

In our submissions to the Ad-hoc track, $\lambda_1 = 0.8$ and $\lambda_2 = \lambda_3 = 0.1$.

The ranker R_{HITS} is used when the filter is F_{HITS}. Recall that when F_{HITS} is used, the HITS algorithm assigns a rank to each document D in the filtered corpus. Let that rank be denoted as $HITS(D)$. Then, R_{HITS} simply multiplies the rank of R_{LM} by $HITS(D)$, i.e.,

$$R_{HITS}(E) = R_{LM}(E) \cdot HITS(D),$$

where D is the document that contains E.

The rankers have some additional features that are described next.

[1] We used the JUNG library [6] in order to apply HITS.

Table 1. Results in the Thorough task (106 submissions)

runID	method	co/s	MAep	rnk
18_08_02	F_{LM}/R_{LM}	co	0.0709	1
18_11_06	F_{LM}/R_{LM}	cos	0.0708	2
15_11_38	F_{HITS}/R_{LM}	cos	0.0696	4
14_11_03	F_{HITS}/R_{LM}	co	0.0692	5
16_03_32	F_{HITS}/R_{HITS}	cos	0.0664	6
13_09_11	F_{HITS}/R_{HITS}	co	0.0646	8

- **CO+S.** In the case of CO+S, the probability estimation (1) uses only the terms in the target of the query (i.e., q_{tgd}).
- **Length cutoff.** We filter out elements that are too short. That is, we predefined a *cutoff* value c and ignored all the elements shorter than c.
- **Quoted expressions.** In Equation (1), we consider each quoted expression as a single term. So, for example, the query "'West coast' musician" consists of two terms: t_1 ="West coast" and t_2 ="musician." So, to evaluate the term frequency (in either the corpus or the document) of a term t_i, we counted the number of consecutive appearances of the keywords of t_i (but not necessarily in the original order). Furthermore, to evaluate $P(t_i|E)$, we used a more flexible measure, namely, we allowed the keywords of t_i to be sufficiently "close" to each other (i.e., reside in a small interval).

5 Submissions and Results

In this section, we describe the runs submitted to the INEX 2006 Ad-hoc track. In each task, several combinations of filters, rankers and parameters were used. When needed, additional processing was performed so that each submission fits the specific requirements of its task.

5.1 Thorough Task

The Thorough task does not require elimination of element overlap. The evaluation results of our runs in this task, using the *filtered assessments*, are shown

Table 2. Results in *Focused* task, with overlap on (85 submissions)

			nxCG[5]		nxCG[10]		nxCG[25]		nxCG[50]	
runID	method	co/s	Score	rnk	Score	rnk	Score	rnk	Score	rnk
15_12_28	F_{HITS}/R_{LM}	co	0.3659	5	0.3275	3	0.2678	5	0.2257	5
16_08_52	F_{HITS}/R_{LM}	cos	0.3460	9	0.3103	7	0.2663	6	0.2250	6
16_12_44	F_{HITS}/R_{HITS}	cos	0.3244	25	0.2891	15	0.2449	11	0.2077	12
18_09_38	F_{LM}/R_{LM}	co	0.3547	6	0.3247	4	0.2810	1	0.2450	2
18_12_32	F_{LM}/R_{LM}	cos	0.3366	15	0.3103	6	0.2736	2	0.2474	1

Table 3. Results in *Focused* task, with overlap off (85 submissions)

runID	method	co/s	nxCG[5] Score	rnk	nxCG[10] Score	rnk	nxCG[25] Score	rnk	nxCG[50] Score	rnk
15_12_28	F_{HITS}/R_{LM}	co	0.4066	4	0.3827	2	0.3312	1	0.2770	2
16_08_52	F_{HITS}/R_{LM}	cos	0.3890	6	0.3697	4	0.3302	2	0.2816	1
16_12_44	F_{HITS}/R_{HITS}	cos	0.3999	5	0.3626	8	0.3152	5	0.2660	3
18_09_38	F_{LM}/R_{LM}	co	0.3878	7	0.3670	5	0.3163	4	0.2620	5
18_12_32	F_{LM}/R_{LM}	cos	0.3684	12	0.3506	9	0.3081	7	0.2639	4

in Table 1. Later, we consider the results using the non-filtered assessments. For each run, the table specifies the run identifier (**runID**), the filter-ranker combination (**method**), whether the run is CO or CO+S (**co/s**) and the *rank* of the result (**rnk**), i.e., its position among the submissions of all the participants in this task. For example, the first line describes Run 18_08_02 with the following properties. The filter is F_{LM}, the ranker is R_{LM} and the query was considered as content-only (co). The MAep score of this run is 0.0709 and it is the best run in the Thorough task. Our runs on this task use the same parameters. In particular, the length cutoff is 20.

Table 1 shows that, in general, our submissions provide a good tradeoff of effort vs. recall-gain (compared to the other submissions). Furthermore, under this yardstick, the use of HITS does not improve our runs, that is, it is best to use the language-model approach for both filtering and ranking.

The ranks of the results of our runs among those using the non-filtered assessments are very similar to those described above (using the filtered assessments). A remarkable difference is the following. In the non-filtered results, Run 15_11_38 (forth in the filtered assessments) jumped to the first place, pushing Runs 18_03_02 and 18_11_06 to the second and third places, respectively.

5.2 Focused Task

In the Focused task, overlapping elements were eliminated as follows. For each document D in the filtered corpus, we listed all the elements of D in descending

Table 4. Results in *BestInContext* task, Metric BEPD (77 submissions)

runID	method	co/s	A=0.01 Score	rnk	At A=0.1 Score	rnk	At A=1.0 Score	rnk	At A=10.0 Score	rnk	At A=100.0 Score	rnk
19_08_40	F_{LM}/R_{LM}	cos	0.1604	8	0.2329	7	0.3502	8	0.5437	8	0.7451	10
20_12_09	F_{LM}/R_{LM}	cos	0.1441	15	0.2166	16	0.3365	18	0.5348	15	0.7430	11
19_03_22	F_{LM}/R_{LM}	co	0.1610	6	0.2334	6	0.3493	9	0.5404	9	0.7374	13
19_06_40	F_{LM}/R_{LM}	co	0.1469	11	0.2190	15	0.3374	17	0.5327	16	0.7357	17
17_11_09	F_{HITS}/R_{HITS}	co	0.1127	33	0.1644	40	0.2474	48	0.3946	48	0.5548	50
17_09_18	F_{HITS}/R_{HITS}	cos	0.1097	34	0.1619	41	0.2458	49	0.3886	49	0.5448	53

Table 5. Results in *BestInContext* task, Metric EPRUM-BEP-Exh-BEPDistance (77 submissions)

			A=0.01		At A=0.1		At A=1.0		At A=10.0		At A=100.0	
runID	**method**	**co/s**	**Score**	**rnk**	**Score**	**rnk**	**Score**	**rnk**	**Score**	**rnk**	**Score**	**rnk**
20_12_09	F_{LM}/R_{LM}	cos	0.0251	21	0.0428	31	0.0752	30	0.1421	30	0.2320	25
19_08_40	F_{LM}/R_{LM}	cos	0.0292	7	0.0486	13	0.0809	21	0.1460	22	0.2325	23
19_03_22	F_{LM}/R_{LM}	co	0.0286	9	0.0477	16	0.0801	22	0.1481	18	0.2372	18
19_06_40	F_{LM}/R_{LM}	co	0.0258	19	0.0433	30	0.0758	29	0.1460	22	0.2367	19
17_11_09	F_{HITS}/R_{HITS}	co	0.0239	22	0.0423	32	0.0741	37	0.1469	21	0.2515	16
17_09_18	F_{HITS}/R_{HITS}	cos	0.0258	16	0.0441	26	0.0768	25	0.1489	15	0.2528	15

rank. We then traversed the list (in the order of descending rank) and removed every element that overlapped with any previous element.

Tables 2 and 3 show the results in the Focused task, with overlap on and off, respectively. These results correspond to the filtered assessments. (We later consider the non-filtered ones). All the runs use the same parameters (the length cutoff is 20). Note that the specified ranks (denoted **rnk** in the tables) are the *effective* ranks of our runs, i.e., with the invalid submissions excluded.

Consider Table 2, with overlap on. In all the runs, we used R_{LM} for element ranking. The first run (using F_{HITS} as a filter) yields the best scores among our runs for nxCG[5] and nxCG[10]. However, for nxCG[25] and nxCG[50], the fourth and fifth runs (that use the filter F_{LM}) are better and, in fact, the best among all submissions. A different behavior is exposed in Table 3 (overlap off). There, the first run is superior to the others under all the nxCG metrics, except for nxCG[50] where it is second (and the second run is the best).

In the non-filtered assessments, our runs got almost identical ranks, except for a few minor differences. The most significant difference is that Run 18_12_32, ranked 6 in the overlap-on task under the metric nxCG[10], is only ranked 9 in the corresponding list of non-filtered results.

5.3 Best-in-Context and All-in-Context Tasks

In the Best-in-Context task, the element with the highest score was chosen for each document. The results in this task, under the metrics BEPD and EPRUM-BEP-Exh-BEPDistance, are shown in Tables 4 and 5, respectively. The length cutoff was 30 in runs 19_06_40 and 20_12_09. In the other runs, it was 20.

In the All-in-Context task, overlap was eliminated similarly to the Focused task. Table 6 shows the results in the All-In-Context task. The length cutoff was 10 in runs 19_01_56 and 18_11_02; in the other runs, it was 20.

Compared to other submissions, our methods obtained better results in the first two tasks than in the third and fourth. Note, however, that in the Best-in-Context task, Run 19_08_40 (Table 4) was among the top 10 under the BEPD metric for all values of A. Furthermore, in the All-in-Context task, the first four runs of Table 6 are always among the top 13 and they are significantly better than the fifth and sixth.

Table 6. Results in *AllInContext* task (56 submissions)

			MAgP		gP[5]		gP[10]		gP[25]		gP[50]	
runID	**method**	**co/s**	**Score**	**rnk**	**Score**	**rnk**	**Score**	**rnk**	**Score**	**rnk**	**Score**	**rnk**
18_11_02	F_{LM}/R_{LM}	co	0.1601	7	0.3176	13	0.2585	13	0.1956	9	0.1446	8
18_09_30	F_{LM}/R_{LM}	co	0.1599	8	0.3198	12	0.2603	11	0.1957	8	0.1440	9
19_12_30	F_{LM}/R_{LM}	cos	0.1584	9	0.3303	7	0.2631	9	0.1927	11	0.1411	12
19_01_56	F_{LM}/R_{LM}	cos	0.1584	10	0.3262	10	0.2600	12	0.1927	10	0.1418	10
17_03_42	F_{HITS}/R_{HITS}	cos	0.0353	52	0.0925	52	0.0788	52	0.0625	52	0.0441	53
17_02_23	F_{HITS}/R_{HITS}	co	0.0348	53	0.0886	53	0.0781	53	0.0610	53	0.0445	52

6 Conclusion

In our participation in the INEX 2006 Ad-hoc retrieval track, we mainly studied two retrieval techniques. The first is a preliminary filtering of the corpus in order to obtain the documents from which the actual elements are considered. The second is the use of HITS for either filtering of documents or ranking of elements. The results of our submissions show that preliminary filtering improves the quality of retrieval, since our runs were among the best in the Thorough and Focused tasks. Furthermore, the use of HITS is useful for appropriately identifying the few top elements. In comparison, language models generally yielded better results in identifying large collections of relevant elements. In future work, we plan to further study the integration of the techniques presented in this paper in order to achieve the best of both worlds. In particular, we will study the use of PageRank [7], instead of HITS, for link analysis.

References

1. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR, pp. 275–281. ACM Press, New York (1998)
2. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. In: SODA, pp. 668–677. ACM Press, New York (1998)
3. Kamps, J., de Rijke, M., Sigurbjörnsson, B.: Length normalization in XML retrieval. In: SIGIR, pp. 80–87. ACM Press, New York (2004)
4. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to ad hoc information retrieval. In: SIGIR, New York, NY, USA, pp. 334–342. ACM Press, New York (2001)
5. Trotman, A., Sigurbjörnsson, B.: Narrowed extended XPath I (NEXI). In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 16–40. Springer, Heidelberg (2005)
6. The JUNG Framework Development Team: JUNG java universal network/graph framework (2006) `http://jung.sourceforge.net`
7. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks 30(1-7), 107–117 (1998)

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 ad hoc 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 to XML elements in the Wikipedia collection. We also investigate whether using non-uniform priors is beneficial for the ad hoc tasks.

1 Introduction

In this paper we describe the Queen Mary, University of London's participation in the INEX 2006 ad hoc track.

Content-oriented XML retrieval systems aim at supporting more precise access to XML repositories by retrieving XML document components (the so-called XML elements) instead of whole documents in response to users' queries. Therefore, in principle, XML elements of any granularity (for example a paragraph or the section enclosing it) are potential answers to a query, as long as they are relevant. However, the child element (paragraph) may be more focused on the topic than its parent element (the section), which may contain additional irrelevant content. In this case, the child element is a better element to retrieve than its parent element, because not only it is relevant to the query, but it is also specific to the query.

To score XML elements according to how exhaustive and specific they are to a given 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 (e.g., [6,7,5]). 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 means to capture specificity. Next, to compare this new feature to element length, we follow the spirit of [5], and incorporate this feature within a language modeling

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 261–270, 2007.

framework, and examine its effects on the Wikipedia collection, compared to element length, to estimate prior probability of relevance in XML retrieval. Our previous work on IEEE collection showed that for retrieving highly specific and highly exhaustive elements, using topic shifts as prior is useful in the framework of language modeling [2].

For our experiments, we have implemented retrieval approaches for ranking XML elements based on a statistical language modelling approach [4]. Language modelling approaches have shown satisfactory results in content-oriented XML retrieval (e.g. [5,8]). This approach allows us to combine "non-content" features of elements (or documents) (e.g. length, topic shifts) with the scoring mechanism. We incorporate this new source of evidence, the number of topic shifts, as prior probability of relevance in the framework of language modeling. We also incorporate the number of topic shifts in the smoothing process within this framework as a means to capture specificity.

For the Thorough task, we experiment with two different ways of smoothing, element-specific smoothing and fixed smoothing approaches within the language modeling framework. We also compare topic shifts to element length, by incorporating each of them as prior probability of relevance and examining their effects on the retrieval effectiveness. For the Focused task, we apply a post-filtering algorithm to remove overlapping elements from our Thorough runs. We investigate whether using element-specific smoothing is beneficial for the Focused task. We also examine the usefulness of non-uniform priors for the Focused task. For the All In Context task, we took our runs for the Focused task, reordered the first 1,500 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 6. Section 7 concludes the paper.

2 Topic Shifts

In this section, we describe how we measure the number of topic shifts of the elements forming an 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[1] [3], which has been successfully used in several IR applications. 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

[1] http://elib.cs.berkeley.edu/src/texttiles/

segments. TextTiling is a linear segmentation algorithm that 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 is defined as:

$$score(e) := actual\_topic\_shifts(e) + 1 \tag{1}$$

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

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

3 Retrieval Framework

For our experiments, we have implemented retrieval approaches for ranking XML elements based on a statistical language modelling approach [4]. We rank elements based on the likelihood for a query $q = (t_1, t_2, ..., t_n)$ to be generated from an element e as:

$$P(e|q) \propto P(e) * P(t_1, .., t_n|e) \tag{2}$$

where

$$P(t_1, .., t_n|e) = \prod_{i=1}^{n} (\lambda_e P(t_i|e) + (1 - \lambda_e) P(t_i|C)) \tag{3}$$

and

- t_i is a query term in q,
- $P(e)$ is the prior probability of relevance for element e,
- $P(t_i|e) = \frac{tf(t_i,e)}{\sum_t tf(t,e)}$ is the probability of generating the query term t_i from element e,
- $tf(t, e)$ is the number of occurrences of term t in element e,
- $P(t_i|C) = \frac{ef(t_i)}{\sum_t ef(t)}$ is the probability of query term t_i in the collection,
- $ef(t)$ is the total number of XML elements in which term t occurs, and
- λ_e (weight on the element language model) is a weighting parameter between [0,1] which is used in smoothing the element model with the collection model.

We experiment with two ways of smoothing. First, we set λ_e to 0.1 for all elements, referred as fixed smoothing. This value is close to the traditional setting for document retrieval (λ=0.15), which has shown satisfactory results [4].

Secondly, to accommodate for the specificity dimension, we propose to set λ_e, the amount of smoothing, to be proportional to the number of topic shifts in the element, referred as element-specific smoothing. The idea of incorporating topic shifts in this manner originates from the fact that if the number of topic shifts in an element is low and an element is relevant, then it is likely to contain less non-relevant information compared to the case where a high number of topic shifts exists (For a complete argument see [1]). We define the element-specific smoothing parameter, λ_e, to be inversely proportional to the number of topic shifts in element e:

$$\lambda_e = \frac{\lambda}{score(e)} \tag{4}$$

where λ is a constant parameter between [0,1]. We set λ to 0.1 in the experiments where we use the element-smoothing approach.

We experiment with two different prior probabilities of relevance $P(e)$. First, following the work of Kamps etal in [5], we define the prior probability of relevance to be proportional to the length of an element. We refer to it as **length prior**:

$$P(e) = \frac{\sum_t tf(t,e)}{\sum_e \sum_t tf(t,e)} \tag{5}$$

Second, we define the prior probability to be proportional to the number of topic shifts in an element. We refer to it as **topic shifts prior**:

$$P(e) = \frac{score(e)}{\sum_e score(e)} \tag{6}$$

We also compare these two approaches with a baseline using a uniform prior. The uniform prior gives all elements an equal prior probability of being relevant.

4 Retrieval Setting

For calculating the number of topic shifts in each XML element, 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 $K = 6$. As a heuristic $W * K$ is equal to the average paragraph length (in terms of the number of terms) [3].

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 Evaluation

For all tasks, we use the official metrics of INEX 2006. Since we only index and retrieve elements in the paragraph level or above, using the filtered assessment set will not change the relative order of our approaches considerably. Therefore we only reports the results using the full assessment set. In addition we report results for the Focused task using the strict quantization function. The strict quantization function is used to evaluate XML retrieval methods with respect to their capability of retrieving highly specific elements (s=1).

6 Experiments

6.1 Thorough Task

For the Thorough task we experiment with two different ways of smoothing, element-specific smoothing and fixed smoothing approaches ($\lambda = 0.1$ for all elements) in the framework of language modeling. We also consider both length and the number of topic-shifts as prior in addition to the uniform prior probability of relevance. Therefore, we consider six retrieval approaches in our experiments. Table 1 shows the details of our retrieval approaches where those runs submitted to INEX 2006 are marked with $*$.

Table 1. Thorough Retrieval Approaches

Approach	Prior	Smoothing
Lm_T	uniform	fixed
Lm_ToicShiftsPrior_T*	topic shifts	fixed
Lm_LengthPrior_T*	length	fixed
Lm_TermWeighted_T	uniform	element-specific
Lm_ToicShiftsPrior_TermWeighted_T*	topic shifts	element-specific
Lm_LengthPrior_TermWeighted_T	length	element-specific

Table 2. Thorough retrieval task: Evaluation based on Mean Average effort precision (MAep), using generalized quantization function

Approach	*MAep*
Lm_T	0.0179
Lm_ToicShiftsPrior_T*	**0.0185**
Lm_LengthPrior_T*	0.0181
Lm_TermWeighted_T	0.0144
Lm_ToicShiftsPrior_TermWeighted_T*	0.0163
Lm_LengthPrior_TermWeighted_T	0.0168

Table 2 presents, the evaluation results for Mean Average effort precision (*MAep*) for the six retrieval approaches. Focusing on either fixed smoothing approaches or approaches using element-specific smoothing, we observe that, using either the length prior or the topic shifts prior leads to slightly improvements of performance.

Focussing on the approaches employing non-uniform priors, we observe that they perform comparably, but none of them considerably improves the retrieval effectiveness compared to uniform prior, when evaluated with *MAep*.

When comparing the results for the approaches using fixed smoothing and element-specific smoothing, we see that fixed smoothing approaches are more effective in terms of *MAep* when evaluated under the generalized case.

6.2 Focused Task

The INEX 2006 Focused task asks systems to find the most focused elements that satisfy an information need, without returning "overlapping" elements. We experiment with the same approaches as we discussed for the Thorough task, and remove Overlap by applying a post-filtering on the retrieved ranked list. We select 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. Therefore, we consider six retrieval approaches in our experiments. Table 3 shows the details of our retrieval approaches where those runs submitted to INEX 2006 are marked with $*$.

Table 3. Focused Retrieval Approaches

Approach	Prior	Smoothing
Lm_F	uniform	fixed
Lm_ToicShiftsPrior_F*	topic shifts	fixed
Lm_LengthPrior_F*	length	fixed
Lm_TermWeighted_F	uniform	element-specific
Lm_ToicShiftsPrior_TermWeighted_F*	topic shifts	element-specific
Lm_LengthPrior_TermWeighted_F	length	element-specific

Table 4. Focused retrieval task: *MAep* and normalised eXtended Cumulated Gain (*nxCG*) at different cut-off

Approach	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
General					
Lm_F	0.3477	**0.3075**	**0.2368**	**0.1818**	**0.0392**
Lm_TopicShiftsPrior_F*	0.3429	0.2953	0.2223	0.1649	0.0365
Lm_LengthPrior_F	0.3386	0.2751	0.2038	0.1559	0.035
Lm_TermWeighted_F	**0.3532**	0.2983	0.2282	0.1704	0.0369
Lm_TopicShiftsPrior_TermWeighted_F*	0.3455	0.2965	0.2351	0.1774	0.0382
Lm_LengthPrior_TermWeighted_F*	0.3525	0.2957	0.2276	0.1708	0.0381
Strict					
Lm_F	0.2937	0.2578	0.1929	0.1447	0.025
Lm_TopicShiftsPrior_F*	0.2721	0.2225	0.1627	0.1188	0.0193
Lm_LengthPrior_F	0.2342	0.182	0.131	0.101	0.016
Lm_TermWeighted_F	**0.3117**	**0.2614**	0.1965	0.1444	**0.0262**
Lm_TopicShiftsPrior_TermWeighted_F*	0.3009	0.256	**0.1993**	**0.1475**	0.0261
Lm_LengthPrior_TermWeighted_F*	0.3045	0.2487	0.1872	0.1377	0.0245

The evaluation results with respect to the *MAep* and *nxCG* at four different early cut-off points (5, 10, 25, 50) are shown in Table 4. For both evaluations, both strict and generalised quantization functions are used.

Under the generalized case, with all early precision measures apart from *nxCG*@5 using uniform prior and fixed smoothing approaches is the most effective approach.

Under the strict case, when comparing the results for the approaches using fixed smoothing and element-specific smoothing, we see that element-specific smoothing approaches, the bottom three runs, are more effective at early precision. This observation indicates the potential use of element-specific smoothing for retrieving highly specific elements at the early ranks.

The results also suggest that using non-uniform prior is not beneficial for the Focused task.

6.3 All in Context

For the All In Context task, we took our runs for the Focused task, reordered the first 1,500 elements in the list such that results from the same article are clustered together. We aim at examining the capability of our approaches in locating the relevant content within the relevant articles. Table 5 shows the details of our retrieval approaches where those runs submitted to INEX 2006 are marked with $*$.

The evaluation results with respect to Mean Average Generalized Precision (*MAgP*) and Generalized Precision (*gP*) at four different early cutoff point (5, 10, 25, 50) are shown in Table 6. Under all the official metrics of INEX 2006 for this task, using length prior and fixed smoothing provides the most effective approach in locating the relevant content within the relevant article. Using length

prior leads to considerable improvement over the uniform prior for $gP@5$. For the other measures used for this task, using both length and topic shifts prior leads to slight improvements of performance compared to the uniform prior.

Table 5. All In Context Retrieval Approaches

Approach	Prior	Smoothing
Lm_F_Clustered_R	uniform	fixed
Lm_TopicShiftsPrior_F_Clustered_R*	topic shifts	fixed
Lm_LengthPrior_F_Clustered_R	length	fixed
Lm_TermWeighted_F_Clustered_R	uniform	element-specific
Lm_TopicShiftsPrior_TermWeighted_F_Clustered_R*	topic shifts	element-specific
Lm_LengthPrior_TermWeighted_F_Clustered_R*	length	element-specific

Table 6. All In Context retrieval task Mean Average Generalized Precision ($MAgP$) and Generalized Precision at early ranks (gP) different cut-off

Approach	gP@5	gP@10	gP@25	gP@50	MAgP
Lm_F_Clustered_R	0.2572	0.2308	0.1760	0.1259	0.1147
Lm_TopicShiftsPrior_F_Clustered_R*	0.2593	0.2318	0.1759	0.1262	0.1179
Lm_LengthPrior_F_Clustered_R	**0.2833**	**0.2386**	**0.1766**	**0.1278**	**0.1232**
Lm_TermWeighted_F_Clustered_R	0.2489	0.2221	0.1637	0.1128	0.1028
Lm_TopicShiftsPrior_TermWeighted_F_Clustered_R*	0.2582	0.2270	0.1692	0.1193	0.1084
Lm_LengthPrior_TermWeighted_F_Clustered_R*	0.2595	0.2254	0.1702	0.1188	0.1105

6.4 Best in Context

The INEX 2006 Best In Context task asks systems to find the XML elements that corresponds to the best entry points to read articles. For the Best In Context task, we examine whether the most focused element in a relevant document is a good choice for the best entry point in a relevant article. For this task we took our official runs for the Focused task, and return for each article, the element with the maximum score as the best entry point. Table 7 shows the details of our official retrieval approaches. The last run marked with Δ 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.

Table 7. Best in Context Retrieval Approaches

Approach	Prior	Smoothing
Lm_TopicShiftsPrior_F_B*	topic shifts	fixed
Lm_TopicShiftsPrior_TermWeighted_F_B*	topic shifts	element-specific
Lm_TopicShiftsPrior_TermWeighted_Foarent_B*Δ	topic shifts	element-specific

We report the INEX 2006 official results using the EPRUM-BEP-Exh-BEPDistance and BEPD metrics at five different values for A (0.01, 0.1, 1, 10, 100) as shown

in table 8. Low values of A (e.g. 0,1) favour runs that return elements very close to a best entry point.

Comparing the results of our runs, using fixed smoothing is the most effective runs for both metrics and for all values of A except at $A = 0.01$ where the approach based on element-specific smoothing outperformed. When evaluating these runs with EPRUM-BEP-Exh-BEPDistance at $A = 0.01$ (*low value*), our runs ranked very high among all participants. This shows that element-specific smoothing is useful at returning the elements very close to a best entry point in relevant articles.

Table 8. Best In Context task: EPRUM-BEP-Exh-BEPDistance and BEPD metrics

Approach	A=0.01	A=0.1	A=1	A=10	A=100
EPRUM-BEP-Exh-BEPDistance					
Lm_TopicShiftsPrior_F_B	0.0314	**0.0468**	**0.0735**	**0.1284**	**0.2024**
Lm_TopicShiftsPrior_TermWeighted_F_B	**0.0325**	0.0410	0.0610	0.1134	0.1855
Lm_TopicShiftsPrior_TermWeighted_Fparent_B	0.0300	0.0393	0.0607	0.1134	0.1855
BEPD					
Lm_TopicShiftsPrior_F_B	0.1129	**0.1611**	**0.2536**	**0.4092**	**0.5760**
Lm_TopicShiftsPrior_TermWeighted_F_B	**0.1259**	0.1591	0.2315	0.3815	0.5490
Lm_TopicShiftsPrior_TermWeighted_Fparent_B	0.1201	0.1587	0.2344	0.3836	0.5494

7 Discussion and Summary

This paper describes the retrieval approaches used by Queen Mary, University of London in the INEX 2006 ad hoc track. We participated in all four ad hoc track tasks. In this work, we experimented with two different ways of smoothing, fixed smoothing and element-specific smoothing approaches within the language modeling framework. We also investigated whether using non-uniform priors is beneficial for the ad hoc tasks in the Wikipedia collection. Our main findings are the following:

- Our results suggest that the the fixed smoothing approach is useful in several cases: (i) for the Thorough task, in terms of *MAep* and under the generalized quantization function, (ii) for the Focused task, for all early precision measures apart from $nxCG@5$ and under the generalized quantization function, (iii) for the All In Context task, in locating the relevant content within the relevant article with all the measures used for this task, (iv) for the Best In Context task, at finding the best entry point in the relevant elements for the values of $A > 0.01$ (we are looking for the elements very close to a best entry point when $A = 0.01$).
- For the element-specific smoothing, we used the number of topic shifts in the smoothing process. The idea of incorporating topic shifts in the element-specific smoothing approach originated from the fact that if the number of topic shifts in an element is low and an element is relevant, then it is likely to

contain less non-relevant information compared to the case with high number of topic shifts. Therefore, in this way of smoothing, in fact, we reward the presence of a query term in an element with a lower number of topic shifts (a more specific element). This means that we are capturing specificity with the number of topic shifts. Our results suggest that element-specific approach is useful in the following cases: (i) for the Focused task, in finding the highly specific elements at the early ranks, (ii) for the Best In Context, in finding the elements very close to a best entry point in relevant documents, i.e., for $A = 0.01$. These results indicate that the number of topic shifts is a useful evidence, as it seems to capture the specificity dimension of relevance.

– Finally, we observed that in general, using non-uniform prior is slightly beneficial for the Thorough and All In Context tasks, but not beneficial for the Focused task.

References

1. Ashoori, E., Lalmas, M.: Using topic shifts for focussed access to XML repositories. In: Advances in Information Retrieval: Proceedings of the 29th European Conference on IR Research (ECIR) (April 2007)
2. Ashoori, E., Lalmas, M., Tsikrika, T.: Examining Topic Shifts in Content-Oriented XML Retrieval, submitted (2006)
3. Hearst, M.A.: Multi-paragraph segmentation of expository text. In: Proceedings of the 32nd annual meeting on Association for Computational Linguistics, pp. 9–16 (1994)
4. Hiemstra, D.: Using Language Models for Information Retrieval. Phd thesis, University of Twente (2001)
5. Kamps, J., de Rijke, M., Sigurbjörnsson, B.: The importance of length normalization for XML retrieval. Information Retrieval 8(4), 631–654 (2005)
6. Mass, Y., Mandelbrod, M.: Using the inex environment as a test bed for various user models for XML retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 187–195. Springer, Heidelberg (2006)
7. Ogilvie, P., Callan, J.: Hierarchical language models for XML component retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 224–237. Springer, Heidelberg (2005)
8. Ramirez, G., Westerveld, T., de Vries, A.P.: Using structural relationships for focused XML retrieval. In: Larsen, H.L., Pasi, G., Ortiz-Arroyo, D., Andreasen, T., Christiansen, H. (eds.) FQAS 2006. LNCS (LNAI), vol. 4027, pp. 147–158. Springer, Heidelberg (2006)

XSee: Structure Xposed

Roelof van Zwol[1] and Wouter Weerkamp[2,⋆]

[1] Yahoo! Research, C/ Ocata 1, 08003 Barcelona, Spain
roelof@yahoo-inc.com
[2] University of Amsterdam, ISLA, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
weerkamp@science.uva.nl

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 the effectiveness of the retrieval model can be enhanced using different reranking methods based on the structural characteristics.

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 [3]. 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 *X*ML *Se*arch *e*ngine, nicknamed XSee. The retrieval model for XSee is derived from the GPX [2] and the B3-SDR model [6], and is developed to provide a scalable solution for the rapidly growing XML collections that are available, while maintaining a good retrieval performance.

⋆ The research described in this article was caried out at Utrecht University. In the meantime both authors have moved forward to a new position elsewhere.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 271–283, 2007.

Second, we discuss how reranking methods can be deployed to achieve a significant increase in retrieval performance, based on the analysis of the document structure present in both the IEEE and Wikipedia XML collections. Comparison of the characteristics of the XML collection, with the full and ideal recall base provides insight into the type of elements that should be retrieved as the result of a particular task. We will discuss four of the reranking methods that we have deployed, based on the characteristics of the retrieved elements: path depth of the element, number of textual terms (node size) contained in the element, number of descendant nodes for an element, and the element being a leaf node. The reranking methods can be used both on the Thorough and Focused task. In this article we will present a summary of our findings; more detailed information can be found in [7]. We will present a straightforward procedure for overlap removal that is used to construct a focused run, and we present a so called de-reranking of the result list, which can be applied to construct a focused run that obeys the original ranking of elements.

Based on the official results for the Thorough task we discuss the optimal configuration of the XSee retrieval model, and discuss its stability on different document collections. Evaluating the effectiveness of the reranking methods is a complex task, due to the number of variables involved. We will present the effect of the reranking methods on both the Thorough and Focused task, and compare the outcome with the baseline run for each task.

Organisation

In the remainder of the article, we will first present the XSee retrieval model for content-only queries in Section 2. In Section 3 we discuss structural aspects of the IEEE and Wikipedia collections, and describe the four reranking methods that we have deployed to improve the effectiveness of the retrieval performance. In Section 4 we present the results on the Thorough task, compare the performance of the model on both the IEEE, and Wikipedia collection, and discuss the results for the reranking methods. The results of the Focused task are presented in Section 5, where we discuss the configuration for the baseline run, and the effect of the reranking methods on the Focused task. Finally, we will come to our conclusions and present future work in Section 6.

2 *XSee*, the *X*ML *Se*arch *E*ngine

The XSee XML search engine is primarily build for participation in INEX 2006. In this section, a brief overview of the internals of the XSee system is given to provide more insight into the particulars of an XML retrieval system in general, but also as a reference for the explanation of the results of the experiment described below.

2.1 XSee Data Structure

The XSee system uses the regular pre-processing steps to clean and standardize the data: lexical cleaning, stopword removal, stemming (optional), and term

Table 1. XSee data structure

TermIndex(term, stemmed term, start, end, term frequency)
TermLeaveFrequency(node id, position)
Node(id, parent, name, document, path, depth)

internationalization (optional). The XSee data structure is based on the traditional inverted file approach used for many IR tasks. Table 1 shows the underlying structure used by the retrieval strategy that is explained hereafter.

A term index is constructed to contain a list of unique terms appearing in the document collection, with their stemmed variant and the frequency of term appearances throughout the document collection. A start- and end position is stored for each term that refer to a slice of the TermLeaveFrequency table. This table contains tuples with a reference to an XML node and the position of the term within that node. The third table contains node information, like a pointer to the parent node, the node name, the document and path combination, that uniquely identifies a node, and the depth of the node in the XML tree.

2.2 XSee Retrieval Model

In this section we will shortly introduce the retrieval strategy used by XSee. It is derived from the GPX [2] and B^3-SDR models [6] that have been developed and evaluated within INEX. The XSee system is custom built for participation in INEX 2006. The objective was to derive a model that was comparable in retrieval performance to the GPX model, but also to have a scalable approach in terms of system performance. In the next section we will discuss the main differences with the GPX model by the hand of an example.

At indexing time, a termweight for a term q and a node n is calculated using:

$$termweight(n,q) = \begin{cases} \frac{tf_{n,q}}{TF_q}, if \quad is\_leaf(n), \\ \frac{tf_{n,q}}{TF_q} + \sum_{c \in children(n)} D_{node} \cdot termweight(c,q) \end{cases} \tag{1}$$

For a leaf node, the weight of a term is determined using the term frequency of that term within the node divided by the number of times the term occurs in the document collection. If the node is not a leaf, the termweight also depends (recursively) on the termweight of its children. The influence of a term occuring in one of its children is reduced via the node decay D_{node}. Typical values for D_{node} are in the range [0.09..0.49], as is heuristically determined.

At query time, the relevance score of a node for a given query Q is calculated as:

$$rsv(n,Q) = F_t^{|\{q|q \in Q \ \wedge q \in terms(n)\}|-1} \cdot \sum_{q \in Q} termweight(n,q) \tag{2}$$

For each query term q in the query, it is determined whether this term is contained in the node. The sum over the corresponding termweights is taken and multiplied by the termfactor F_t to the power of the number of unique query terms

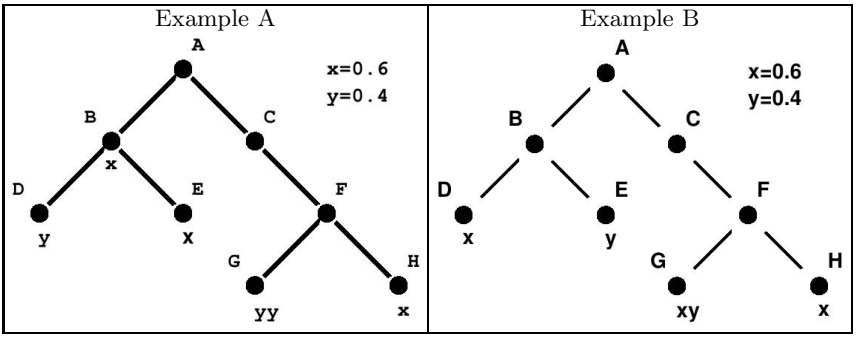


Fig. 1. XML tree of Example A and B

occuring in node n. This assures that nodes containing all the query terms will be pushed to the top of the ranking.

Additional features of the retrieval model include phrase-detection, detection of exclusionary terms, the 'too small' result filter[1], and CAS-based extensions for efficient use of structural clues.

2.3 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 gives 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. That way only the result for Equation 1 needs to be calculated at query time.

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 2. The Figure shows an XML tree with 8 nodes, and the occurrance 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 ($F_t = 5$) respectively. For both examples we assume the following term weights: $x = 0.6$, and $y = 0.4$.

The data under Example A of Table 2 shows the results for this example XML tree. For each node, its position in the ranking and the computed relevance

[1] Initially there was no notion of 'too small' elements in INEX 2006 due to the assessment method used and this filter was therefore not used.

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.

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.

3 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.

3.1 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 3 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.

Table 2. Relevance scores for Example A and B

	Example A		Example B	
Node	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 3. 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

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.

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.

3.2 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.

The idea of using structural characteristics to increase ranking performance has been applied before; In [4] a ranking schema *PTF* is used in which assumed relevant elements are first ordered by decreasing path depth, before looking at (distinct) query term frequency. In [5] the assumption is made that smaller nodes are more important and these are therefore pushed up in the ranking. Neither of the two articles though presents evidence on which these assumptions could be justified, nor do they explore the influence of using different structural characteristics. In the previous section we already showed the differences in characteristics between collections and recall-bases, on which we can justify the reranking

methods described below. Comparing the performances of the various reranking methods also allows us to get insight in the influence of available structural information.

Rerank on node size. Given a run R and a result element $r \in R$ 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 recalculating 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 ranked higher, using the formula: $rsv_{r'} = path\_depth_r \cdot rsv_r$.

4 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 we present improvements over the baseline that are obtained by applying the reranking methods discussed in the previous section.

4.1 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 models should be stable for different collections. In Table 4 the overall results for the official runs are presented. It shows that the configuration of the 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.

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

Table 4. Overall performance of the official runs on the Thorough task

	$nd:.39 F_t:4$	$nd:.29 F_t:3$	$nd:.49 F_t:5$
MAep on IEEE	0.0767	0.0757	0.0708
MAep on Wikipedia	0.0302	0.0294	0.0290
Ranked	14	15	16

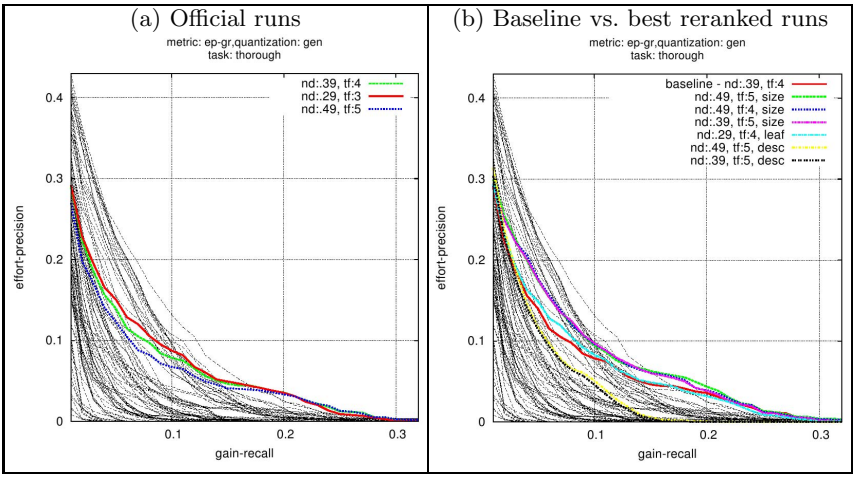


Fig. 2. Thorough runs: Effort precision vs. Gain recall

Thorough task on a range of node decays (0.09 - 0.49) and term factors (1 - 10), which results in the matrix of Table 5. 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 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.

4.2 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 3, we have post-processed the set of Thorough runs to see how we can benefit from the available of structure. Table 6 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,

Table 5. 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

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 an increase in performance of 8.3% on nxCG@5 is obtained for the configuration $[nd : .49, F_t : 5, desc]$, where reranking is based on the number of descendants.

Table 6. Effect of rerank based on structure

Run	MAep	nxCG@5	nxCG@10	nxCG@25	nxCG@50
$nd : .39, F_t : 4$	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
Color coding:	**Best run**	**Top 3**	Top 10	Top20	Other

Figure 2.b 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 nxCG@5 measures.

5 Focused Task

Goal of the Focused task is to return relevant elements regarding a query without elements overlapping each other. In Section 3 various methods 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 measures, 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.

5.1 Focused Runs - Submissions

Based on evaluation results of various configurations 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 summarized in Table 7.

Table 7. Performance results for the official Focused runs

characteristic	$nd : .19, F_t : 5$ *size rerank*	$nd : .04, F_t : 3$ *leaf rerank*	$nd : .04, F_t : 2$ *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 measure with low cut-off levels we see that the run with leaf reranking $[nd : .04, F_t : 3]$ 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 $[nd : .19, F_t : 5]$ performs best, and is ranked 11–16 on nxCG. The descendants de-reranked run $[nd : .04, F_t : 2]$ cannot compete with other runs, even though it was the best performing run (on MAep) on the IEEE collection.

Figure 3.a shows the performance of the official runs and the baseline run on nxCG to rank%. Especially on higher rank% the two official runs and the baseline run perform good, but on lower rank% there is room for improvement.

5.2 Focused Runs - Wikipedia Evaluation

The differences between the IEEE and Wikipedia collections identified in Section 3 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 8 shows the top configurations on nxCG@5 and MAep without (de-)reranking strategies applied.

Table 8. Performance results for baseline Focused runs

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
$nd : .04, F_t : 3$	0.3099	0.2699	**0.2062**	**0.1772**	**0.0494**
$nd : .04, F_t : 4$	**0.3324**	**0.2760**	**0.2104**	**0.1766**	**0.0486**
$nd : .04, F_t : 6$	0.3311	0.2703	**0.2100**	**0.1734**	**0.0476**
$nd : .04, F_t : 10$	**0.3396**	0.2706	0.2047	0.1656	0.0462
$nd : .04, F_t : 4$	**0.3324**	**0.2760**	**0.2104**	**0.1766**	**0.0486**
$nd : .04, F_t : 5$	**0.3320**	**0.2730**	0.2029	0.1709	0.0472
Color coding:	**Best run**	**Top 3**	Top 10	Top 20	Other

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. Comparing the results of the official runs to the results of the baseline runs also shows that using structure in reranking the runs has a positive influence on MAep.

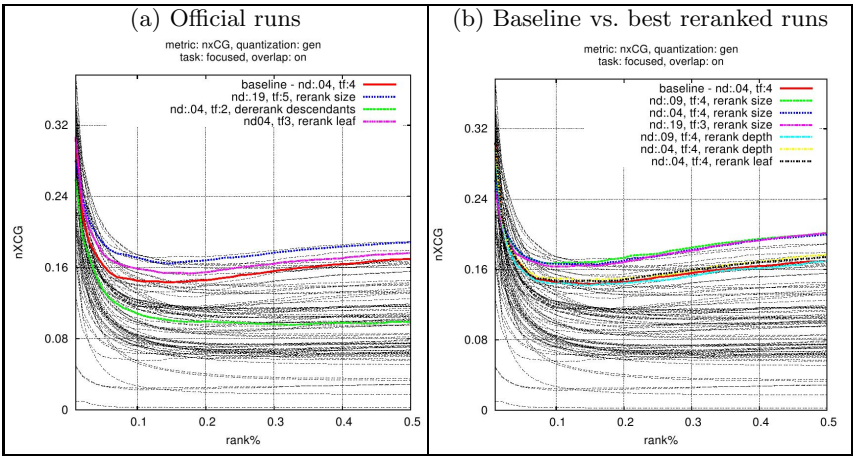


Fig. 3. Focused runs: nxCG vs. rank%

For each of the node decays [nd : .04 nd : .09 nd : .19] we selected the best configuration [F_t : 4 F_t : 4 F_t : 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, F_t : 4] are shown in Table 9.

Table 9. Performance results for (de-)reranked Focused runs

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
nd : .04, F_t : 4, *baseline*	0.3324	0.2760	0.2104	0.1766	0.0486
nd : .09, F_t : 4, *rerank size*	0.2808	0.2408	0.2105	**0.1864**	**0.0563**
nd : .04, F_t : 4, *rerank size*	0.2762	0.2375	0.2111	**0.1875**	**0.0556**
nd : .19, F_t : 3, *rerank size*	0.2522	0.2269	0.2064	**0.1811**	**0.0547**
nd : .09, F_t : 4, *rerank depth*	**0.3288**	0.2695	0.2063	0.1676	0.0486
nd : .04, F_t : 4, *rerank depth*	**0.3260**	0.2751	0.2093	0.1756	0.0497
nd : .04, F_t : 4, *dererank size*	**0.3236**	0.2724	0.2071	0.1747	0.0479
Color coding:	**Best run**	**Top 3**	Top 6	Top 12	Other

The size reranking strategy shows excellent results on MAep, with a maximum performance increase of 15.8% compared to the [nd : .04, F_t : 4] baseline run. The results also show that the best Focused baseline configuration [nd : .04] is not necessariliy the best configuration for the (de–)reranking input.

In Figure 3.b the performances of the baseline run and the best (de–)reranked runs (on MAep) are compared on nxCG to rank%. Just as could be seen in Figure 3.a performances of the various runs at higher rank% levels are good, but at lower levels the performances are less good. We can also see that the reranked runs perform less good on these lower levels than the baseline run and the submitted runs.

6 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 useful 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} measures. Improvement can however be found on the nxCG family of measures when inspecting the higher recall levels.

References

1. Fuhr, N., Lalmas, M., Malik, S., Kazai, G.: Advances in XML Information Retrieval and Evaluation. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, Springer, Heidelberg (2006)
2. Geva, S.: Gpx - gardens point xml information retrieval at inex 2004. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 211–223. Springer, Heidelberg (2005)
3. Gövert, N., Kazai, G.: Overview of the initiative for the evaluation of xml retrieval (inex) 2002. In: Fuhr, N., Gövert, N., Kazai, G., Lalmas, M. (eds.) INEX Workshop, pp. 1–17 (2002)
4. Pehcevski, J., Thom, J.A., Tahaghoghi, S.M.M.: Rmit university at inex 2005: Ad hoc track. In: Fuhr, et al, [1], pp. 306–320 (2005)
5. Sauvagnat, K., Hlaoua, L., Boughanem, M.: Xfirm at inex 2005: Ad-hoc and relevance feedback tracks. In: Fuhr, et al, [1], pp. 88–103 (2005)
6. van Zwol, R.: B3-sdr and effective use of structural hints. In: Fuhr, et al. [1], pp. 146–160
7. Weerkamp, W.: Optimising structured document retrieval using focused and relevant in context strategies. Master's thesis, Utrecht University, the Netherlands (2006)

Shallow Parsing of INEX Queries

Haïfa Zargayouna, Victor Rosas, and Sylvie Salotti

LIPN, Université Paris 13 - CNRS UMR 7030,
99 av. J.B. Clément, 93440 Villetaneuse, France
haifa.zargayouna@lipn.univ-paris13.fr
firstname.lastname@lipn.univ-paris13.fr

Abstract. This article presents the contribution of the LIPN : Laboratoire d'Informatique de Paris Nord (France) to the *NLQ2NEXI (Natural Language Queries to NEXI)* 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 combine 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 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 are 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 and structure, enriched by additional informations such as boolean operators. This analysis produces an annotated NLQ which can then be formalized in any formal language.

The remainder of this paper is organized as follows. First, we introduce the INEX campaign context. Next, we present the motivations for this research. We overview our propositions in section 4. Section 5 to 8 detail the proposed components. We comment our results and conclude with a discussion of future work.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 284–293, 2007.

2 INEX Context

In the 2006 campaign, the 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 (but they are not mandatory) to help retrieval systems to perform a better search.

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 formal queries are derived.

The proposed systems have to analyse automatically the *descriptive* 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:

$$//s_1[c_1]//\ldots//s_{k-1}[c_{k-1}]//s_k[c_k]$$

It is applied 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}, ..., which are elements of type s_1 about c_1.

Predicates c_1, ..., 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 descriptive part of the submitted topics in 2006. The queries are very diverse but we can make these distinctions between different types of queries:

- Question Answering type : This type of queries requires more precise responses to users' queries than Information Retrieval. There is no structural hint to help the system generating a formal (structured) query: for example the query 380 (*How should I prepare acorns for eating?*). This type of queries is hard to analyse to find a relevant structure in the context of the actual INEX corpora. As structural elements 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 queries is asked by a user who does not know (or does not want to use) the actual structure of the XML documents: for example the 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 structural elements that she looks for or want to be returned.

Table 1 presents statistics[1] about the percentage per type of queries in the topics proposed by participants. 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 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.

Table 1. Query types statistics

Q/A queries	CO queries	COS queries
14 → 11%	80 → 64%	31 → 25%

3 Motivations

The problem with formal XML Information Retrieval query languages is that they are difficult to use and require an intimate knowledge of documents structure from users. Formulate the user need is then limited by these two aspects.

[1] This is an approximate repartition analysis, we can consider that "what" questions can refer to the definition structure or, on the contrary, that queries containing "I want articles" do not provide structure hints.

We submitted two runs in 2006 campaign: manual and automatic. We discovered that our 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 structural elements is also important (see table 2), this can be due to the complexity of the collection schema (see section 5).

Table 2. Differences between humain 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.

Difference degree	1	2	3	4
Percentage	33,6%	38,4%	64,8%	11,2%

We propose a shallow parsing method that does not go in disambiguation processes or anaphora recognition. The offered services by the NLP interface are influenced by the formal language expressiveness. There is no need, for example, to extract boolean operators if the formal language does 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 langage.
2. Query Generation: This phase consists in taking into account the language expressiveness and syntax requirements to generate a well-formed query. This module generates NEXI queries and can be adapted to generate formal queries in other languages such as XOR [9], XQuery [10], etc.

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 structural elements. We associate these NLQ chunks to the relevant structuring elements using an external resource: taxonomy of Wikipedia structuring elements[2].

[2] For instance, build manually.

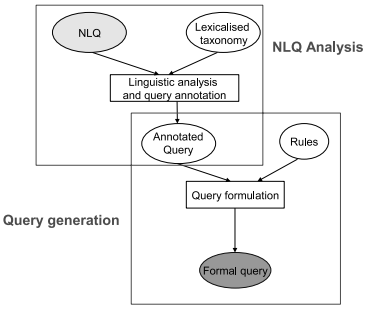


Fig. 2. Two main modules

- 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 are associated with boolean operators. The boolean operators can be associated to content and/or 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].

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 new collection based on the English Wikipedia collection [12]. The collection contains 1241 different tag names. 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 the problem of structure complexity, we define a taxonomy[3] 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/section/,*

[3] A taxonomy is the classification, or categorization, of objects. It is composed of concept related by hierarchical relations (is-a, part-of).

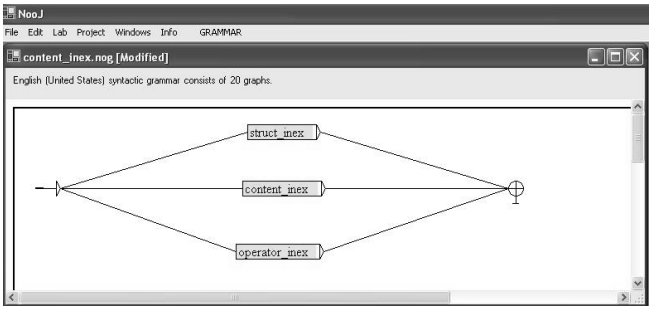


Fig. 3. Extraction of Content, structure and boolean operators with NooJ

article/body/noinclude/section/, etc. Possible paths are quite complex, without semantics we are unable to choose one. One possible solution is to associate the concept to the disjunction of all possible paths. We choose to compute an abstract path which summarizes the possible paths (i.e. an abstract path that contains common elements). The concept section, for example, is related to the path *article//section*. Concepts are also lexicalized (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 lexicalized by chapter, chapters, chapitre, etc.). The use of lexicalized taxonomy is particularly interesting in heterogeneous collections or multilingual ones.

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)*.

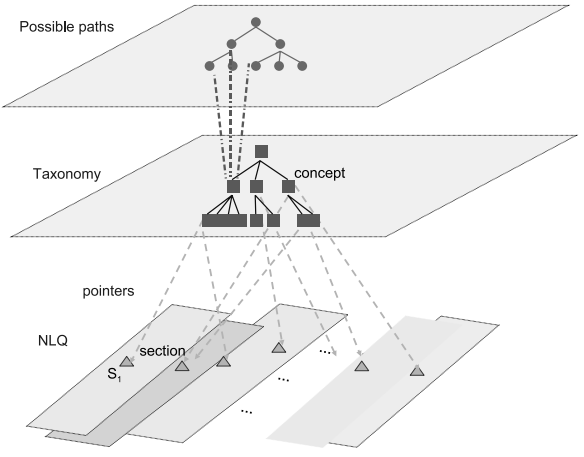


Fig. 4. The use of taxonomy : an intermediary abstraction between structural hints and elements

6 Identifying Content Terms

We have defined a set of rules that describes the grammatical construction of terms expressing users content requirements. The following syntactic rules identify "content" terms:

$Term- > number^{+}$
$Term- > adjective^{*}noun^{+}number^{*}$
$Term- > (noun|adjective|prefix)^{+}(dash)(noun|verb|adjectif)^{+}$
$Term- > "character^{+}"$

Where| is a disjunction connector, X^{*} means that X is optional and X^{+} means that there is at least one occurrence of X.

The minimal expression of content terms is a noun (common or proper noun) or a number. We prefer complex terms to simple ones, for example we choose *Vector Space Model* instead of *Vector* or *Vector Space*. Figure 5 summarizes these rules with a content graph expressed in the NooJ platform. The content graph can be considered like a content term model. The sequence of words which instantiate this model are extracted and annotated as <Content value="selected chunk">.

7 Extracting Boolean Operators

Some NLQ chunks and punctuation marks express users constraints on content or structure. In this part of the analysis we identify and extract boolean operators chunks and punctuation marks from the NLQ. The NLQs boolean operators chunks and punctuation marks considered in the analysis are:

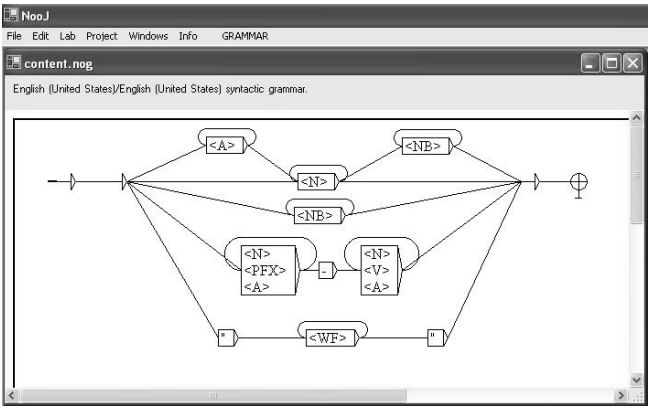


Fig. 5. "Content" Graph

- 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 NEXI Queries

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 there is 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 Runs and Results

We submitted two runs in 2006 campaign: the first run corresponds to manually generated NEXI queries, the second run corresponds to the automated translation.

Our objective is to compare the results obtained by the same search engine (namely GPX [13]) when using original participant formal NEXI expressions (the *baseline*), manually generated queries and automatically generated ones.

The runs was submitted to *GPX* and produces 24 pooled results according to the search engine (i.e. a run for each task: Thorough, Focused, All In Context, Best In Context, etc.).

The baseline almost always outperforms the other runs. This may be due to the fact that in some cases, it is more suitable to specify vague structure (i.e. //article//) than a very specific one. For example, four of our proposed queries had no results.

Our system is better ranked on Focused and Best In Context task than on Thorough task. Our manual run always outperforms the automatic one. Unfortunately, we could not interpret in detail our results. We need to analyse the overall behavior of our method and its efficiency for the different types of queries (QA, CO and COS). More experimental studies are necessary.

10 Conclusion and Future Work

Formulating NEXI queries to express their information needs is a difficult task for casual users.

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 do not meet the descriptive part[4].

We proposed a shallow parsing method to extract main components of NL queries. Considerable efforts need to be done to digest the vast number of elements introduced in this year's INEX DTD, in order to identify correctly the structural hints and to make sense of their implications for future adaptations of our system. We are actually adapting our propositions to a specialized corpora in the medical domain. We believe that "Semantically" tagged documents enable more fine-grained translations.

References

1. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: Natural language interfaces to databases - an introduction. Natural Language Engineering 1(1), 29–81 (1995)
2. Hulgeri, A., Bhalotia, G., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword search in databases. In: IEEE Data Engineering Bulletin. pp. 22–32 (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. pp. 149–157 (2003)
4. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: A semantic search engine for XML. In: VLDB. pp. 45–56 (2003)
5. Li, Y., Yang, H., Jagadish, H.: Constructing a generic natural language interface for an XML database. In: International Conference on Extending Database Technology EDBT, LNCS 3896. pp. 737–754 (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. pp. 373–387 (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. pp. 358–372 (2005)
8. O'Keefe, R.A., Trotman, A.: The simplest query language that could possibly work. In: Proceedings of the second Workshop of the INitiative for the Evaluation of XML retrieval. pp. 167–174 (2003)
9. Geva, S., Hassler, M., Tannier, X.: XOR - XML Oriented Retrieval Language. In: Proceedings of SIGIR 2006, XML Element Retrieval Methodology Workshop. pp. 5–12 (2006)
10. Boag, S., Chamberlin, D., Fernández, M., Florescu, D., Robie, J., Siméon, 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. pp. 10–11 (2005)
12. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. SIGIR Forum (2006)
13. Geva, S.: GPX - Gardens Point XML IR at INEX 2005. In: Advances in XML Information Retrieval: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval. pp. 240–253 (2006)

[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.

Using Rich Document Representation in XML Information Retrieval

Fahimeh Raja[1], Mostafa Keikha[1], Masued Rahgozar[1], and Farhad Oroumchian[2]

[1] 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
[2] The College of IT, University of Wollongong in Dubai
FarhadO@uowdubai.ac.ae

Abstract. In this paper we present a method of document representation called Rich Document Representation (RDR) to build XML retrieval engines with high specificity. RDR is a form of document representation that utilizes single words, phrases, logical terms and logical statements for representing documents. The 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 the Content Only queries of the given collection. The results have been very weak but a failure analysis has revealed that it has been caused by an error in document processing which has produced inconsistent IDs and caused a mismatch between the IDs assigned to document elements such as single terms, phrases and logical terms. However similar experiment on INEX2004 collection yielded very good precision on high specificity task with s3e123 quantization.

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 to 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] [2].

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 terms where stemmed single words are used as a representation of a document [3]. 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

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 294–301, 2007.

contribution largely depends on the type of the system and weighting scheme used [4]. Adding phrases to a single term representation in vector space system with a good weighting such as Lnu.ltu will only add 3-5% [5] to precision. However, in a system with weaker weighting the contribution of the phrases could be as much as 10% [5].

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 [6], [7]. However in this paper we are using 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 [8].

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 [9]. 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 [10]. Previously, it was proven that RDR is better representation for clustering than single words in the second stage of a two stage retrieval system [11].

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 of 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 uses all the statements representing the documents in the collection to build a single knowledge base that describes a possible world which includes the documents. In such a space, statements of different documents could complement each other and/or provide more information. In that space, PLIR uses reasoning in order to guess the relevancy of documents to queries. PLIR's strength comes from representing document content as logical statements and relating them together through ISA or other types of relationships and reasoning with these statements and relationships. ISA is an important relationship which represents Type-of or Kind-of relationships and creates hierarchies in the knowledge base that can be used by inferences. For example Excel is a Type_of spreadsheet so if a user needs information about spreadsheets he/she would be interested in Excel which is a type of spreadsheet. Another feature of PLIR is a local weighting called Dominance which is not dependent on Inverse Document Frequency. Dominance weights the strength of relationships between the concepts and/or between documents and concepts. Its formulation does not require document frequency therefore its calculation is independent of the size of collection or number of documents that have any particular concept. In processing large collections this would be advantageous because it means at any stage of the processing documents all the documents processed so far have their complete weight and can be added to the collection and index and used in retrieval. All these relationships and logical statements are extracted automatically from the text. In a normal vector space model, there is no reasoning and ISA relations are not useful, so the only use of RDR would be providing a deeper representation for the text. RDR has been used in documents clustering as well. It has been tested with several clustering methods and different similarity measures and it has outperformed single word and phrase representations consistently [11]. This paper explores RDR's application in retrieving more specific XML elements.

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 a Part of Speech (POS) tagger.
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', may signify a relationship between two noun phrases that it connects.

Table 1 shows a few sentence fragments and their equivalent 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} ISA (island, Kish)	Logical Statement

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.

3 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 the regular keyword matching by the 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 is to treat them as phrases and weight them accordingly. However, PLIR uses Dominance scheme for weighting them. In this work we use *"tf.idf"* in our weighting model. We apply the following formula for weighting each index term of queries:

$$w(t,q) = \alpha * termFreq(t,q) * idf(t) * nf(q) \tag{1}$$

Where:

$$\alpha = \frac{2}{3} \text{ , for terms with "_" before them in the query} \tag{2}$$

$$= \frac{4}{3} \text{ , for terms with "+" before them in the query}$$

$$= 1 \text{ , otherwise}$$

$$idf(t) = \ln\left(\frac{N}{n(t)}\right) \tag{3}$$

$$nf(q) = \frac{1}{lenq} \tag{4}$$

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:

$$w(t,e) = tf(t,e) * idf(t) * nf(e) * childEffec\ t(e) \tag{5}$$

Where:

$$tf(t,e) = \frac{(1 + \log(termFreq(t,e))}{(1 + \log(avg(termFreq(e))} \tag{6}$$

$$idf(t) = \ln(\frac{N}{n(t)}) \tag{7}$$

$$nf(e) = \frac{1}{\sqrt{len(e)}} \tag{8}$$

$$childEffect(t,e) = \frac{\#\text{of sublements of e with term t}}{\#\text{of sublements of e}} \tag{9}$$

termFreq(t,e): frequency of occurrence of term t within the element e
avg(termFreq(e)): average term frequency in element e
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
nf(e): normalization factor of element e
len(e): element length which is equal to number of terms in the element
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:

$$V_{Single\_Terms} = (w(a,q), w(b,q), w(c,q))$$
$$V_{Phrases} = (w(a\_b,q))$$
$$V_{Logical\_Terms} = (w(a\_b(c),q))$$

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 [12]. 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.

4 Results

Our implemented system has participated in INEX 2006 [13]. Our results are very weak but in a failure analysis it was discovered that we have made mistake in preprocessing of the collection.

In the preprocessing phase of our experiments we assumed that the content of the XML documents is placed only in paragraphs. Therefore, if we have text in elements other than paragraphs, we created a child for it, called *"VirtualParagraph (VP)",* and assign an ID to it. This assumption and preprocessing has led 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 the element does not contain any logical term and statement. Therefore there would be mismatch between the counters that assign IDs to single words and phrases and those of the logical terms. This was a pure programming bug and avoidable but nevertheless had a disastrous effect on the performance of the system. A major problem we faced in this round of the INEX was the size of the collection. The

collection size for INEX2006 is much larger than previous years and it exceeds the processing power of the shared server that was allocated to this research for a limited time. Therefore we spent a lot of time on trying to optimize the programs in order to be able to process the entire collection. The size of collection also contributed to late detection of the error in preprocessing the collection.

We resolved the problem in our system and tried to process the collection correctly and get the results on the INEX 2006 collection. However after INEX, the server assigned to this project was allocated to another research project and we were not able to finish the document processing phase.

In another set of experiments we used INEX 2004 test collection with the correct pre-processing and because of the smaller size of the collection we were able to finish the processing and get results. The results show that RDR yields in better precision with the quantization in which the specificity is more important, especially s3e321 quantization; that is RDR improves specificity of the elements returned to the user which is one of the goals of XML information retrieval. This is consistent with other tests and reported results by others too. In all previous reported experiments, PLIR and RDR have produced high precision but low recall systems. RDR is a precision oriented instrument and is not suitable for recall oriented tasks. RDR was average or below average on recall oriented measures on INEX2004 data. Figure 1 depicts the performance of the RDR representation in CO tasks with s3e123 and so quantizations.

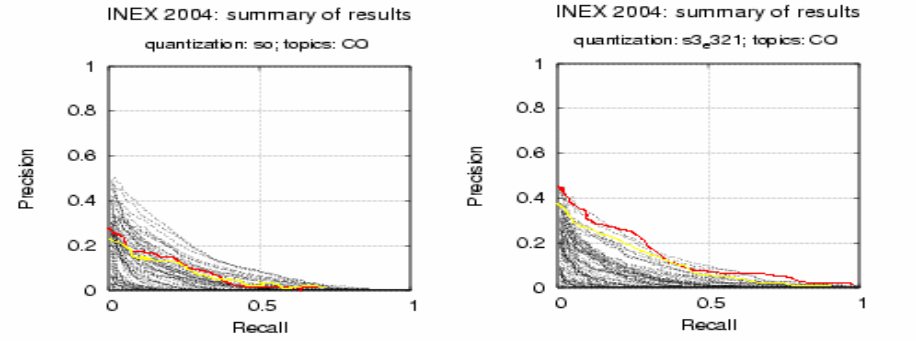


Fig. 1. The result of the RDR representation on INEX2004 collection on CO task

5 Conclusion

In this paper we presented 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 the text by using linguistic clues. The simplest clue is the proposition. This representation is called RDR (Rich Document Representation).

We have experimented with INEX2004 test collection; the results indicate that the use of richer features such as logical terms or statements for XML information retrieval tends to produce more focused retrieval. Therefore it is a suitable document representation when users need only a few more specific references and are more interested in precision than recall.

However, when we conducted our experiments using INEX 2006 test collection (a much larger collection than INEX 2004 test collection), the performance of our runs has been very poor due to preprocessing/indexing errors. However because of the larger size of the INEX2006 collection and limited processing resources available to this project at the time we were not able to finish re-processing of the INEX2006 collection and improve our results.

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 future, we are going to conduct more tests on different domains and collections and to improve on document representation by using automatic relevance feedback and machine learning methods. We are hoping to improve our information extraction method and produce better and more reliable logical statements which will result in even higher precision.

References

1. Oroumchian, F., Karimzadegan, M., Habibi, J.: XML Information Retrieval by Means of Plausible Inferences. In: RASC 2004, 5th International Conference on Recent Advances in Soft Computing, RASC, Nottingham, United Kingdom pp. 542–547 (2004)
2. Fuhr, N., Gövert, N., Kazai, G., Lalmas, M.: INEX: INitiative for the Evaluation of XML retrieval. In: Baeza-Yates, R., Fuhr, N., Maarek, Y. S. (eds.) Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval (2002)
3. Salton, G., Allan, J., Buckley, C.: Approaches to passage retrieval in full text information systems. In: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval pp. 49–58 (1993)
4. Fox, E.A.: Lexical relations: enhancing effectiveness of information retrieval systems. SIGIR Newsletter 15(3), 5–36 (1981)
5. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval pp. 21–29 (1996)
6. Evens, M., Vandendrope, J., Wang, Y.-C.: Lexical Relations in Information Retrieval, Human and Machines. In: Proceedings of the 4th Delware Symposium on Language Studies, Albex Norwood N.J (1985)
7. Cohen, P.R., Kjeldsen, R.: Information retrieval by constrained spreading activation in semantic networks. 23, pp. 255–268. Pergamon Press, Inc, Tarrytown, NY, USA (1987)
8. Collins, A., Michalski, R.: The Logic of Plausible reasoning A Core Theory. Cognitive Science 13, 1–49 (1989)
9. VAN RIJSBERGEN, C. J.: Logics for Information Retrieval, AL4T 88, Rome (March 1988)
10. Oroumchian, F., Oddy, R.N.: An application of plausible reasoning to information retrieval. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Switzerland, pp.18-22 (August 1996)
11. Oroumchian, F., Jalali, A.: Rich document representation for document clustering. RIAO 2004 Conference. In: Proceedings Coupling approaches, coupling media and coupling languages for information retrieval, Le Centre de Hautes Etudes Internationalies d'Informatique Documenataire - C.I.D., France pp. 1–9 (2004)
12. Greengrass, E.: Information Retrieval: A Survey. DOD Technical Report TR-R52-008-001 (2000)
13. accessed 11th of February 2007 http://inex.is.informatik.uni-duisburg.de/2006/

NLPX at INEX 2006

Alan Woodley and 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 needs than traditional IR systems by returning results lower than the document level. In order to use XML-IR systems users must encapsulate their structural and content information needs in a structured query. 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 specify 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 2006 INEX track where NLPX was the best performing participant in the Thorough and Focused tasks.

1 Introduction

Information retrieval (IR) systems respond to users' queries with a ranked list of relevant results. In traditional IR systems these results are whole documents, but since XML documents separate content and structure XML-IR systems are able to return highly specific results that are lower than the document level. But, if users are going to take advantage of this capability in an operational (and possibly even in a laboratory setting) then they require an interface that is powerful enough to express their content and structural requirements, yet user-friendly enough that they can express their requirements intuitively.

Historically, XML-IR systems have used two types of interfaces: keyword based and formal query language based. Keyword based systems are user-friendly, but are unable to express the structural needs of the user. In comparison, formal query language-based interfaces are able to express users' structural needs (as well as their content needs) but are impractical for operational use since they too difficult to use — especially for casual users [7,14] — and are bound to the physical structure of the document. The purpose of INEX's natural language processing (NLP) track is to investigate a third interface option that encapsulates users' content and structural needs intuitively, in a natural language query (NLQ). Participants in the NLP track

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 302–311, 2007.

develop systems that translate natural languages queries to formal language queries (NEXI). The translated queries are executed on a single backend retrieval system (GPX) and their retrieval performance is compared amongst them themselves and with a baseline system consisting of manually constructed NEXI queries. The NLP task uses the same topics, documents and relevance assessments as the Ad-hoc task to enable comparison between participants in the NLP tracks as well as with participants in the Ad-hoc track (although, NLP participants process the topics' *description* elements rather than their *title* elements).

QUT has participated in the NLP task since its inception with its natural language interface NLPX. Here, we discuss the changes made to NLPX since last year and discuss its performance at this year's INEX. The changes subsume two parts. First, we outline the special connotations and templates added to NLPX that allow it to process a greater range of structured NLQs. Second, we discuss the process of porting NLPX from the IEEE collection to the Wikipedia collection. We also present the results from this the 2006 NLP track where NLPX was the best performing participant in the Thorough and Focused tasks.

2 Motivation

We have already outlined the motivations for an XML-IR natural language interface in our previous work [12,13]; however, for completeness we include them here. The motivations stem from weakness with current XML-IR interfaces.

The reason that keywords are unsuitable for XML-IR is that they can only contain users' content information need and not their structural information need. It has long been assumed that a user's information need will be better fulfilled if they specify their structural need, that is, the location within the document that contains their desired content. However, recent research has shown that this assumption may not be correct when considered over a number of queries and retrieval algorithms [9]. However, it is unclear if this outcome is because the specification of structural requirements does not assist retrieval at all or for other reasons (such as users not being able to correctly specify structural requirements, lack of meaningful structure within INEX"s previous IEEE document collection or an inability for existing XML-IR systems to handle structural requirements).

While formal languages are able to fully capture users' structural requirements, they too have problems. First, formal query languages are too difficult for both expert and casual users to correctly express their structural and content information needs. The difficulty that experts have in using formal languages has been recorded in INEX's use of XPath and NEXI at the 2003 and 2004 Workshops were 63% and 12% of proposed queries were either syntactically or semantically incorrect [7]. Therefore, if experts in the field of structured information retrieval are unable to correctly use complex query languages, one cannot expect a casual user to do so. This theory was verified by researchers in INEX's interactive track [14] who observed the difficulty that casual users had in formatting formal queries. However, we feel that users would be able to intuitively express their information need in a natural language.

A second problem with formal query languages is that they are too tightly bound to the physical structure of documents; and therefore, users require an intimate knowledge of the documents' composition in order to fully express their structural requirements. So, in order for users to retrieve information from abstracts, bodies or bibliographies, they will need to know the actual names of those tags in a collection (for instance: *abs*, *bdy*, and *bib*). While this information may be obtained from a document's DTD or Schema there are situations where the proprietor of the collection does not wish users to have access to those files. Or, in the case of a heterogeneous collection, a single tag can have multiple names (for example: abstract could be named *abs*, *a*, or *abstract*). This is a problem identified by participants in the INEX 2004 heterogenous track who have proposed the use of metatags to map between collections [6] and extensions to NEXI [10] to handle multiple tag names. Naturally, neither of these solutions are trivial, which is why INEX has multiple tracks (heterogenous and document mining) devoted to investigating this problem. In contrast, structural requirements in NLQs are inherently expressed at a higher conceptual level, allowing the underlying document's structure to be completely hidden from users, although NEXI could also be extended to handle conceptual tag names.

3 Previous Work by Authors

This paper expands on the previous work of the authors [12,13]. We submitted our system, NLPX, to INEX's the 2004 and 2005 Natural Language Processing track where it has performed strongly. INEX's NLP track used the same topics and assessments as its Ad-hoc track; however, participating systems used a natural language query as input, rather than a formal language (NEXI) query. Examples of both query types are expressed in Figure 1. Note that the query actually contains two information requests, first, for sections about compression, and second, for articles about information retrieval. However, the user only wants to receive results matching the first request. We refer to the former as returned requests/results and the latter as support requests/results.

NEXI: //article[about(.,'information retrieval')] //sec[about(./, compression)]

NLQ: Find sections of articles about image and text compression in articles about efficient information retrieval

Fig. 1. A NEXI and Natural Language Query

As with last year, the goal of the 2006 NLP participants was to produce a natural language interface that translated NLQs to into NEXI queries. The translated queries were executed by a single backend system (GPX) and the retrieval performance of translated queries was recorded as if it were a standard Ad-hoc system. Participants in the NLP track compare their retrieval performance amongst each other as well as a baseline system that uses manually formed NEXI expressions (that is the original *title* tags) as input. NLPX's translation process involves four steps that derived syntactic

and semantic information from the natural language query (NLQ). We refer to these four steps as the NLPX framework and outline them below:

1. First, NLPX tags words in the NLQ as either a special connotation or by their part of speech. Special connotations are words of implied semantic significance. Words corresponding to special connotations can either be hard-coded into the system and matched to query words by a dictionary lookup or tagged using a modified Brill Tagger that also considers a word's context when tagging. Non-connotations are tagged by their part of speech (such as noun, verb, conjunction) via a Brill Tagger [2].
2. Second, words are grouped together into phrases using a process called Chunking [1]. The reason that NLPX recognises Chunks is to reduce ambiguity and to facilitate further content analysis at a later stage. There are three main types of chunks NLPX recognises: Instructions (for example "I want to retrieve"), Structures (for example "sections of articles") and Content (for example "information retrieval").
3. Third, NLPX matches the tagged NLQs to query templates. The templates were derived from the inspection of previous INEX queries. Since the NLQs occurred in shallow context they required only a few templates, significantly less than if one wished to capture natural language as a whole. Each template corresponded to an information request. Each request had three attributes: Content, a list of terms/phrases expressing content requirements, Structure, a logical XPath expression expressing structural requirements, and an Instruction, "R" for return requests, and "S" otherwise.
4. Finally, the requests are merged together and output in NEXI format. Return requests are output in the form **A[about(.,C)]** where **A** is the request's structural attribute and **C** is the request's content attribute. When all return requests are processed, support requests were inserted. The insert position was located by comparing the structural attributes of return and support requests and by finding their longest shared descendant. The output of support requests had the form **D[about(E,F)]** where **D** is the longest matching string, **E** is the remainder of the support's structural attribute and **F** is the support's content attribute. Note, that while NLPX outputs NEXI queries this step has been modulated so that NLPX could be extended to include any number of formal query languages.

4 Improvements

As usual, we have made several improvements to NLPX since last year's participation. However, the number of improvements was less than in previous years that may indicate the research is reaching a plateau. The major change this year was porting NLPX to the new Wikipedia collection. Fortunately, this was not an extraneous task since the only change that was required was in the third step. Other changes made this year was the addition of strengtheners, that is content words signalled by the user as being of high importance, and a constraint added by the system to ensure that support requests are not placed in the last about clause even if they have the same return type as return requests (as defined by NEXI standards).

5 System Backend

Once the NLQ was tagged, chunked and matched to templates it was transformed into a NEXI query using the existing NLPX system. This is a two stage process. First we expanded the content of the query, by deriving phrases based on its lexical properties, such as noun phrases that include adjectives and participles. Then we formatted a NEXI query based upon its instruction, structure and content values. We passed the NEXI query to our existing GPX system for processing as if it were a standard Ad-hoc query. To produce its results list GPX collects leaf elements from its index and dynamically creates their ancestors. GPX's ranking scheme rewards leaf elements with specific rather than common terms, and elements that contain phrases. It also rewards ancestors with multiple relevant children rather than a single relevant child. A more comprehensive description of GPX can be found in our accompanying paper as well as earlier work [5].

6 Results

Here we present the performance results from NLPX. The results are split into two parts. The first part discusses how well NLPX was able to translate natural language queries to NEXI. The second part presents the retrieval performance of the translated queries in comparison with the original NEXI queries.

6.1 Translation Performance

The *description* elements of the 125 INEX Ad-hoc topics were translated in NEXI format by NLPX. These translations were then manually compared with each topic's *description* and *castitle* elements to test their accuracy. Furthermore, a comparison was made between each *castitle* and *description* elements themselves, to test how faithfully the topic's originator was able to express their information need in natural and formal language. The results of these comparisons are presented in Table 1.

Table 1. Comparison between translation, description and castitle

	Description	**Castitle**
Translation	0.704	0.352
Description		0.408

The overall accuracy rate between the translation and *descriptions* was high (70.4 per cent), however the similarity between the translations and the original *castitles* was much lower (35.2 percent). This is largely because the similarity between the *descriptions* and *castitles* were also low (40.8 percent) Examples of successful translations were seen in INEX topics 292 and 311 as shown in Figures 2 and 3.

Description: I want figures containing Renaissance paintings of Flemish or Italian artists, but not French or German ones
Translation: //article//section//(figure|caption)[about(.,"Renaissance paintings" Renaissance paintings Flemish "Italian artists" Italian artists "-German -ones" -German -ones)]
Title: //article//figure[about(., Renaissance painting Italian Flemish -French -German)]

Fig. 2. INEX Topic 292

Description: Find sections about global warming cause and effects.
Translation: //article//section[about(.,"global warming cause" global warming cause "global cause" "warming cause" effects)]
Title: //section[about(., global warming cause and effects)]

Fig. 3. INEX Topic 311

Description: Find articles about Novikov self-consistency principle that contain a section about time travel.
Translation: //article[about(.,"Novikov self consistency principle" "Novikov self-consistency principle" "self consistency" self consistency Novikov self-consistency principle "Novikov principle") OR about(.,"time travel" time travel)]
Title: //article[about(., Novikov self-consistency principle) and about(.//section, time travel)]

Fig. 4. INEX Topic 310

Description: Retrieve sections of articles about the use of ontologies in information retrieval such as semantic indexing.
Translation: //article[about(.//*,retrieval "semantic indexing" semantic indexing)]//section[about(.,ontologies)]
Title: //article//section[about(.,ontologies information retrieval semantic indexing)]

Fig. 5. INEX Topic 358

However, other times NLPX was unsuccessful in translating the *description*. Examples of this occurred in INEX topics 310 and 358, shown in Figures 4 and 5. For Topic 310 NLPX was unable to determine that a second information requestregarding sections was made by the user, while for Topic 358 NLPX was unable to recognise the term “information” as content-bearing rather than as a structural request.

Furthermore, a lot of times the *castitle* (59.2) and *description* did not match, particularly in terms of structural requests. This is worrying since the *castitle* and *description* should be faithful representations of the user's information need. For instance, in INEX Topic 402, shown in Figure 6, the *description* asks for ***information*** about capitals of European counties, presumably this means that the structural constraints are arbitrary and therefore a wildcard element (//*) should be specified as the NEXI path (essentially turning the topic into a Content Only query). However, the *title* specially specifies ***articles*** as the NEXI path. Another example occurs in INEX topic 400, shown in Figure 7, where the *description* specifically requests ***documents*** but the *title* specifies for ***sections*** in the NEXI path. Therefore, in these types of instances it would be impossible for the translated query to match both the *description* and *title*.

Description: Find information on capitals of European countries.
Translation: //article//*[about(.,capitals "European countries" European countries)]
Title: //article[about(.,country european)]//section[about(.,capital)]

Fig 6. INEX Topic 402

Description: Find documents about countries that had non-violent revolutions
Translation: //article[about(.,countries "non violent revolutions" "non-violent revolutions" "non violent" non violent non-violent revolutions)]
Title: //article[about(., country revolutions)]//section[about(., "non violent")]

Fig 7. INEX Topic 400

6.2 Retrieval Performance

As with previous years the translated queries from NLPX and the other NLP participants were executed by a single backend system GPX. This allowed for the same analysis of the NLP participants retrieval performance as is seen in traditional information retrieval. An additional "baseline" consisting of the original *castile* queries was also executed by GPX, allowing for a comparison to be made between automatic and manually created NEXI statements. Furthermore, since the same INEX topics and relevance assessments are used by the NLP and Ad-hoc tracks cross-track comparisons are valid. At the time of publication results for the Thorough, Focussed and Best-In-Context tasks were available. This section begins with a detailed analysis of NLPX's performance in the Thorough task before presenting an outline of NLPX's performance across all tasks. Also, note that the set of relevance judgements used to evaluate these systems included all relevant elements, including those removed form later relevance assessments for being too small (mainly link elements).

Table 2. MAep Results of the 2006 INEX Thorough Task

Translation	Root Orientated	Leaf Orientated
Baseline (NEXI)	0.0323	0.0284
NLPX06	0.0298	0.0251
NLPX05	0.0279	0.0229
Robert Gordon	0.0235	0.0171
Robert Gordon 2	0.0224	0.0159
Exoles des Mines de Saint-Erienne	0.0235	0.0205
Exoles des Mines de Saint-Erienne XOR	0.0231	0.0197

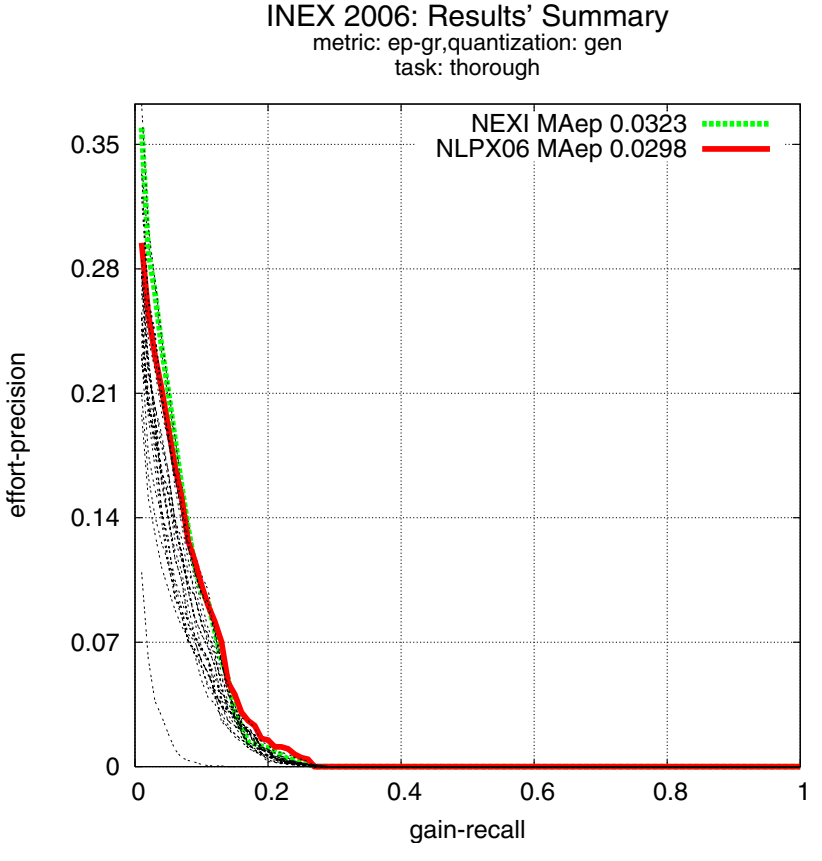


Fig. 8. INEX 2006 NLP Through ep-gr graph

As with previous years, the aim of INEX's Thorough task was to retrieve as many relevant information items as possible, regardless of how many "overlapping" items were returned. The metric used to evaluate the Thorough task is Mean Average effort-precision (MAep), analogous to traditional mean average precision. NLPX produced two translations for each of the tasks: one produced by the current implementation of NLPX (NLPX06) and one produced by a previous implementation of NLPX (NLPX05). GPX produced two submissions for each of the translations , one that one that favoured leaf elements and one that favoured root elements. The MAep results of each of the submissions are presented in Table 2. The results show that NLPX performed very strongly, outperforming alternative NLP approaches. Furthermore, it performed comparable to the Baseline achieving a ratio of around 0.8. This is highlighted in the effort-precision gain-recall graph presented in Figure 8 where the NLPX and Baseline submissions produce similar plots.

Similar results were achieved by NLPX across all the tasks. Table 3 shows the retrieval performance of the best performing NLPX, Baseline and Ad-hoc submission for the Thorough, Focussed and Best-in-Context tasks. The table contains both the retrieval score (using the appropriate metric) and the pseudo-rank of the submission if it was submitted to the Ad-hoc track. Once again, NLPX performs strongly in the Thorough task, and while its rank is affected severely in the other two tasks, its ratio to the Baseline remains fairly consistent. It must be noticed however, that these results are derived from the original INEX pools that rewarded link elements very highly. However, GPX removed all links from each of the submissions (including all NLP participants). Hence, different scores would be recorded if the second INEX pool was used that scored link elements as too small to be relevant. In this scenario, the retrieval scores and pseudo-ranks of the Baseline and NLPX submission would probably increase, however the ratio between the scores should not be greatly affected.

Table 3. Retrieval performance across all tasks

Task **Metric**	**Thorough** **Maep**	**Focussed Overlap On nxCG[50]**	**Focussed Overlap Off nxCG[50]**	**Best-In-Context BEPD At A=100.0**	**Best-In-Context EPRUM At A=100.0**
Best NLPX	0.0298 (15)	0.139 (47)	0.1989 (21)	0.6551 (37)	0.1974 (44)
NEXI Baseline	0.033 (10)	0.173 (20)	0.2405 (10)	0.6895 (26)	0.2243 (33)
Best Adhoc	0.0384 (1)	0.2265 (1)	0.2802 (1)	0.7983 (1)	0.3146 (1)

7 Conclusion

This paper presented the results of NLPX's participation in INEX 2006. This is the third year that NLPX has participated in INEX's NLP task, each year improving its performance. This year it was able to correctly translate a majority of topics from natural language to formal language. It outperformed the alternative NLP approaches

and was comparable to a baseline formal language system. These results validate the potential of natural language queries as alternative to formal language queries in the domain of XML retrieval.

References

1. Abney, S.: Parsing by Chunks. In: Principle-Based Parsing, Kluwer Academic Publishers, Boston, MA (1991)
2. Brill, E.: A Simple Rule-Based Part of Speech Tagger. In: Proceedings of the Third Conference on Applied Computational Linguistics (ACL), Trento, Italy pp. 152–155 (1992)
3. Clark J., DeRose, S.: XML Path Language XPath Version 1.0. W3C Recommendation, The World Wide Web Consortium (November 1999) available at http://www.w3.org/TR/xpath.
4. Fox, C.: Lexical Analysis and Stoplists. In: Frankes, W.B., Baeza-Yates, R. (eds.) Information Retrieval: Data Structures and Algorithms. 7, pp. 102–130. Prentice-Hall, Upper Saddle River (1992)
5. Geva, S.: GPX - Gardens Point XML Information Retrieval INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 221–222. Springer, Heidelberg (2005)
6. Larson, R.: XML Element Retrieval and Heterogenous Retrieval. In Pursuit of the Impossible? In: Proceedings of INEX 2005 Workshop on Element Retrieval Methodology, Glasgow, Scotland pp. 38-41 (2005)
7. O'Keefe, R., Trotman, A.: The Simplest Query Language That Could Possibly Work, In: Fuhr N., Malik, S. (eds.) INEX 2003 Workshop Proceedings. Dagstuhl, Germany pp. 167–174 (2003)
8. Ramshaw, L., Marcus, M.: Text Chunking Using Transformation-Based Learning, In: Proceedings of the Third Workshop on Very Large Corpora pp. 82-94 (1995)
9. Trotman, A., Lamas, M.: Why structural hints in queries do not help XML-retrieval, In: Efthimiadis E. N., Dumais S. T., Hawking, D., Järvelin, K. (eds.) SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, pp. 711–712 (2006)
10. Trotman, A., Sigurbjörnsson, B.: NEXI: Now and Next, In: Fuhr, N. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 410–423. Springer, Heidelberg (2005)
11. Woodley, A., Geva, S.: NLPX: An XML-IR System with a Natural Language Interface, In: Bruza, P., Moffat, A., Turpin, A (eds.) Proceedings of the Australasian Document Computing Symposium, Melbourne, Australia pp. 71–74 (2004)
12. Woodley, A., Geva, S.: NLPX at INEX 2004, In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 393–406. Springer, Heidelberg (2005)
13. Woodley, A., Geva, S.: NLPX at INEX 2005, In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 358–372. Springer, Heidelberg (2006)
14. van Zowl, R., Bass, J., van Oostendorp, H., Wiering, F.: Query Formulation for XML Retrieval with Bricks. In: Fuhr, N., Lamas, M., Trotman, A. (eds.) In Proceedings of INEX 2005 Workshop on Element Retrieval Methodology, Glasgow, Scotland pp. 75-83 (2005)

The Heterogeneous Collection Track at INEX 2006

Ingo Frommholz[1] and Ray Larson[2]

[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 of 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 in terms of 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:

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 312–317, 2007.

- 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 its 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 in terms of time, communication costs, and processing.

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?
- For content-and-structure queries, 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?
- For all types of queries, 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.

Table 1. Components of the heterogeneous collection. *Element counts estimated for large collections.*

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

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 Chesire II[1] system. Converting the subcollections into a unified internal representation turned out to be a very time-consuming task as new collections had to be incorporated.

Appendix A shows the topic DTD used. Besides keywords and titles, also content-and-structure titles (`<castitle>` in our DTD) were given for most topics in order to make them suitable for the CAS tasks. Content-and-structure titles do not only contain keywords, but also a NEXI [1] path expression for the desired structural elements. For instance, the `castitle` expression

```
//article[about(.,user interface)]//section[about(.,design)]
```

requests sections about design in articles about user interfaces. 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).

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 for both CAS tasks 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). The system should thus return only the title and the summary of relevant documents, but might base the relevance decision on, e.g., the whole document fulltext.

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 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.

[1] `http://cheshire.berkeley.edu/`

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 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 have heterogeneous syntax, semantics and document genres. We also set up a number of test topics for evaluation. We have, therefore, laid the foundations for a new heterogeneous track which may now concentrate on submitting runs, creating a pooled result set and providing relevance assessments, and these in turn will be used to evaluate the submitted runs.

Acknowledgement

Special thanks go to Miro Lehtonen for providing us with the IDEAlliance collection, and to Saadia Malik and Benjamin Piwowarski for technical support.

Reference

1. Trotman, A., Sigurbjornsson, B.: Narrowed extended XPath I (NEXI). In: Fuhr, N., Lalmas, M., Malik, S., Szlavik, Z. (eds.) Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers, vol. 3493. Springer-Verlag GmbH, (May 2005) http://www.springeronline.com/3-540-26166-4

A Topic DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY   lt      "&#38;#60;">
<!ENTITY   gt      "&#62;">
<!ENTITY   amp     "&#38;#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.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 318–330, 2007.

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 $P(R \mid Q, D)$ is calculated and the documents or components are presented to the user ranked in order of decreasing values of that probability. To avoid invalid probability values, the usual calculation of $P(R \mid Q, D)$ uses the "log odds" of relevance given a set of S statistics derived from the query and database, such that:

$$\log O(R|C,Q) = log\frac{p(R|C,Q)}{1-p(R|C,Q)} = log\frac{p(R|C,Q)}{p(\overline{R}|C,Q)} \tag{1}$$

$$= b_0 + b_1 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\frac{qtf_i}{ql+35} \tag{2}$$

$$+ b_2 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\log\frac{tf_i}{cl+80} \tag{3}$$

$$- b_3 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\log\frac{ctf_i}{N_t} \tag{4}$$

$$+ b_4 * |Q_c| \tag{5}$$

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,

$p(\overline{R}|C,Q)$ the probability that document component C is not relevant to query Q, (which is 1.0 - $p(R|C,Q)$)

$|Q_c|$ is the number of matching terms between a document component and a query,

qtf_i is the within-query frequency of the ith matching term,

tf_i is the within-document frequency of the ith matching term,

ctf_i is the occurrence frequency in a collection of the ith matching term (i.e., the number of time the term occurs in the entire collection, for a given component type),

ql is query length (i.e., number of terms in a query). This is simply $|Q|$, the number of terms in a query, for non-feedback situations but takes on otther values for feedback,
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).
b_i are the coefficients obtained though 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, b_k, used for our official runs are the same as those described by Chen[3]. These were: $b_0 = -3.51$, $b_1 = 37.4$, $b_2 = 0.330$, $b_3 = 0.1937$ and $b_4 = 0.0929$. 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:

$$\begin{aligned}\log O(R \mid Q, C) = \; & b_0 + \left(b_1 \cdot \left(\frac{1}{|Q_c|} \sum_{j=1}^{|Q_c|} \log qtf_j \right) \right) \\ & + \left(b_2 \cdot \sqrt{|Q|} \right) \\ & + \left(b_3 \cdot \left(\frac{1}{|Q_c|} \sum_{j=1}^{|Q_c|} \log tf_j \right) \right) \\ & + \left(b_4 \cdot \sqrt{cl} \right) \\ & + \left(b_5 \cdot \left(\frac{1}{|Q_c|} \sum_{j=1}^{|Q_c|} \log \frac{N - n_{t_j}}{n_{t_j}} \right) \right) \\ & + (b_6 \cdot \log |Q_d|)\end{aligned} \tag{6}$$

Where:

Q is a query containing terms T,
$|Q|$ is the total number of terms in Q,
$|Q_c|$ is the number of terms in Q that also occur in the document component,
tf_j is the frequency of the jth term in a specific document component,
qtf_j is the frequency of the jth term in Q,
n_{t_j} is the number of components (of a given type) containing the jth term,
cl is the document component length measured in bytes.
N is the number of components of a given type in the collection.
b_i are the coefficients obtained though the regression analysis.

This equation, used in estimating the probability of relevance 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 coeffients 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:

$$T = \frac{df}{df + 50 + 150 \cdot cw/\overline{cw}} \tag{7}$$

$$I = \frac{\log(\frac{|DB|+0.5}{cf})}{\log(|DB| + 1.0)} \tag{8}$$

$$p(r_k|db_i) = 0.4 + 0.6 \cdot T \cdot I \tag{9}$$

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 Hetergeneous 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:

$$P(R \mid Q, C_{new}) = (X * P(R \mid Q, C_{comp})) + ((1 - X) * P(R \mid Q, C_{art})) \quad (10)$$

Where X is a pivot value between 0 and 1, and $P(R \mid Q, C_{new})$, $P(R \mid Q, C_{comp})$ and $P(R \mid Q, C_{art})$ are the new weight, the original component weight, and article weight for a given query. Although we found that a pivot value of 0.54 was most effective for INEX04 data and measures, we adopted the "neutral" pivot value of 0.5 for all of our 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 (made available to us by other INEX participants) 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 resulted in a large number of nonsensical tags and attributes being encoded into the DTD. For example, in the final DTD constructed the tag "wikipedialink" has over 2000 defined attributes (apparently 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).

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.

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.

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.

Table 1. Wikipedia Article-Level Indexes for INEX 2006

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 2. Wikipedia Components for INEX 2006

Name	Description	Contents
COMPONENT_SECTION	Sections	//section
COMPONENT_PARAS	Paragraphs	//p | //blockquote | //indentation1| //indentation2|//indentation3
COMPONENT_FIG	Figures	//figure

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.

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 "//USMARC/VarFlds/VarDFlds/Fld24*/a|b|n|p|f|l" (where "*" is a wildcard matching anything up to the end of the tag name and "|" indicating a union of

Table 3. Wikipedia Component Indexes for INEX 2006†Includes all subelements of section or paragraph elements

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

subordinate tag names), while in the IEEE collection they mapped to "//fm/tig/atl" and in the CompuScience collection simply to "//title".'

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 focused 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

Table 4. Components of the heterogeneous collection. *Element counts estimated for large collections.*

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

algorithms, respectively, with no blind feedback. The two Content-Only runs used the TREC2 and TREC3 algorithms, also with no blind feedback.

Our Resource Selection runs used the same techniques described in our earlier work on distributed IR[12,13]. The first step in this process is creating "document-like" collection representatives for each collection. This is done by "harvesting" the collection representatives using some features of the Z39.50 Information Retrieval protocol. The harvested collection representatives are then used for the subsequent retrieval and ranking of those collections. The collection representatives created are XML structures containing all of the terms from the indexes for each of the index type shown in the tables above, along with the collection frequency for each term. Because our indexes are based on the XML structure of the documents in the collection, the segments in the collection representatives also reflect that structure. During indexing of the collection representatives the frequency information included is used as if it were the within-document term frequency and incorporated into the collection representative indexes.

During retrieval the ranking algorithms describe above were used to rank the collection representatives as if they were normal documents, and that rank reported in the results.

The Content-Only runs used data fusion methods to merge the result sets of separate searches of each collection for a given element type. Because both of the Content-Only runs used probabilistic methods for each search, the fusion method used was to simply normalize the scores to the same basic range (using MINMAX normalization) and then simply merge all result sets based on this normalized

Table 5. Document Indexes for all Heterogeneous collections for INEX 2006

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

score. A better approach would probably be to first use the collection ranking method and give differing weights to items from particular collections. But for this initial work, we decided to use only the simplest approach, and elaborate later once we had result data to help tune the merging method.

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 stategies and to adapt previous work to the new collections.

However, in setting up to submit runs, and actually running full query sets for the Heterogeneous track, we were able to find that using "concept mapping" for each of the collections and using distributed search methods allowed us to resolve and obtain results for each collection (although we will not know the actual effectiveness of the approach until we have a complete evaluation). As noted above, this approach used a set of "element classes" as the the search elements in place of the actual structural elements for searching. Thus, a "title" search was mapped to the appropriate structural element in each collection. Obviously, there are limits to this approach. If, for example, appropriate structural elements for searches are not known at the point of database creation and indexing, then they are not made accessible. Another issue is whether collections should be weighted differently in calculation of the merged ranked results list. As suggested above, it would be possible to the collection-selection ranking to give priority to the collections estimated to be most likely to contain relevant items.

We hope to be able to try some of these other methods, and additional new techniques for INEX 2007.

References

1. Callan, J.: Distributed information retrieval. In: Croft, W.B. (ed.) Advances in Information Retrieval: Recent research from the Center for Intelligent Information Retrieval. chapter 5, pp. 127–150. Kluwer, Boston (2000)
2. Callan, J.P., Lu, Z., Croft, W.B.: Searching Distributed Collections with Inference Networks. In: Fox, E.A., Ingwersen, P., Fidel, R. (eds.) Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, pp. 21–28. ACM Press, New York (1995)
3. Chen, A.: Multilingual information retrieval using english and chinese queries. In: Peters, C., Braschler, M., Gonzalo, J., Kluck, M. (eds.) CLEF 2001. LNCS, vol. 2406, pp. 44–58. Springer, Heidelberg (2002)
4. Chen, A.: Cross-Language Retrieval Experiments at CLEF 2002. In: Peters, C., Braschler, M., Gonzalo, J. (eds.) CLEF 2002. LNCS, vol. 2785, pp. 28–48. Springer, Heidelberg (2003)
5. Chen, A., Gey, F.C: Multilingual information retrieval using machine translation, relevance feedback and decompounding. Information Retrieval 7, 149–182 (2004)
6. Cooper, W.S., Chen, A., Gey, F.C.: Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression. In: Text REtrieval Conference (TREC-2), pp. 57–66 (1994)
7. Cooper, W.S., Gey, F.C., Chen, A.: Full text retrieval based on a probabilistic equation with coefficients fitted by logistic regression. In: Harman, D. K. (ed.) The Second Text Retrieval Conference (TREC-2) (NIST Special Publication 500–215), Gaithersburg, MD, National Institute of Standards and Technology pp. 57–66, (1994)
8. Cooper, W.S., Gey, F.C., Dabney, D.P.: Probabilistic retrieval based on staged logistic regression. In: 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, pp. 21–24. ACM, New York (June 21-24, 1992)
9. Gravano, L., García-Molina, H.: Generalizing GlOSS to vector-space databases and broker hierarchies. In: International Conference on Very Large Databases, VLDB, pp. 78–89 (1995)
10. Gravano, L., García-Molina, H., Tomasic, A.: GlOSS: text-source discovery over the Internet. ACM Transactions on Database Systems 24(2), 229–264 (1999)
11. Harman, D.: Relevance feedback and other query modification techniques. In: Frakes, W., Baeza-Yates, R. (eds.) Information Retrieval: Data Structures & Algorithms, pp. 241–263. Prentice-Hall, Englewood Cliffs (1992)
12. Larson, R.R.: 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, pp. 399–400. ACM, New York (2002)
13. Larson, R.R.: Distributed IR for digital libraries. In: Koch, T., Sølvberg, I.T. (eds.) ECDL 2003. LNCS, vol. 2769, pp. 487–498. Springer, Heidelberg (2003)
14. Larson, R.R.: A fusion approach to XML structured document retrieval. Information Retrieval 8, 601–629 (2005)

15. Larson, R.R.: Probabilistic retrieval, component fusion and blind feedback for XML retrieval. In: INEX 2005. LNCS, vol. 3977, pp. 225–239. Springer, Heidelberg (2006)
16. Mass, Y., Mandelbrod, M.: Component ranking and automatic query refinement for xml retrieval. Advances in XML Information Retrieval. In: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX2004, pp. 73–84. Springer, Heidelberg (2005)
17. Robertson, S. E., Jones, K.S.: Relevance weighting of search terms. Journal of the American Society for Information Science, pp. 129–146, (May– June 1976)
18. Voorhees, E., Harman, D. (eds.): In: The Seventh Text Retrieval Conference (TREC-7). NIST (1998)
19. Voorhees, E., Harman, D. (eds.): In: The Eighth Text Retrieval Conference (TREC-8). NIST (1999)
20. Xu, J., Callan, J.: Effective retrieval with distributed collections. In: Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 112–120 (1998)

The INEX 2006 Multimedia Track

Thijs Westerveld[1] and Roelof van Zwol[2]

[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, it's 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 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`). Wikitext pages have been converted to XML [1] making a structured collection of

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 331–344, 2007.

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. 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. 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)]
```

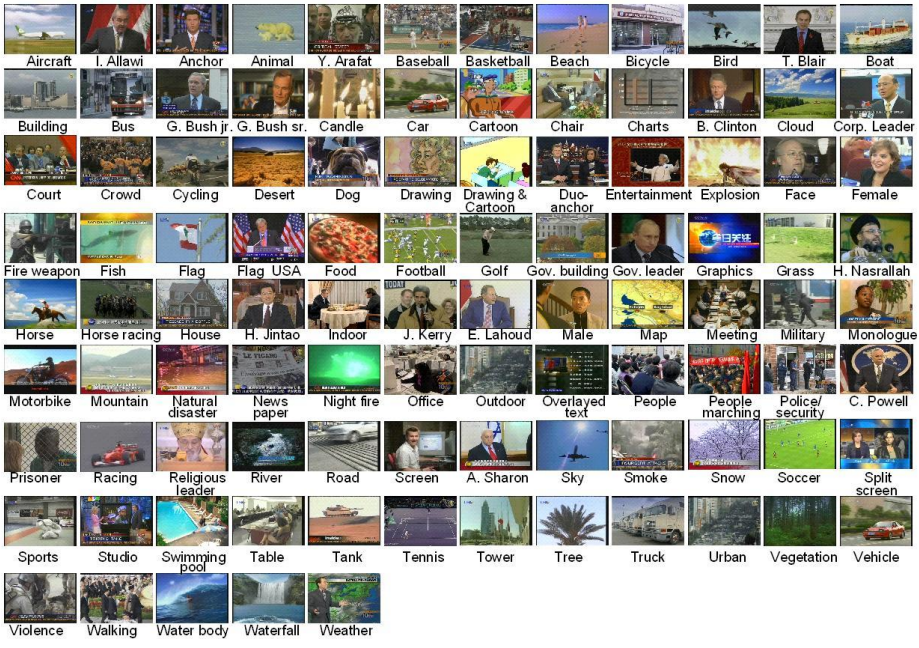


Fig. 2. The 101 concepts for which classification scores are available. This image is taken from [4]

Terms following the keyword *concept:* are obviously restricted to the 101 concepts for which classification results are provided (cf. Figure 2).

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/or 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 than 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>`.

The *MMimages* task is a document retrieval tasks. A document, i.e., an image with its metadata, is either relevant or not. For this task we adopted TREC style document pooling of the documents and binary assessments at the document level.

5.1 Pool Quality and Assessor Agreement

We managed to have a few topics in the *MMimages* task assessed by multiple assessors. In addition, these new assessors looked at pools of a greater depth than the original assessments: 1500 instead of 500 documents per topic. This allows us to study inter-assessor agreement as well as the quality of the pool. For six topics we had a second person assessing the relevance of the documents. The assessment procedure was the same as for the primary assessor and the second assessor had no access to the original assessments.

Agreement between the assessors was extremely high, measured only on the pools of depth 500 that were seen by both assessors, we get 95% agreement on average. For most topics this was much higher, for one topic (topic 2: *images of castles*) the agreement is considerably lower, apparently the second assessor was less strict in the definition of a castle. Table 3 shows the details.

Starting from the relevance assessments we have for the top 500 and top 1500 pools, we artificially constructed the judgements we would have had by pooling the top 10, 20 50, 100, 200, 500, 1000 and 1500 (Note that the top 1000 and top 1500 pools are only available for six out of 10 topics). Figure 3 shows the mean average precision scores for all submissions for the various pool depths. The relative ordering does not seem to change much for the pool depths above 100, below some instability is visible. This is even more apparent if we look at the correlation between the system orderings based on the various judgement sets (Figure 4). The correlation between the adapted pool based rankings and the official system ranking are above 0.90 for pool depths of 50 and up, and above 0.95 for pool depths over 100.

Table 3. Assessor agreement: general agreement, agreement on relevant and agreement on non-relevant documents

topic	agree	agreeREL	agreeNONREL
1	0.94	0.77	0.97
2	0.77	0.56	0.84
8	0.99	0.73	1.00
9	1.00	0.89	1.00
10	0.98	0.88	0.99
13	1.00	0.88	1.00
average	0.95	0.78	0.97

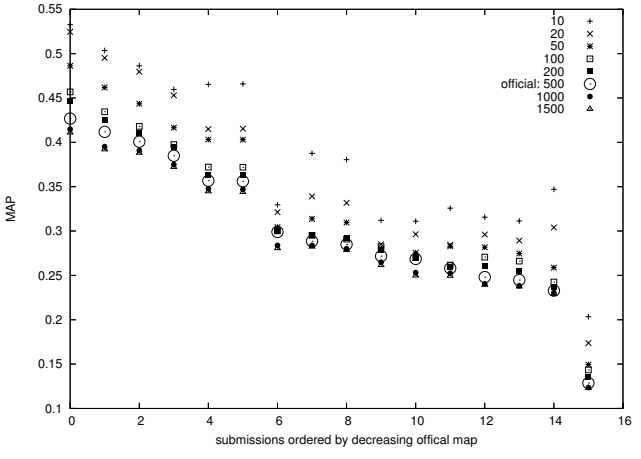


Fig. 3. Mean average precision after evaluating using assessments based on various pool depths

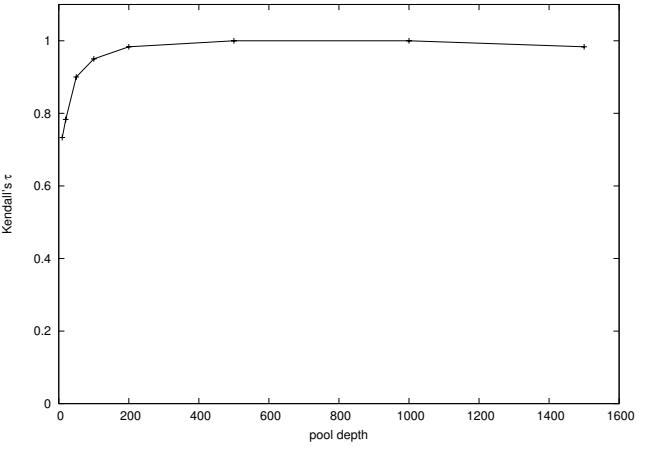


Fig. 4. Kendall's τ between assessments based on various pooldepth and the original assessments based on the top 500 pooled documents

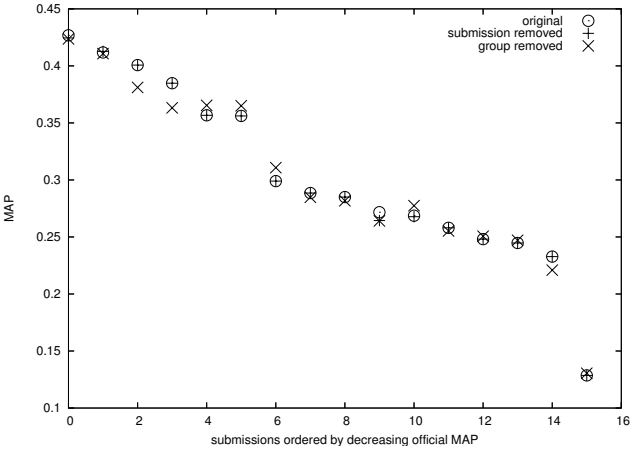


Fig. 5. The effect of pool contribution on mean average precision

Finally, we tested the usefulness of the *MMimages* test collection for future use. There exists a potential bias against approaches that were not able to contribute to the pool. Especially with the low number of submissions, this risk is high. To test the effect of this we re-evaluated each submission after removing the documents uniquely contributed to the pool by this submission. Since submissions from the same group are often based on highly similar approaches, we repeated the experiment with removing the documents uniquely contributed by the group. Figure 5 shows for each submission the original mean average precision scores as well as the scores based on these modified sets of assessments. Again, the ordering of the runs is hardly affected by whether groups did or did not contribute to the pool. The correlation between the original system ranking and the rankings based on the assessments with submission or group removed is 0.98 and 0.95 respectively.

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 4 and 5 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.

Table 4. Topic fields used by the submitted runs

field	*#MMfragments* runs using it	*#MMimages* runs using it
title	10	14
castitle	7	7
description	2	3
narrative	0	0

Table 5. Resources used by the submitted runs

resource	*#MMfragments* runs using it	*#MMimages* runs using it
wikipedia	15	0
wikipedia_IMG	3	16
UvAfeatures	0	3
UvAconcepts	0	1
RMIT_GIFT	2	10

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 6 shows mean average precision scores for all submitted *MMfragments* runs, Figure 6 shows the ep/gr graphs. Judging from the fields 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.

MMimages. In the *MMimages* task some topics contain image examples from the test collection. To not give an unfair advantage to systems that use the image examples, retrieving the example images should not be rewarded. Therefore, we re-interpreted these queries as "Find *other* images about X like this one". We removed the example images from the assessment files and from the submissions and based the evaluations on that.

Figures 7 shows the interpolated recall precision graphs for the submitted runs for the *MMimages* task, Table 7 shows the mean average precision 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 appproaches taken by the groups participating in the multimedia track at INEX 2006.

Table 6. 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

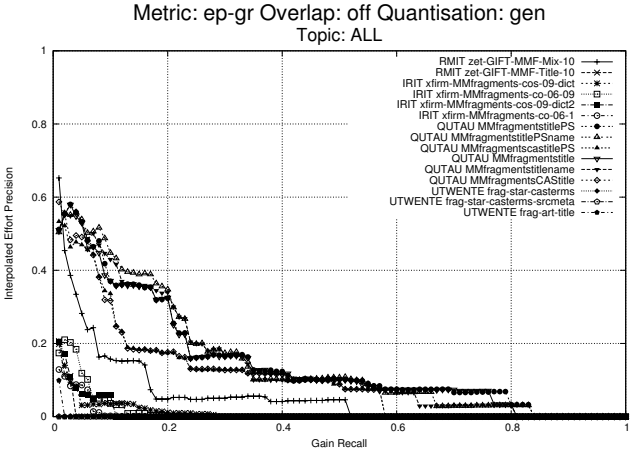


Fig. 6. MMfragments: Effort-precision/gain-recall (ep/gr), overlap not penalised

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.

Table 7. Mean average precision (MAP) for submitted *MMimages* runs

group	run	MAP
UTWENTE	frag_article_title	0.3835
UTWENTE	img_cas_noMM	0.3681
QUTAU	Uva_Castitle_Fusion	0.3430
QUTAU	HOT_CasTitle_06	0.3327
RMIT	zet-GIFT-MMI-Title-05	0.3104
RMIT	zet-GIFT-MMI-Title-10	0.3098
QUTAU	HOT_Title_Fusion	0.2686
RMIT	zet-GIFT-MMI-Mix-05	0.2645
RMIT	zet-GIFT-MMI-Mix-10	0.2611
QUTAU	Uva_Title	0.2364
IRIT	CASMETHOD	0.2270
RMIT	zet-GIFT-MMI-Title-00	0.2184
IRIT	ImagesMethodV2	0.2159
IRIT	ImagesMethod1.1	0.2122
RMIT	zet-GIFT-MMI-Mix-00	0.2074
IRIT	COMethod	0.1140

IRIT. IRIT participated in both the *MMfragments* and *MMimages* tasks of the INEX 2006 MM track. For *MMfragments* topics, the XFIRM "Content Only" and "Contend And Structure" methods intended for adhoc retrieval are used 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 engine[1] for XML retrieval and combined these text based results with content based image

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

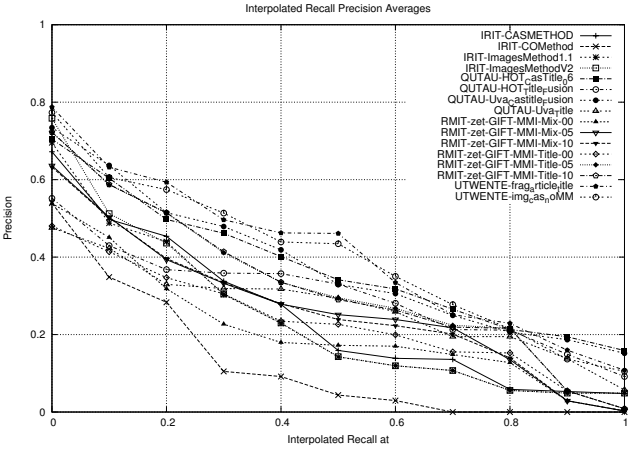


Fig. 7. MMimages: Interpolated Recall Precision Averages

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 slighly 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.

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 creates 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 indicatiors of relevance. Also, a topic by topic analysis could shine some light. Perhaps these techniques did contribute for only a limited number of topics and hurt for others.

Despite the low number of participants the quality of the *MMimages* test collection is good. Assessor agreement on this collection is high and system comparison are hardly affected by pool depth or by whether they did or did not contribute to the pool. This makes the *MMimages* collection with the topics and judgements a useful and reusable test collection for image search.

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. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. SIGIR Forum (2006)
2. Ingewersen, P., Järvelin, K.: The Turn, Integration of Information Seeking and Retrieval in Context, chapter 5, pp. 206–214. Springer, Heidelberg (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. Snoek, C., Worring, M., van Gemert, J.C., Geusebroek, J.-M., Smeulders, A.W.M.: The challenge problem for automated detection of 101 semantic concepts in multimedia. In: MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia, pp. 421–430. ACM Press, New York (2006)
5. Theobald, M., Schenkel, R., Weikum, G.: An efficient and versatile query engine for topx search. In: VLDB '05: In: Proceedings of the 31st international conference on Very large data bases, pp. 625–636. VLDB Endowment (2005)
6. Trotman, A., Sigurbjörnsson, B.: Narrowed extended xpath I (NEXI. In: Fuhr, N., Lalmas, M., Malik, S., Szlavik, Z. (eds.) INEX 2004, vol. 3493, Springer, Heidelberg (2004) http://www.springeronline.com/ 3-540- 26166-4.
7. van Gemert, J.C., Geusebroek, J.-M., Veenman, C.J., Snoek, C.G., Smeulders, A.W.: Robust scene categorization by learning image statistics in context. In: CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop, Washington, DC, USA, p. 105. IEEE Computer Society, Los Alamitos (2006)

Fusing Visual and Textual Retrieval Techniques to Effectively Search Large Collections of Wikipedia Images

C. Lau, D. Tjondronegoro, J. Zhang, S. Geva, and Y. Liu

Faculty of Information Technology, Queensland University of Technology,
2 George Street, GPO Box 2434, Brisbane, QLD 4001 Australia
{andy.lau,dian,jinglan.zhang,s.geva}@qut.edu.au,
y53.liu@student.qut.edu.au

Abstract. This paper presents an experimental study that examines the performance of various combination techniques for content-based image retrieval using a fusion of visual and textual search results. The evaluation is comprehensively benchmarked using more than 160,000 samples from INEX-MM2006 images dataset and the corresponding XML documents. For visual search, we have successfully combined Hough transform, Object's color histogram, and Texture (H.O.T). For comparison purposes, we used the provided UvA features. Based on the evaluation, our submissions show that Uva+Text combination performs most effectively, but it is closely followed by our H.O.T- (visual only) feature. Moreover, H.O.T+Text performance is still better than UvA (visual) only. These findings show that the combination of effective text and visual search results can improve the overall performance of CBIR in Wikipedia collections which contain a heterogeneous (i.e. wide) range of genres and topics.

1 Introduction

Recently there has been a renewed spurt of the research activity in multimedia information retrieval [1-3]. This can be partly due to the rapid proliferation of the Internet. The current World Wide Web increasingly uses structured XML documents that contain not only the text information, but also other media information such as images, videos, and speech. Images have been playing an important role in human life as one of the most important information resources. The amount of images on the Web is increasing exponentially due to the advent of various digital devices such as digital cameras and scanners. Nowadays the problem is not *finding* information, but how to find *useful* information efficiently and effectively. Image retrieval techniques have attracted intensive interests from both the industry and the research communities.

Currently there are two paradigms in image retrieval: text-based and content-based. Text-based image retrieval techniques perform retrieval using keywords. Most of the state-of-the-art systems, such as Google and Yahoo!, belong to this paradigm. Although text description is closer to the concepts in human mind, they rely heavily on manual annotation of images for keyword matching, which is tedious and costly. In addition, it is hard to define a unified set of keywords to be used for annotation. Although the concept of using game players to help label images on the web via an online game is very interesting [4], the final effect of this approach to effectively label

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 345–357, 2007.

images still needs to be verified due to Internet abuse [24]. Content-based image retrieval turns to visual features for searching similar images, which alleviates the burden of manually labelling images. However, due to the well-known semantic gap between low-level visual features and the corresponding high-level concepts, visual features are generally not sufficient for searching similar images [5].

In order to mitigate this semantic gap, multi-modal image retrieval, which uses both text and content-based searching, is attracting uprising interests [6]. It has been proved that better retrieval results will be achieved by appropriately fusing different modal information [7]. In most cases, especially on the Web, images do not exist separately. Instead, there is much relevant information surrounding these images. Combing different modality, e.g. images and textual information, would improve the retrieval accuracy; fusion of the image and the text will also make the query more flexible for users. A user could search the image database by the image and/or the text. Although combining the image with the text together has been studied just recently, there are still some open issues needing further study, such as how to combine the content and text-based image retrieval results together.

The main goal of this research is to develop and evaluate algorithms for structured document retrieval systems using comprehensive database of XML documents containing text and image 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 color depths. This project aims at creating 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 engine 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 of how to fuse multiple results to form 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.

2 Related Work

This section summarizes some previous work in image feature extraction and multi-modal image retrieval.

Image Feature Extraction

Compared with text-based image retrieval which takes advantage of keywords or metadata such as captions, authors, and textual descriptions, content-based image retrieval is more complicated and has to extract the appropriate visual features first [5].

Color is one of the most widely used visual features. The choice of a color space is of great importance to the proper color-based image retrieval. HSV and YCbCr are two commonly used color spaces which can better model the human color perception [8].

The histogram, which describes the distribution of colors in an image, is a traditional representation of the "color" feature. It is usually high dimensional and contains more global information of the image.

Texture is another important feature which represents some important aspects of many real-world images, such as bricks, coins, trees, etc. Texture has characteristics such as periodicity and scale, and could be represented in terms of direction, coarseness, contrast, etc. In this sense, texture features contain the high-level semantics for image retrieval [5, 8]. Texture features could be divided into two categories: the structural texture and the statistical texture. The structural method represents the texture by identifying structural primitives and their location rules, which consists of morphological operator and adjacency graph. The statistical approach, which is one of the earliest methods to classify textures, describes texture by the spatial distribution of image density.

Hough transform is a kind of feature extraction method which can identify objects with a particular shape in an image. It includes two kinds of transform methods: the classical transform and the generalized transform. The classical Hough transform is usually used to detect the regular curves such as lines, circles, ellipses, etc. The generalized transform is applicable for the detection of positions of arbitrary shapes which cannot be described by using the simple features.

We will use color histogram, statistical texture, and generalized Hough transform in our experiment.

Fusing Image and Text Retrieval

One challenge for image retrieval is to find a simple but effective way to form a query. Most content-based image retrieval systems support query-by-example in which users should provide visual example of the contents they seek. In this case images are searched on the basis of matching of content features such as color, shape, and texture. Therefore, this method is more intuitive, and an appropriate key-image is dispensable to start a query. However, this query method has two drawbacks. Firstly, a user may not find such an appropriate image which can represent the use's query need completely in some cases. Secondly, the representation of the image is not as flexible as the textual description, and most users have been used to adopting keywords to start their query and describe their needs.

Text-based image retrieval can address these problems, which is based on the assumption that the textual description can express the semantics of images [9]. It allows users to search images by specifying their own query in terms of a limited vocabulary of semantic concepts. But *synonymy* and *polysemy* are two large existing problems in information retrieval [10]. Many words have more than one distinct meaning. Users in different contexts may express the same needs by using different terms. In addition, there are many ways to describe the same object. In different contexts or when used by different people, the same terms can be taken differently. The prevalence of *synonymy* and *polysemy* tends to degrade *precision* and *recall* performance of the system.

It is believed by many researchers that combining the keyword-based approach and the content-based approach together can benefit from the strengths of both paradigms, and these two paradigms can supplement each other. R. Zhang et al. [6] in their paper show that multi-modal image retrieval is a promising way to improve image retrieval and enhance users' querying modalities. There are several ways for multi-modal

fusion, such as linear combination, min-max aggregation, and voting production combination [7]. These methods can be classified into two categories: fusion at feature level and fusion at output level. It has proven that the fusion on output level outperforms the fusion on feature level in most cases [7].

E. Chang et al. [11] suggest that the user can start a query by a few keywords, and after a few relevant images are returned, the image features with their annotation can be used to perform a multi-modal query refinement. J.L. Martinez-Fernandez et al. presents similar ideas [12]. They refine the text-based search results with the addition of content-based image retrieval. Based on their successful previous work on organizing images according to the visual similarity for image retrieval, D. Cai et al. [13] use low-level features to cluster images into semantic clusters obtained by the textual features. R. Basancon et al. [14] presents the opposite idea. They first search the candidate images using content-based methods, and then use the textual description of these candidate images as query keywords to search again.

However, regardless of whether the text-based results are refined using content-based methods or vice versa, it is still insufficient to improve the performance of image retrieval. The reason is that owing to the intrinsic drawbacks of text- and content-based approaches, it is hard to confirm which method could achieve better results in terms of a specific query. It is our hypothesis that the late combination would assure better searching results. D. Tjondronegoro et al. [15]indicate, by a series of experiments, that the results of content-based retrieval with the help of text-based retrieval is much better than any individual text-based or image-based retrieval. During the work in the INEX 2005, they designed two search engines: an XML document search engine using both structural and content information and a content-based image search engine. When a query is submitted, these two engines work respectively at the same time, and then the retrieval results are merged together by treating the visual features as the text terms. The question of how to fuse content-based retrieval with text-based retrieval, however, still needs further consideration. There are no common agreements on the fusion approaches.

3 System Architecture

The framework of the prototype system is shown Figure 1. Users interact with a Graphical User Interface (GUI) to write queries and browse results. The GUI merges results from an integrated document searching that fuses image-text results. The database stores images and the XML documents which display (i.e. provide links) them. Three (3) image related tables are used in the database: Images, Features, and AdhocXML (as shown in Figure 2). 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 also allows it to fuse the image and text search results. The selected image features for this experiment include color 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, Color 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].

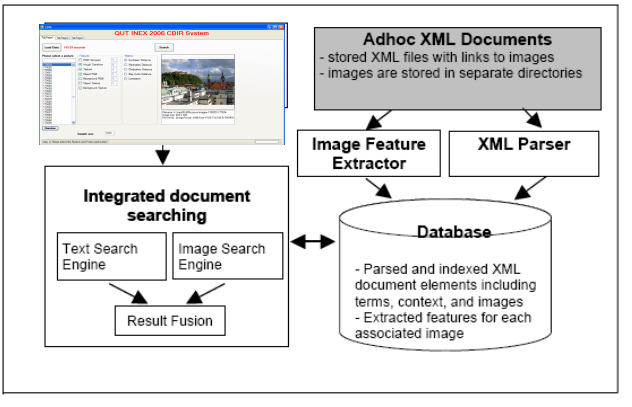


Fig. 1. System Architecture

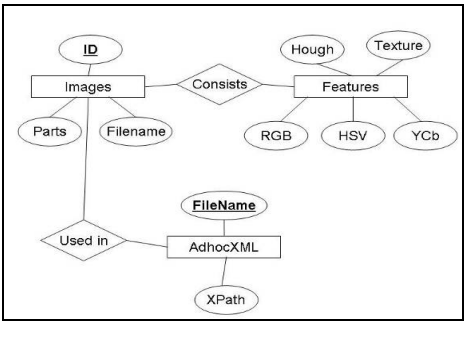


Fig. 2. Database Entity-Relationship

Graphical User Interface (GUI)

Using the GUI, users can choose which features to be combined and adjust the weights of each individual feature, as well as the metrics of distance calculation. The GUI also provides a preview picture box and the information of the image.

Memory Management

Due to the large amount of data and high dimensionality of the image feature, a data access technique is required to avoid system resources overload (while calculations

are performed in the memory). The system is designed to perform calculation in sequential order. Each time a block of 100 image features is read and stored in the memory buffer, only the normalized distance (from the query image) is kept in the buffer, while the rest are overwritten. This method is found to be efficient and reliable (i.e. no buffer overflow) during our experiments.

4 Using Visual Retrieval Techniques

The image features used in this project are color histogram, texture and detectable lines (Hough transform).

Color Histogram
Color histogram has been widely used in CBIR to represent the global image color attribute. For our experiment, we compared the use of RGB, HSV, and YCbCr color spaces. RGB color histogram is invariant with translation and viewing-axis rotation, and varies only slowly with the angle of view. Therefore, it is particularly suited for recognizing an object of unknown position and rotation within a scene. Moreover, translation of an RGB image into the illumination invariant rg-chromaticity space allows the histogram to operate well in varying light levels [16]. HSV Histogram is one of the most commonly used by people (especially artist) to select color. It is a cylindrical coordinate system where the angle defines hue, the radius defines saturation, and the vertical axis defines color of an image. As compared to RGB, which is an additive model, HSV encapsulates information about each color in a more understandable and recognizable manner to humans who perceive color via its color name, vibrancy and darkness/brightness. Similarity distance between HSV color space often performs better than RGB (e.g. [17]). The YCbCr color histogram represents the luminance level using a single component, Y, and other color information is stored individually using Cb and Cr. This feature is frequently used in skin color detection projects (e.g. [18]) as values extracted from Cb and Cr can effectively represent skin color of most human races. Figure 3 illustrates that these various color spaces will return different results for the same query, thus we aim to compare the effectiveness of each color representation for different topics.

Fig. 3. Sample Results from Retrieval using Various Color Histogram (From top to bottom: RGB, HSBV, YCbCr Results)

Object Histogram

A pre-processing of object extraction is performed on every image and hence only the histogram of the extracted segments is calculated. This technique is useful when performing queries with standout and dominant object(s). The algorithm described in [19] can efficiently extract building and human apart from the background. The object extraction accuracy generally increases when the color of the background is much different from the object itself. Several researches, including our previous work [15, 20], shows that object extraction can be used to enhance the image retrieval system performance. Figure 4 shows an example of object-extraction result on an airplane image which has a distinctive object features.

Hough Transform

Hough transform is widely used in digital image processing to detect arbitrary shapes and identify lines (e.g. [21]). For the Wikipedia images dataset, we expect that this method can effectively distinguish some topics such as buildings, coins and some arbitrary shapes since they have unique lines strength characteristics. Figure 5 shows the use of Hough transform to detect coin shapes.

Fig. 4. Object extraction algorithm

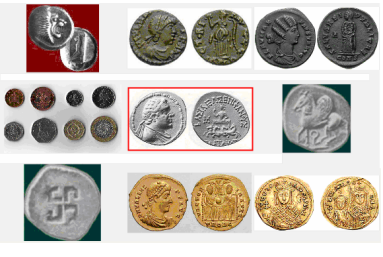


Fig. 5. Hough Transform detect Coin Shapes

Texture

Texture can often represent the visual properties of an image which are unable to represent by color intensity alone. For example, while sea and sky are both blue, but their texture features are different In this project we used a 6D texture-vector which comprises of the intensity, contrast, smoothness, skewness, uniformity and randomness, using a method which has been described in our previous work [19]. We expect that texture feature is likely to perform well in locating images with distinctive and strong unique patterns.

UvA Features
This feature uses Natural Image Statistics, Color invariant edge detection and regional descriptors to represent an image. The paper [22] presents a 120D features using different texture parameters. In our project, we will compare UvA features with our features set.

Similarity Measurement Metrics
We implemented and compared the performance of the following similarity measurement metrics: *Euclidean* [23], *Manhattan* [23], *Chebyshev*, *Bray Curtis* and *Correlation*. Euclidean distance is commonly used in most of the image retrieval system and Correlation is able to indicate the strength and direction of 2 vectors to show how closely one image correlate to another. To represent the greatest of their differences along any coordinate dimension, the Chebyshev distance of any two points is defined as:

$$D = \max(|x_2 - x_1|, |y_2 - y_1|)$$

Whereas, Bray Curtis distance is defined as:

$$D = \frac{\sum_{k=1}^{n} |x_{ik} - x_{jk}|}{\sum_{k=1}^{n} (x_{ik} + x_{jk})}$$

5 Using Text Retrieval Techniques

The system that we have used for XML text based retrieval is the GPX search engine [23]. The system is described in detail in this proceedings collection under the title "GPX - Gardens Point XML IR at INEX 2006". The MM Fragments task was applied without any changes to the system. For the MM Images task we had to index the new collection but no other work or modifications were required. The system ignored search by image source. All clauses such as about (.,src:988336) were simply ignored by the system. It was left to the image retrieval system to take account of example images and fuse the CBIR results with the text retrieval results, as outlined in the next section.

6 Combining Visual and Textual Search Results

The fusion of search results is performed in twofold: 1) combination of image features, and 2) fusion of text-based search and image-based search results. To combine multiple image features, a weighted sum is performed on the individual distances calculated using different features. This is based on the hypothesis that the combination of more evidence will increase the reliability of the similarity calculation.

To combine the results of text-based and image-based search engines, we use the result of the text search as the base, while the image search result is used to boost the confidence. Our text-image fusion algorithm relies on a hypothesis that if a document

appears in both text and image based search results, then the final ranking of the document should be pushed to the top. This hypothesis has not been thoroughly proven as our experiment shows that sometimes the individual search results (either image-based or text-based) can still be better than the combined. More details will be discussed in Section 7 (Evaluation).

7 Evaluation

Our system's prototype is designed using VB.Net language in Microsoft .NET 2.0. The database used is Microsoft Access. The image feature extraction is performed using MATLAB with Image Processing Toolbox. The image collection used is provided by INEX 2006 (http://inex.is.informatik.uni-duisburg.de/2006/). The corpus contains 166,559 images of different format such as PNG, JPG and GIF. All of the images come in different size, different color depth and present various type of content from landscape, buildings, people, and buildings. Table 1 summarizes the topics of INEX 2006 MM tasks. For each task, we will re-interpret the queries as "Find *other* images about X like this one".

Table 1. INEX 2006 Image Retrieval Topics

Topic ID	Topic Title
1	Sunset sky
2	Historic castle
3	Barcelona
4	Bactrian Coins
5	Masjid (Mosque) Malaysia
6	Da Vinci's Color Paintings
7	Mountain for hiking
8	Images of bees with flowers
9	Golden gate bridge
10	Stringed musical instruments
11	Frank Lloyd Wright Buildings
12	Rodin "Burghers of Calais"
13	Official logos of parties in Sweden

We experimented with various combinations of features and found that the combination of Hough transform (H), the Object color histogram (O) and the Texture (T) with equal weighting performs best (among other combinations).

To illustrate the benefits/weaknesses of using text- or image- results only, or using the fusion of text-image results, Figure 6 and 7 shows the text-only search results and fused search results of Topic 9, respectively. For this case, our system is able to refine the text search by bringing forward the 2 visually similar images with lower ranking.

To illustrate the benefits of our H.O.T, Figure 8 and 9 shows the results for Topic 4 using visual features only and fusion of text-image, respectively. In this case, visual

features effectively locate images with similar color, shapes and texture. However, with the fusion of the text and image, the results are actually affected by the *Bactrian camel* which is irrelevant.

After evaluating various similarity metrics, we found that the results of applying different similarity measurement metrics are similar and we decided to use *Euclidean* distance as it is one of the most commonly used similarity measurements,

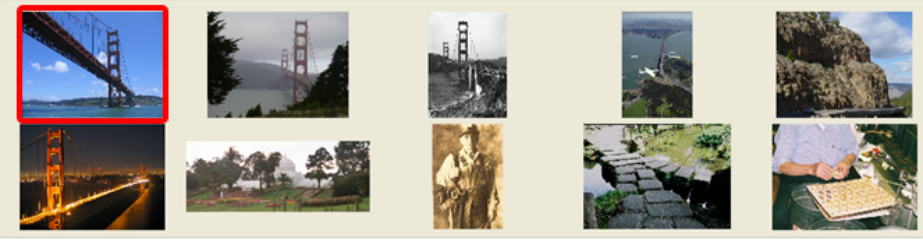


Fig. 6. Text Results of Topic 9

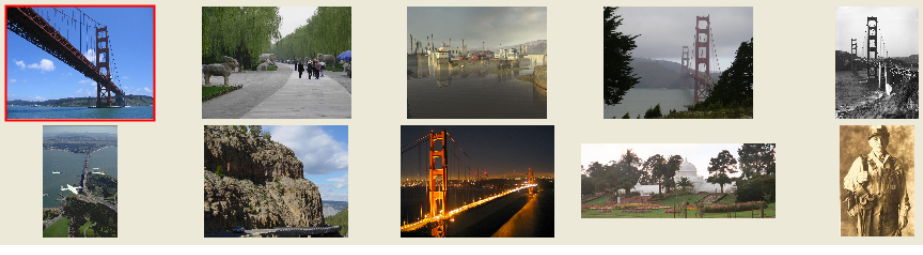


Fig. 7. Fused (i.e Text-Image) Search Results of Topic 9 (This shows that fusion of text and image search results is better than text search alone)

Fig. 8. H.O.T image-based search result for Topic 4

Figure 10 depicts the official evaluation result from INEX 2006 for QUT submissions (QUT's performance is highlighted- while the non-highlighted are from other participants). Based on the interpolated precision-recall performance, the best technique is UvA+Text results. However, the H.O.T (visual only) feature performed at a similar quality, and in fact is better than UvA (visual) alone. This shows that for

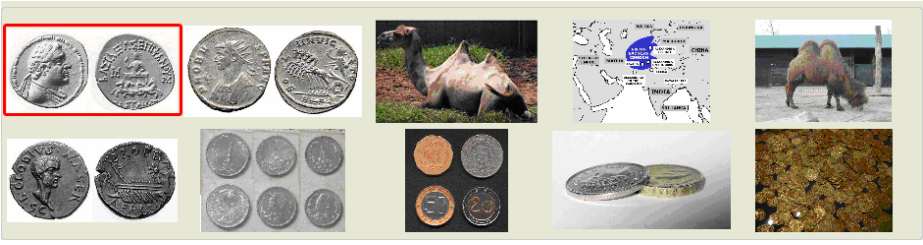


Fig. 9. Fused search results of Topic 4 (This shows that H.O.T feature performs better when it is used without merging with text results)

certain image retrieval task, a combination of simpler low-level features can be sufficient to produce a decent result. However, visual features cannot fully represent the semantics for an image, which is why text search needs to be exploited to improve the accuracy of search results. Moreover, XML documents which use images usually provide useful contextual information which sometimes can be close to users' search intention.

Furthermore, it is worth noting that detectable lines and shape description are effective to search images with distinctive objects such as cars, airplanes, coins, and buildings.

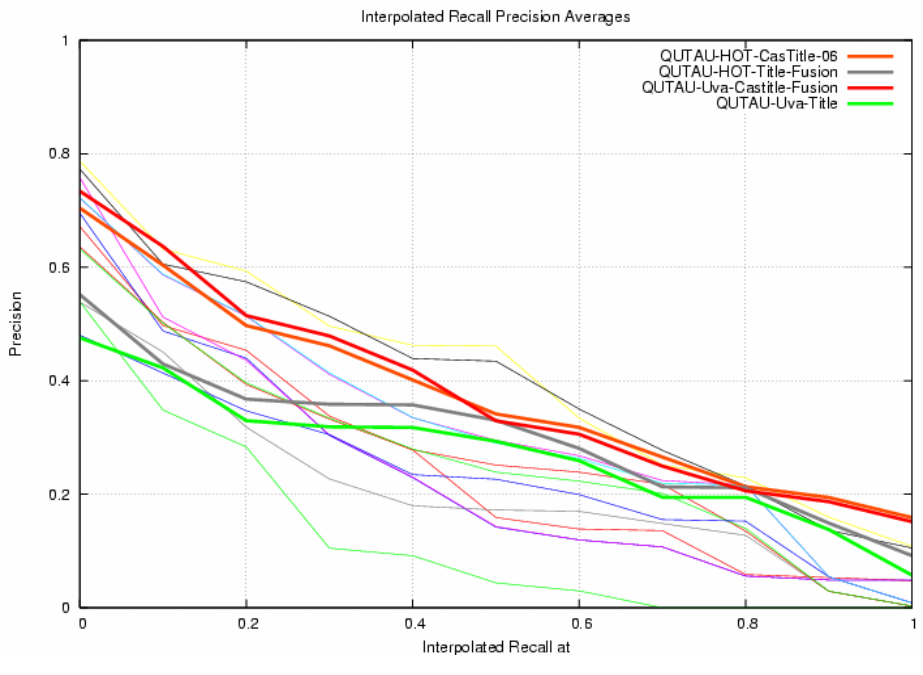


Fig. 10. Graph of Interpolated recall precision averages

8 Conclusions and Future Work

The experiment result shows that once the text-based image searching results are refined using content-based image searching results, the final combined result is generally better than the individual text or image-based searching results.

The semantic relation between keywords has not been investigated, and remains a subject of future investigation as it is one of the main reasons incurring low precision in image retrieval. Ontology contains concepts and their relations, therefore introducing ontology into multi-modal image retrieval will not only help better analyze the text, but are also useful for keyword matching. In this sense, text-based retrieval will become concept-based retrieval. We hypothesize that the fusion of concept and content will achieve much better results than the fusion of text and content.

In addition, fast access of the image database needs to be studied as it is very large and image features are usually high dimensional (from 100 to 1000) and slow down content-based matching.

References

[1] Lew, M.S., Sebe, N., Djeraba, C., Jain, R.: Content-based multimedia information retrieval: state of the art and challenges, ACM Transactions on Multimedia Computing. Communications and Applications 2, 1–19 (2006)

[2] Kherfi, M.L., Ziou, D., Bernardi, A.: Image retrieval from the World Wide Web: Issues, techniques and systems. ACM Computing Surveys 36, 35–67 (2004)

[3] Smeulders, A., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 1349–1380 (2000)

[4] von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: Proceedings of the 2004 conference on Human factors in computing systems, pp. 319–326 (2004)

[5] Liu, Y., Zhang, D., Lu, G., Ma, W.-Y.: A survey of content-based image retrieval with high-level semantics. Pattern Recognition 40, 262–282 (2007)

[6] Zhang, R., Zhang, Z.M., Li, M., Ma, W.-Y., Zhang, H.-J.: A probabilistic semantic model for image annotation and multi-modal image retrieval. Multimedia Systems 12, 27–33 (2006)

[7] Tong, H., He, J., Li, M., Zhang, C.: Graph based multi-modality learning. In: presented at Proceedings of the 13th annual ACM international conference on Multimedia, Hilton, Singapore (2005)

[8] Gevers, T., Smeulders, A.W.M.: Content-based image retrieval: an overview, in Emerging Topics in Computer Vision, Gerard Mediomi, S. B. k. (ed.) USA: IMSC, pp. 333–384 (2004)

[9] Gong, Z., Liu, Q., Zhang, J.: Web image retrieval refinement by visual contents. In: presented at The 7th International Conference on Web-Age Information Management (WAIM), Hong Kong, China (2006)

[10] Zhao, R., Grosky, W.: Narrowing the Semantic Gap - Improved Text-Based Web Document Retrieval Using Visual Features. IEEE TRANSACTIONS ON MULTIMEDIA 4, 189–200 (2002)

[11] Chang, E., Goh, K., Sychay, G., Wu, G.: CBSA: Content-based soft annotation for multimodal image retrieval using Bayes Point Machines. IEEE Transactions on Circuits and Systems for Video Technology, 13, 26–38 (2003)
[12] Martinez-Fernandez, J.V.R.J.L., Garcia-Serrano, A.M., Gonzalez-Cristobal, J.C.: Combining textual and visual features for image retrieval. In: Accessing Multilingual Information Repositories, vol. 4022, pp. 680–691. Springer, Heidelberg (2006)
[13] Deng Cai, X.H., Li, Z., Ma, W. -Y., Wen, J. -R.: Hierarchical clustering of WWW Image search results using visual, textual and link Information. In: presented at the 12th annual ACM international conference on Multimedia, New York, NY, USA (2004)
[14] Besancon, P.H.R., Moellic, P.-A., Fluhr, C.: Cross-media feedback strategies: merging text and image information to improve image retrieval, in Multilingual Information Access for Text, Speech and Images, vol. 3491, pp. 709–717. Springer, Heidelberg (2005)
[15] Tjondronegoro, J. D. Z., Gu, J., Nguyen, A., Geva, S.: Integrating Text Retrieval and Image Retrieval in XML Document Searching, presented at INEX 2005 (2006)
[16] Tapus, A., Vasudevan, S., Siegwart, R.: Towards a Multilevel Cognitive Probabilistic Representation of Space, In: Proceedings of SPIE (2005)
[17] Ma, W.-Y., Zhang, H.J.: Benchmarking of image features for content-based retrieval, Signals, Systems & Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference 1, 253–257 (1998)
[18] Chai, D., bouzerdoum, A.: A Bayesian approach to skin color classification in YCbCr colorspace, presented at TENCON, Kuala Lumpur, Malaysia (2000)
[19] Gonzalez, R.C., Woods, R.E., Eddins, S.L.: Digital Image processing using MATLAB. Upper Saddle River, N. J. :: Pearson Prentice Hall, (c2004)
[20] Kam, A.H., Ng, T.T., Kingsbury, N.G., Fitzgerald, W.J.: Content based image retrieval through object extraction andquerying, Content-based Access of Image and Video Libraries, 2000. In: Proceedings. IEEE Workshop, pp. 91–95 (2000)
[21] Duda, R.O., Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. Commun. ACM 15, 11–15 (1972)
[22] Gemert, J. C. v., Geusebroek, J.-M., Veenman, C. J., Snoek, C. G. M., Smeulders, A. W. M.: Robust scene categorization by learning image statistics in context. In: presented at CVPR Workshop on Semantic Learning Applications in Multimedia, New York, USA, (2006)
[23] Zhang, D., Lu, G.: Evaluation of similarity measurement for image retrieval, presented at Neural Networks and Signal Processing (2003)
[24] Google Image Labeler, Accessed: (March 17, 2007) http://en.wikipedia.org/wiki/Google_ Image_Labeler

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. Use of XML offers a structured approach for representing information while maintaining separation of form and content. XML information retrieval is different from standard text retrieval in two aspects: the XML structure may be of interest as part of the query; and the information does not have to be text. In this paper, we describe an investigation of approaches to retrieve text and images from a large collection of XML documents, performed in the course of our participation in the INEX 2006 Ad Hoc and Multimedia tracks. We evaluate three information retrieval similarity measures: Pivoted Cosine, Okapi BM25 and Dirichlet. We show that on the INEX 2006 Ad Hoc queries Okapi BM25 is the most effective among the three similarity measures used for retrieving text only, while Dirichlet is more suitable when retrieving heterogeneous (text and image) data.

Keywords: 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. Retrieving the desired information from an eXtensible Markup Language (XML) document involves retrieval of XML elements. This is not a trivial task as it may involve retrieving text and other multimedia elements.

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 two tracks: the Ad Hoc track, where we investigate the effects of using different information retrieval (IR) similarity measures; and the Multimedia (MM) track, where we combine retrieval techniques based on text and image similarity.

There are four XML retrieval tasks within the INEX 2006 Ad Hoc track: *Thorough, Focused, All In Context (AIC)* and *Best In Context (BIC)*. Using three IR similarity measures — Pivoted Cosine, Okapi BM25, and Dirichlet — in this paper we focus on the results obtained under *Thorough* and *AIC* tasks. Since the system we used is a full-text IR system which only does retrieval at document level, we only expected it to perform well on article retrieval in the *AIC* task.

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

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 358–372, 2007.

retrieval performance of an XML-driven multimedia information retrieval system.[1] Existing research on multimedia information retrieval from XML document collections is shown to be challenging [3,12,13]. For the *Multimedia Images (MMImages)* and *Multimedia Fragments (MMFragments)* tasks of the INEX 2006 MM track, we explore and analyse methods for combining evidence from content-based image retrieval (CBIR) with full-text IR. 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 encyclopedia. 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 tracks followed by the performance results obtained on the *Thorough* and *AIC* tasks of the Ad Hoc track. In Section 3 we present the INEX 2006 multimedia topics and their corresponding relevance judgements. In Section 4 we describe our approach to retrieve XML articles and the associated images based on the multimedia topics used in the MM track. In Section 5 we present results obtained from our experiments on the two tasks of the INEX 2006 MM track. We conclude in Section 6 with a discussion of our findings and outline directions for future work.

2 Full-Text Information Retrieval

In this section, we describe the three similarity measures implemented in Zettair, and show performance results on the *Thorough* and *AIC* tasks 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 tracks, 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 [14]. A similarity measure is used to rank documents by likely relevance to the query; in this work, we report on experiments using three different similarity measures

[1] INEX 2006 Multimedia Track Guidelines.
[2] http://en.wikipedia.org/wiki/Wiki
[3] http://www.seg.rmit.edu.au/zettair

implemented in Zettair, which respectively follow the three major models to information retrieval: the vector-space model, the probabilistic model, and the language model.

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 [16], we define the basic term statistics as:

- q, a query;
- t, a query term;
- d, a document;
- $N_{\mathscr{D}}$, the number of all the documents in the collection;
- For each term t:
  - $f_{d,t}$, the frequency of t in the document d;
  - $N_{\mathscr{D}_t}$, the number of documents containing the term t; 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:

- $\mathscr{D}$, the set of all the documents in the collection;
- $\mathscr{D}_t$, the set of documents containing term t;
- $\mathscr{T}_d$, the set of distinct terms in the document d;
- $\mathscr{T}_q$, the set of distinct terms in the query, and $\mathscr{T}_{q,d} = \mathscr{T}_q \cap \mathscr{T}_d$.

Vector-Space Model. In this 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 [8] to compute the query-document similarity under the vector-space model:

$$S_{q,d} = \frac{1}{W_D \times W_q} \times \sum_{t \in \mathscr{T}_{q,d}} \left(1 + \log_e f_{d,t}\right) \times \log_e \left(1 + \frac{N_{\mathscr{D}}}{N_{\mathscr{D}_t}}\right) \tag{1}$$

In Equation (1), $W_D = \left((1.0 - s) + s \times \frac{W_d}{W_{AL}}\right)$ 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 $\mathscr{D}$), respectively. We use the standard value of 0.2 for the slope, which is shown to work well in traditional IR experiments [8].

Probabilistic Model. In IR, the probabilistic models 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. [10]:

$$S_{q,d} = \sum_{t \in \mathscr{T}_{q,d}} w_t \times \frac{(k_1+1)\, f_{d,t}}{K + f_{d,t}} \times \frac{(k_3+1)\, f_{q,t}}{k_3 + f_{q,t}} \tag{2}$$

where $w_t = \log_e \left(\frac{N_{\mathscr{D}} - N_{\mathscr{D}_t} + 0.5}{N_{\mathscr{D}_t} + 0.5} \right)$ 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. The chosen values for k_1, b and k_3 are shown to work well with the TREC Collection experiments [10]. 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. Language modelling in IR 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 [15]:

$$S_{q,d} = f_q \times \log \lambda_d + \sum_{t \in \mathscr{T}_{q,d}} \log \left(\frac{N_{\mathscr{D}} \times f_{d,t}}{\mu \times N_{\mathscr{D}_t}} + 1 \right) \tag{3}$$

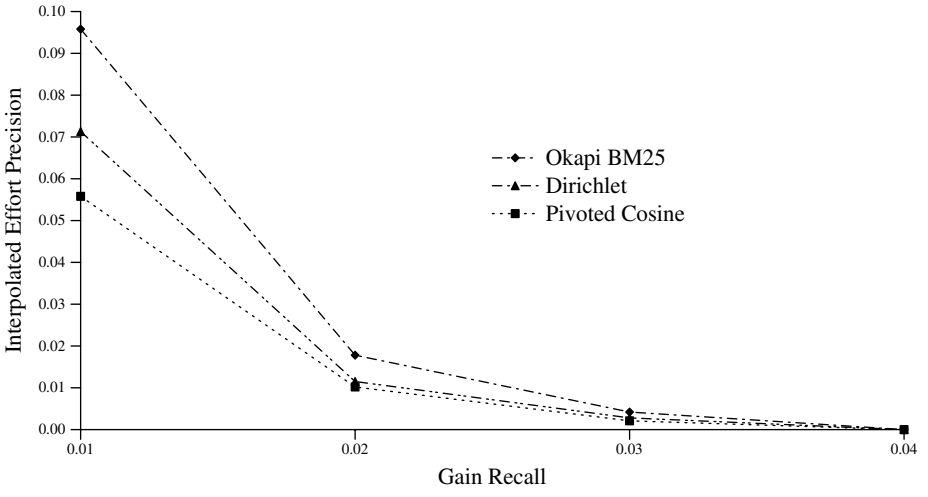
where μ is a smoothing parameter, while λ_d is calculated as: $\lambda_d = \mu / (\mu + f_d)$. We use the value of 2 000 for μ as according to Zhai and Lafferty [15] it is the optimal value used in most IR experiments.

2.3 Performance Results

We now compare the performance of the three similarity measures implemented in Zettair for the *Thorough* and *AIC* tasks of the INEX 2006 Ad Hoc track.[4] We used the information in the `title` element of the topic as the query for Zettair.

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 levels. 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).

[4] Similar relative performance differences between the three similarity measures were also observed on the Focused task of the INEX 2006 Ad Hoc track.



Similarity Measure	MAeP
Okapi BM25	**0.0058**
Dirichlet	0.0052
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

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 BM25 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/gp graphs. Of the other two measures, Dirichlet seems to perform better than the Pivoted Cosine measure. 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 a different XML document collection (IEEE instead of Wikipedia) was used as part of the evaluation testbed [6]. However, 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. Compared to runs submitted by other participants in the INEX 2006 *Thorough* task, all three measures performed relatively poor as our system only returned whole articles (our run using the Okapi BM25 measure was ranked as 82 out of 106 submitted runs).

Table 1 shows that, when using the official evaluation measures for the INEX 2006 *AIC* task, Okapi BM25 again outperforms the other two similarity measures. With the MAgP measure, our run using the Okapi BM25 measure was ranked as fourth out of 56

Table 1. Retrieval performance of the three similarity measures implemented in Zettair on the *AIC* task of the INEX 2006 Ad hoc track

Similarity Measure	MAgP	gP[5]	gP[10]	gP[25]	gP[50]
Okapi BM25	**0.1751**	**0.3766**	**0.3049**	**0.2220**	**0.1566**
Dirichlet	0.1655	0.3266	0.2559	0.1844	0.1372
Pivoted Cosine	0.1489	0.3236	0.2611	0.1830	0.1301

submitted runs in the INEX 2006 *AIC* task. With the measures at rank cutoffs, this run was consistently ranked among the top five best performing runs in the INEX 2006 *AIC* task.

In the next section we describe our research activities 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 topics were organised differently compared to the INEX 2005 MM topics. The INEX 2005 MM topics were only based on the *MMFragments* task, whereas the *MMImages* task was additionally introduced in the INEX 2006 MM track.

Since there are two tasks, the Wikipedia collection has been divided into two sub-collections: Wikipedia Ad Hoc XML collection (`Wikipedia`), which contains XML documents as well as images; and Wikipedia image collection (`Wikipedia_IMG`), which 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 were required 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 contains an image. Therefore, the *MMImages* task essentially represents a document retrieval task, as the only results allowed were full documents (articles) from the XML image collection. The path of each of the resulting answers for this task were in the form of `/article[1]`, so no document fragments 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="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>
```

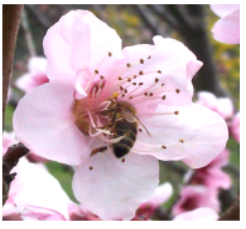


Fig. 2. Example of a *MMImages* query with image ID 60248, a bee and a flower (original in colour)

```
<?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

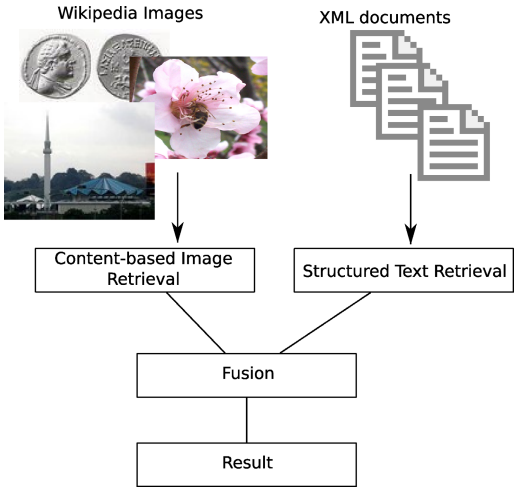


Fig. 4. Conceptual representation of the system (original in colour)

4 Our Approach

In this section, we describe our approach adopted 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),[5] 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 similar approach as the one we used in the INEX 2005 MM track [3]. The only difference is that we now use Zettair instead of the hybrid XML retrieval approach. With Zettair, our officially submitted runs used the Pivoted Cosine similarity measure as it performed best among the three similarity measures in the INEX 2005 Ad Hoc track (using the IEEE document collection) [6]. However, we also performed additional runs to examine the effect of using Okapi BM25 and Dirichlet in the two INEX 2006 MM tasks.

4.1 Content-Based Image Retrieval

The GNU Image Finding Tool was used to retrieve relevant images. The image features from the `Wikipedia_IMG` collection were 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 [11]. For extracting the image texture, a bank of circularly symmetric Gabor filters is used. GIFT evaluates and calculates the query image and the target

[5] `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 used the image references 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.

4.2 Fusing and Ranking the Image and Text Retrieval

To fuse the two retrieval status value (RSV) lists into a single ranked result list R for the multimedia queries, we use a simple linear combination of evidence [1] that is also a form of polyrepresentation [5]:

$$R = \begin{cases} \alpha \cdot S_I + (1-\alpha) \cdot S_T & \text{if the query contains image;} \\ S_T & \text{otherwise.} \end{cases}$$

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 GIFT are used. On the other hand, only the Zettair's RSVs are used when the value of α is set to 0. If there was no image in the query then only the Zettair's RSVs are used, irrespective of the value of α.

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. We then conducted additional runs with the α values ranging between 0.0 to 0.5 to further investigate which α value produces the best retrieval performance. The fusion RSVs of the image and structured text retrieval are then ranked in a descending order of similarity.

5 Experiments and Results

The experiment for the runs was conducted by varying the α values and investigating the retrieval effectiveness of the three similarity measures in Zettair. For each multimedia task, our runs were categorised into two types depending on which INEX 2006 multimedia topic elements were automatically translated as an input query to Zettair:

1. Title runs, which utilise the content of the `title` element; and
2. Extended runs, which utilise the content of the `title`, `castitle`, and `description` elements from each multimedia query.

5.1 Evaluation Metrics

The TREC evaluation metric was adopted to evaluate the *MMImages* task and the evaluation is based on the standard precision and recall retrieval performance measures:

- Mean Average Precision (`MAP`): The mean of the average precisions calculated for each topic. Average precision represents 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%).

For the *MMFragments* task, the EvalJ evaluation software [6] was utilised. We used EvalJ with the following parameters: metrics (ep-gr), overlap (off), quantisation (gen), topic (ALL). The following evaluation measures were used:

- The effort-precision/gain-recall (`ep/gr`) graphs, which provide a detailed view of the run's performance at various gain-recall levels.
- Non-interpolated mean average effort-precision (`MAep`), which provides a single-valued score for the overall run performance. `MAep` is calculated as the average of effort-precision values measured at natural gain-recall levels.

5.2 Multimedia Images Task

For the *MMImages* task we conducted seven Title runs and three Extended runs using each of the IR similarity measures. We varied the α values between 0.0 and 1.0. As the results of the Title runs were promising, we applied a finer variation for the α values between the interval 0.0 and 0.5 (with the step of 0.1) to investigate the best α value for each similarity measure.

In Table 2, we observe that using the `title` element as the query produces better `MAP` and `bpref` performances compared to using the extended query for the *MMImages* task. Among the similarity measures, Dirichlet performed best, and this can be seen in Figure 5 that depicts the interpolated recall/precision averages for all the best runs for each similarity measure.

In the Title runs, having the α values between 0.1 and 0.4 yielded the best `MAP` and `bpref` performance. Using the text retrieval system alone produces better retrieval performance compared to using only the content-based image retrieval system; however the best performance is found when combining evidence and weighting the text retrieval as more important than the content based image retrieval. Comparing the performances between INEX 2005 and INEX 2006 MM track, we observed a similar trend in the α values, where the best α values were in the same range.

Overall, using Dirichlet as the similarity measure produces the best retrieval performance compared to Pivoted Cosine and Okapi BM25 for the *MMImages* task. We also found that the Extended runs performed worse in most cases when compared to the Title runs.

[6] `http://evalj.sourceforge.net`

Table 2. Retrieval performance for the *MMImages* task: mean average precision (MAP) and bpref. *Italic* values – best performance runs using the various α values for each similarity measure **Bold** values – best overall performance among all runs.

Similarity Measure	α value	MAP	bpref
Title runs			
Pivoted Cosine	0.0	0.3054	0.2861
	0.1	*0.3153*	*0.2957*
	0.2	*0.3153*	*0.2957*
	0.3	0.3152	0.2957
	0.4	0.3150	0.2956
	0.5	0.3071	0.2880
	1.0	0.2149	0.2033
Okapi BM25	0.0	0.2679	0.2605
	0.1	0.2686	0.2622
	0.2	*0.2700*	*0.2643*
	0.3	0.2674	0.2599
	0.4	0.2660	0.2572
	0.5	0.2664	0.2592
	1.0	0.1909	0.1814
Dirichlet	0.0	0.3130	0.2973
	0.1	0.3175	0.3014
	0.2	0.3175	0.3014
	0.3	**0.3203**	**0.3034**
	0.4	**0.3203**	**0.3034**
	0.5	0.3202	0.3032
	1.0	0.2158	0.2080
Extended runs			
Pivoted Cosine	0.0	0.2608	0.2307
	0.5	*0.2642*	*0.2366*
	1.0	0.2071	0.1926
Okapi BM25	0.0	*0.2674*	0.2369
	0.5	*0.2674*	*0.2464*
	1.0	0.2087	0.2002
Dirichlet	0.0	0.2988	0.2787
	0.5	*0.3094*	*0.2805*
	1.0	0.2147	0.1987

5.3 Multimedia Fragments Task

For the *MMFragments* task, we conducted six runs using the Pivoted Cosine, Okapi BM25 and Dirichlet similarity measures. We used the default value of $\alpha = 0.0$ for all the runs (since for this task we only used the text retrieval system).

We observe an opposite behaviour for the Title and Extended runs for this task. As reflected from the ep/gr graphs in Figure 6, the Extended runs performed better than the Title runs. This shows that the presence of the title, castitle and description from the query improves the retrieval performance when compared to only using the

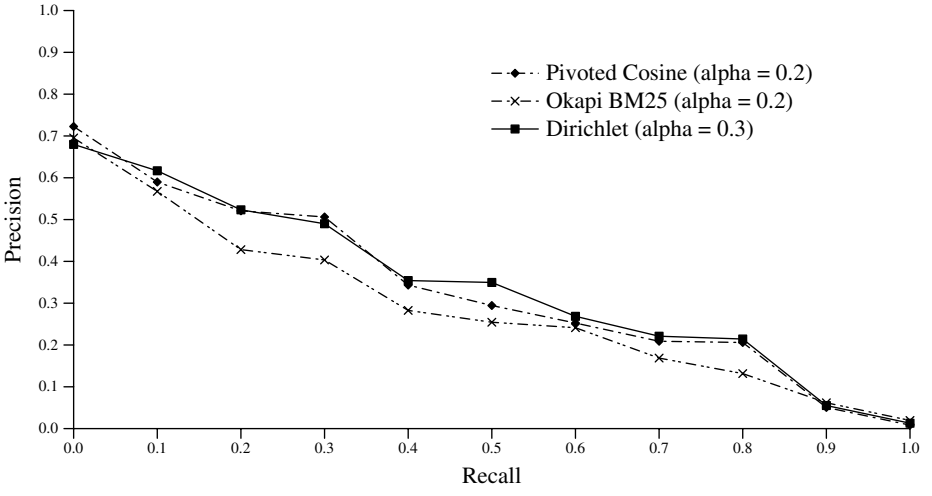


Fig. 5. Interpolated precision averages at eleven standard recall levels for the Title runs of the *MMImages* task

`title` element of the MM queries in the *MMFragments* task. This result also reflects the nature of the task, where XML fragments need to be returned as the retrieved answers.

When comparing the retrieval performance of the three IR similarity measures, we observe that Dirichlet once again outperformed Pivoted Cosine and Okapi BM25. This can also be seen in Figure 6 and from the overall `MAep` scores presented in Table 3.

To investigate whether combining evidence from the CBIR system improves the retrieval performance for this task, we conducted several preliminary runs that fuse the RSVs from the CBIR system and Zettair. This resulted in a minor performance improvement. However, without better fragment retrieval system, we cannot conclude whether combining text and image RSVs will improve retrieval performance for the *MMFragments* task.

6 Conclusions and Future Work

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 combined this with a content-based image retrieval system for the MM track.

For the Ad Hoc track, Okapi BM25 similarity measure produced the best retrieval performance for the *Thorough* and *AIC* tasks.

For the two XML multimedia retrieval tasks, we officially submitted six runs using the Pivoted Cosine similarity measure. We also conducted additional runs to investigate the effectiveness of the Okapi BM25 and Dirichlet similarity measures. The runs for the *MMImages* task reflect the various relative weights of 0.0 to 1.0 for the α values. We found that Dirichlet was the best similarity measure for the *MMImages* task, and that α values between 0.1 and 0.4 produced the best retrieval performance. For the

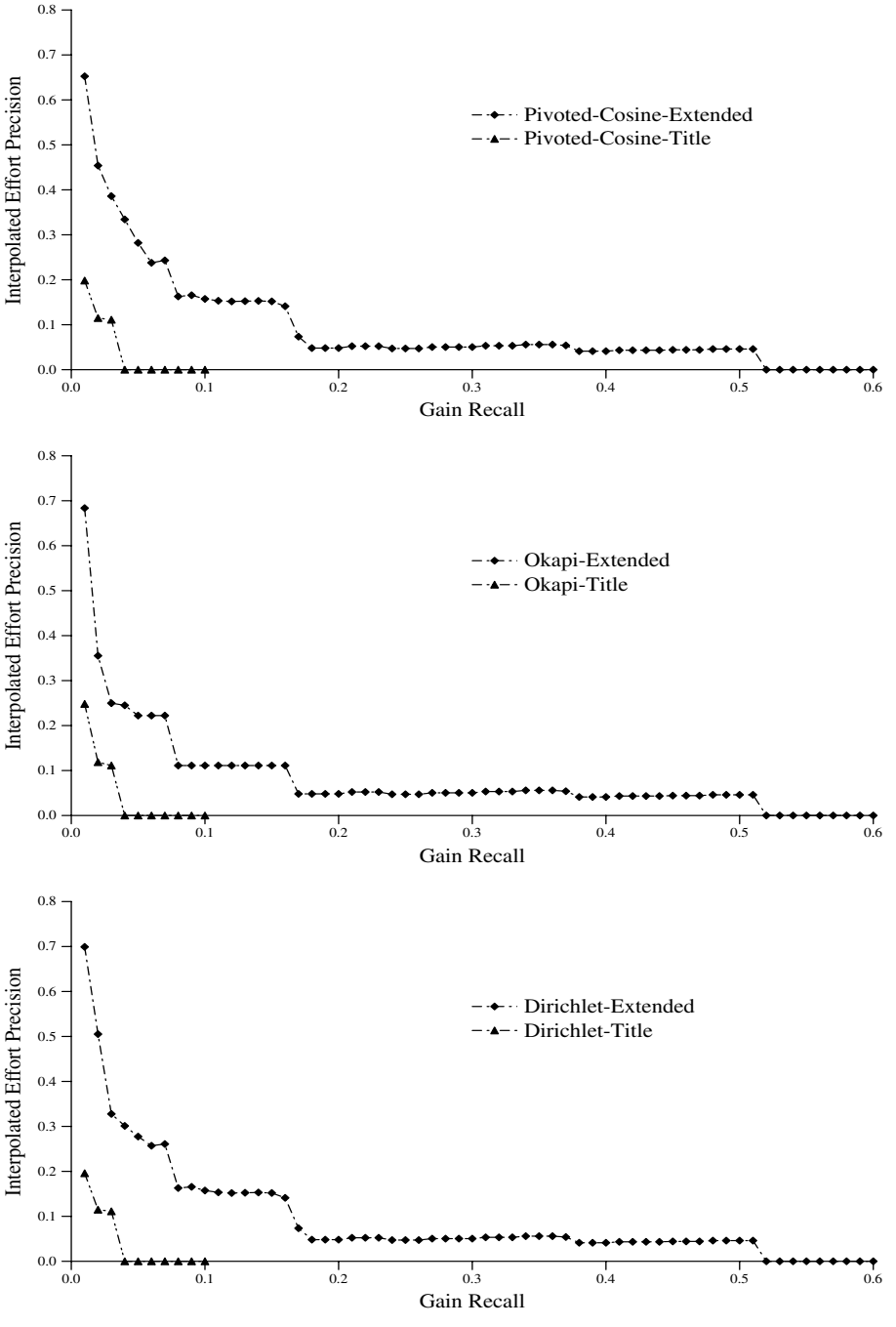


Fig. 6. Interpolated effort-precision averages at standard gain-recall levels for the Title and Extended runs of the *MMFragments* task, using Pivoted Cosine (top), Okapi BM25 (middle), and Dirichlet (bottom) similarity measures in Zettair

Table 3. Retrieval performance of the Extended runs on the *MMFragments* task

Similarity Measure	MAep
Pivoted Cosine	0.0655
Okapi BM25	0.0586
Dirichlet	**0.0663**

MMFragments task, the official runs were only based on the text retrieval system. We executed four additional runs using Okapi BM25 and Dirichlet similarity measures. As for the *MMImages* task, Dirichlet was also found to be the best among the three similarity measures used in the *MMFragments* task.

We have used the linear combination of evidence to merge the RSVs from two retrieval subsystems for retrieving multimedia information. We conclude that a text retrieval system benefits by using some evidence from a CBIR system. More specifically, giving more weight to text retrieval system RSVs in the fusion function yields better performance than when the two subsystems are used on their own.

This work can be extended in two ways. First, to cater for the *MMFragments* task more effectively, the hybrid XML retrieval approach [7] can be used as the content-oriented XML retrieval system. Second, it would also be interesting to fuse the RSVs from CBIR and text systems with the 101 image concepts such as those provided by the University of Amsterdam [9].

Acknowledgments. This research was undertaken using facilities supported by the Australian Research Council, an RMIT VRII grant, and a scholarship provided by the Malaysian Ministry of Higher Education.

References

1. Aslandogan, Y.A., Yu, C.T.: 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, pp. 313–321. ACM Press, New York, USA (2000)
2. Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.): Advances in XML Information Retrieval and Evaluation, In: 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28–30, 2005, Revised Selected Papers, vol. 3977 of Lecture Notes in Computer Science. Springer (2006)
3. Awang Iskandar, D. N. F., Pehcevski, J., Thom J. A., Tahaghoghi, S. M. M.: Combining image and structured text retrieval. In Fuhr et al.: [2] pp. 525–539.
4. Kazai, G., Lalmas, M.: INEX 2005 evaluation measures. In: Fuhr et al.: [2] pp. 16–29.
5. Larsen, B., Ingwersen, P., Kekäläinen, J.: The polyrepresentation continuum in IR. In: IIiX: Proceedings of the 1st international conference on Information interaction in context, pp. 88–96. ACM Press, New York (2006)
6. Pehcevski, J., Thom, J.A., Tahaghoghi, S.M.M.: RMIT University at INEX: Ad Hoc Track. In: Fuhr. et al.: [2] pp. 306–320 (2005)
7. Pehcevski, J., Thom, J.A., Vercoustre, A-M.: Hybrid XML retrieval: Combining information retrieval and a native XML database. Information Retrieval 8(4), 571–600 (2005)

8. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: Proceedings of the ACM-SIGIR International Conference on Research and Development in Information Retrieval, Zurich, Switzerland, pp. 21–29. ACM Press, New York (1996)
9. Snoek, C.G.M., Worring, M., Gemert, J.C.V., Geusebroek, J., Smeulders, A.W.M.: The challenge problem for automated detection of 101 semantic concepts in multimedia. In: MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia, pp. 421–430. ACM Press, New York, USA (2006)
10. SparckJones, K., Walker, S., Robertson, S.E.: A probabilistic model of information retrieval: Development and comparative experiments. Parts 1 and 2. Information Processing and Management 36(6), 779–840 (2000)
11. Squire, D.M., Müller, W., Müller, H., Pun, T.: Content-based query of image databases: Inspirations from text retrieval. Pattern Recognition Letters 21(13–14), 1193–1198 (special edition for SCIA'99) (2000)
12. Tjondronegoro, D., Zhang, J., Gu, J., Nguyen, A., Geva, S.: Integrating text retrieval and image retrieval in XML document searching. In: Fuhr. et al.: [2], pp. 511–524
13. van, R., Zwol.: Multimedia strategies for b^3 -sdr, based on principal component analysis. In Fuhr, et al.: [2], pp. 540–553
14. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes: Compressing and Indexing Documents and Images, 2nd edn. Morgan Kaufmann Publishers, San Francisco (1999)
15. 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)
16. Zobel, J., Moffat, A.: Exploring the similarity space. ACM SIGIR Forum 32(1), 18–34 (1998)

XFIRM at INEX 2006. 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.

1 Introduction

In this paper, we decribe the IRIT/SIG-RFI participation in the INEX 2006 ad-hoc, relevance feedback and multimedia tracks. Our participation is based on the XFIRM system [10], which uses a relevance propagation method.

In the *ad-hoc track* (section 2), we propose to evaluate the algorithms proposed part years [10] on a new collection (the Wikipedia collection) and on new searching tasks (Best in Context and All in Context tasks). We also compare results obtained using structural hints of queries with results obtained with simple keywords terms.

For the *relevance feedback track* and for our second participation to the track (section 3), we propose to apply a Content-Oriented approach inspired from the probabilistic theory and to improve the Structure-Oriented approach presented in [10].

At last, for our first participation to the *multimedia track* (section 4), we describe a context-based approach for multimedia retrieval. This approach uses the text surrounding images and structure of documents to judge the relevance of images or XML components.

2 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 at INEX 2005 [10]. Results presented here are mainly official results and further experiments need to be done to confirm results obtained with this preliminary work.

2.1 The XFIRM Model

The model is based on a relevance propagation method. During query processing, relevance scores are first computed at leaf nodes level. Then, inner nodes

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 373–386, 2007.

relevance is evaluated by doing a propagation of leaf nodes scores through the document tree. An ordered list of subtrees is then returned to the user.

Processing of Content-Only queries. Let $q = t_1, \ldots, t_n$ be a content-only query. Relevance values are computed using the similarity function $RSV(q, ln)$.

$$RSV(q, ln) = \sum_{i=1}^{n} w_i^q * w_i^{ln}, \quad where \quad w_i^q = tf_i^q \quad and \quad w_i^{ln} = tf_i^{ln} * idf_i * ief_i \quad (1)$$

Where w_i^q and w_i^{ln} are the weights of term i in query q and leaf node ln respectively. tf_i^q and tf_i^{ln} are the frequency of i in q and ln respectively, $idf_i = log(|D|/(|di| + 1)) + 1$, with $|D|$ the total number of documents in the collection, and $|di|$ the number of documents containing i, and ief_i is the inverse element frequency of term i, i.e. $log(|N|/|nf_i| + 1) + 1$, where $|nf_i|$ is the number of leaf nodes containing i and $|N|$ is the total number of leaf nodes in the collection.

Each node in the document tree is then assigned a relevance score which is function of the relevance scores of the leaf nodes it contains and of the relevance value of the whole document.

$$r_n = \rho * |L_n^r|. \sum_{ln_k \in L_n} \alpha^{dist(n, ln_k)-1} * RSV(q, ln_k) + (1 - \rho) * r_{root} \quad (2)$$

$dist(n, ln_k)$ is the distance between node n and leaf node ln_k in the document tree, i.e. the number of arcs that are necessary to join n and ln_k, and $\alpha \in]0..1]$ allows to adapt the importance of the $dist$ parameter. $|L_n^r|$ is the number of leaf nodes being descendant of n and having a non-zero relevance value (according to equation 1). $\rho \in]0..1]$, inspired from work presented in [6], allows the introduction of document relevance in inner nodes relevance evaluation, and r_{root} is the relevance score of the $root$ element, i.e. the relevance score of the whole document, evaluated with equation 2 with $\rho = 1$.

Processing of CO+S queries. The evaluation of a CO+S query is carried out with the following steps:

1. INEX (NEXI) queries are translated into XFIRM queries
2. XFIRM queries are decomposed into sub-queries SQ and elementary sub-queries ESQ, which are of the form: $ESQ = tg[q]$, where tg is a tag name, i.e. a structure constraint, and $q = t_1, ..., t_n$ is a content constraint composed of simple keywords terms.
3. Relevance values are then evaluated between leaf nodes and the content conditions of elementary sub-queries
4. Relevance values are propagated in the document tree to answer to the structure conditions of elementary sub-queries

5. Sub-queries are processed using the results of elementary sub-queries
6. Original queries are evaluated by doing an upwards and downwards propagation of the relevance weights

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

$$r_n = \begin{cases} \sum_{ln_k \in L_n} \alpha^{dist(n,ln_k)-1} * RSV(q, ln_k) \ if \ n \ \in construct(tg) \\ 0 \ otherwise \end{cases} \quad (3)$$

where the $construct(tg)$ function allows the creation of set composed of nodes having tg as tag name, and $RSV(q, ln_k)$ is evaluated during step 2 with formula 1. The $construct(tg)$ function uses a *Dictionary* Index, which provides for a given tag tg the tags that are considered as equivalent. This index is built manually.

More details about CO and CO+S queries processing can be found in [10].

2.2 Runs and Results

For all retrieval strategies, runs were submitted using content-only conditions of topics (CO runs) and content and structure conditions (CO+S runs). Results are presented with the official metrics.

Thorough task. For the Thorough task, all nodes having a non-zero relevance value are returned by the XFIRM system. Results are described in Table 1.

Table 1. Thorough task. Generalised quantisation function with filtered assessments. Official runs are in bold.

Queries	Parameters	nxCG[5]	nxCG[10]	nxCG[25]	nxCG[50]	ep/gr MAP
CO	$\alpha = 0.6, \rho = 1$, **eq. 2**	**0.2161**	**0.1822**	**0.1470**	**0.1235**	**0.01831**
CO	$\alpha = 0.6, \rho = 0.9$, **eq. 2**	**0.2167**	**0.1885**	**0.1581**	**0.1526**	**0.0241**
CO	$\alpha = 0.1, \rho = 1$, eq. 2	0.228	0.2002	0.1703	0.1367	0.0230
CO	$\alpha = 0.1, \rho = 0.9$, eq. 2	0.2289	0.2005	0.1763	0.1447	0.0266
CO	$\alpha = 0.2, \rho = 1$, eq. 2	0.2318	0.2117	0.1766	0.1458	0.02352
CO	$\alpha = 0.2, \rho = 0.9$, eq. 2	0.2413	0.2178	0.1857	0.1555	0.02779
CO+S	$\alpha = 0.9$, **eq. 3**					**0.0089**

2.3 Focussed Task

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. Results are described in Table 2.

Table 2. Focussed task. Generalised quantisation function with filtered assessments. All runs are official.

Queries	Parameters	nxCG[5]	nxCG[10]	nxCG[25]	nxCG[50]
CO	$\alpha = 0.6, \rho = 1$, eq. 2	0.1333	0.1104	0.0853	0.0656
CO+S	$\alpha = 0.9$, eq. 3	0.1839	0.1456	0.1015	0.0879
CO+S	$\alpha = 0.8$, eq. 3	0.1820	0.147	0.1088	0.0906

2.4 All in Context Task

We use two different retrieval strategies:

1. Relevance values are computed for each node of the collection according to the focussed stategy and nodes are then ranked by article (we rank first the top ranked node according to equation 2 and then all the nodes having a non-zero relevance value belonging to the same document, and so on)
2. Nodes are first ranked by the relevance of the document they belong to (according the the Mercure system [1]), and then by their own relevance (according to equation 2 used in a focussed strategy).

Official results are described in Table 3, and additional results at element level are provided in Table 4.

Table 3. All in Context task. Generalised quantisation function. All runs are official.

Queries	Parameters	MAgP	gP[5]	gP[10]	gP[25]	gP[50]
CO	$\alpha = 0.6, \rho = 1$, eq. 2, 2nd strategy	0.0683	0.2136	0.1810	0.1359	0.0950
CO	$\alpha = 0.1$, eq. 3, 1st strategy	0.0388	0.1159	0.1042	0.0804	0.0633

Table 4. All in Context task. Element level. All runs are official.

Queries	Parameters	F-avg INTERSECTION	F-avg UNION
CO	$\alpha = 0.6, \rho = 1$, eq. 2, 2nd strategy	0.5050	0.1632
CO	$\alpha = 0.1$, eq. 3, 1st strategy	0.4066	0.1723
CO+S	$\alpha = 0.9$, eq. 3, 2nd strategy	0.4473	0.1632

Best in Context task. 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. Official results are described in table 5.

Table 5. Best in Context task. All runs are official.

Queries	Parameters	BEPD A=0.01	BEPD A=0.1	BEPD A=1	BEPD A=10	BEPD A=100
CO	$\alpha = 0.6, \rho = 1$, eq. 2	0.1082	0.1685	0.2505	0.3586	0.4466
CO	$\alpha = 0.9$, eq. 3	0.0723	0.1215	0.2037	0.3441	0.5086
Queries	Parameters	EPRUM A=0.01	EPRUM A=0.1	EPRUM A=1	EPRUM A=10	EPRUM A=100
CO	$\alpha = 0.6, \rho = 1$, eq. 2	0.0120	0.0269	0.0518	0.0843	0.1230
CO	$\alpha = 0.9$, eq. 3	0.0091	0.0177	0.0340	0.0661	0.1202

Discussion. For all strategies, further experimentations are needed, in order to see whether conclusions drawn with the IEEE collection and past metrics are validated or not. However, we can already say by analysing additional results of the Thorough strategy that:

- document relevance seems to be important when evaluating nodes relevance,
- and very specific nodes are preferred by users,

this validates past years results.

The use of structural constraints in queries do not improve results, this contradicts results found in [9]. This can be explained by the small equivalencies dictionary used for tags: it should be extended before drawing conclusions.
Results obtained with the Focussed strategy and Best In Context tasks are not as good as expected and parameters tuning need to be done.

At last, although our system was ranked first at element level, we were disappointed with the results obtained using the official metrics for the All In Context task. We need to further investigate this matter.

3 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 applied a Content-Oriented approach inspired from the probabilistic theory and to improve our Structure-Oriented approach already presented in [10].

3.1 Content-Oriented Relevance Feedback

Probabilistic method
Simple term extraction (without query terms re-weighting) using the Rocchio's algorithm [7] has already been explored and did not show any improvement [10].

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 $P(t_i/E^r)$. We estimate this probability according to a simplified version of the Robertson's formula [3]:

$$P(t_i/E^r) = r_e/R_e \tag{4}$$

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 $Q = t_1, ..., t_n$ be the initial query. If we choose to add 3 relevant terms to the initial query, the new query will be:
Q'=$(t_1, 1), ..., (t_n, 1), (t_o, w_o), (t_p, w_p), (t_r, w_r)$ where t_o, t_p and t_r are the added terms and w_o, w_p and w_r their associated weights.

3.2 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 reminded 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/bdy/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 ∃e_p(p_p, w_p) ∈ CS/p_p = p_i then
                    w_p ← w_p + w_j
                    return true
          else CS ← e_j(p_j, w_j + w_i)
                    return true
   else
          if head(p_j) ≠ null then
                    p'_j ← head(p_j)
                    w'_j ← 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/bdy/section)=/article/bdy.*

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 (CO+S) 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 $Q1 = t_1, t_2, ...t_n$ 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: $Q1' = S1[t_1, t_2, ...t_n]\ OR\ S2[t_1, t_2, ...t_n]$.

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 4. Terms are added to the original query terms with their associated weights.

Example
Let $Q1 = t_1, t_2, ..., t_n$ 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): $Q1' = S1[(t_1, 1), ..., (t_n, 1), (t_o, w_o), (t_p, w_p)]$ OR $S2$ $[(t_1, 1), ..., (t_n, 1), (t_o, w_o), (t_p, w_p)]$

3.3 Runs

CO.thorough task
For official runs and according to previous experiments, we use

- the Content-Oriented RF approach by adding 3 or 10 relevant terms to the initial query, respectively named in Table 6 by *CO-C3* and *CO-C10*.
- the combined approach with 3 expressive terms and 3 generative structures selected according to the SCA algorithm, and named by *CO-C3S3* in Table 6.

The top 20 elements of each query is used to select relevant terms / relevant structures. We use the base run of CO queries obtained with XFIRM using $\alpha = 0.6$ and $\rho = 1$.

We present in the following table the Absolute Improvement (AI) and the Relative Improvements (RI) according the MAep metric. In Relevance Feedback track, all evaluations are preliminary; official evaluations are not yet available.

Table 6. Impact of Content and Combines-Oriented RF in CO queries

	CO-C3	CO-C10	CO-C3S3
AI(MAep)	-0.00645	-0.00653	-0.00332
RI-(MAep)	-64.67%	-65.48 %	-33.34%

We notice in Table 6 that there is no improvement (negative values of RI). When comparing the columns we can see that the number of added expressive terms does not have a significant impact (CO-C3 and CO-C10). This can be explained by the fact that the suitable number of terms to be added depends on the query length [8].

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 designated respectively in Table 7 by *CO-J10* and *CO-J40*. We also propose to consider a fixed number of relevant elements to extend queries (4 strictly relevant elements are used in this paper) designated by *CO-R4*. We use the base run of CO queries using $\alpha = 0.6$ $\rho = 0.9$. We apply the Content-Oriented RF with adding 3 expressive terms. According to the Table 7, we do not see any improvement for all runs. However, we still think that it is important to choose appropriate strategies of Relevance Feedback. Indeed, we obtained

Table 7. Impact of RF strategies in CO queries

	CO-J10	CO-J40	CO-R4
AI(MAep)	-0.0087	-0.00734	-0.00722
RI-(MAep)	-0.5363%	-48.74%	-47.98%

contradictory results in previous experiments, where a fixed number of relevant elements improved the effectiveness of the Retrieval system [5]. For example, on the INEX 2005 collection, the Relative Improvement for the MAep strict metric was about 25%, whereas no improvement can be observed with the traditional strategy.

CO+S thorough task
For this task we evaluated the content-oriented RF using 3 and 5 terms and the combined approach using 3 expressive terms and 1 generative structure respectively designated by *COS-C3*, *COS-C5* and *COS-C3S* in Table 8. We used the base run of CO+S queries with $\alpha = 0.9$ In this task, we notice that the two ap-

Table 8. Impact of Content and Combines-Oriented RF in CO+S queries

	COS-C3	COS-C5	COS-C3S
AI(MAep)	0.0008	0.00118	0.00205
RI-(MAep)	18.27%	24.42%	42.36%

proaches of Relevance feedback are efficient and the improvement is about 48% when we apply the Combined approach. Moreover, we notice that the addition of 5 terms is more efficient than the addition of 3 terms to initial query.

4 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 [11] : (1) Context Based Image Retrieval and (2) Content Based Image Retrieval:

1. The context of an image is all information about the image issued from other 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.

4.1 Runs

MMImages topics processing (MMI method): 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 documents having a score > 0 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).

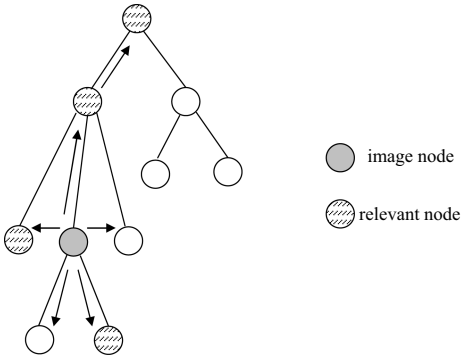


Fig. 1. Use of ancestor, brother and descendant nodes to evaluate images relevance

An image score corresponding to each of the preceding sources of evidence is computed:

- W_d^{im} is the image score computed using descendant nodes,
- W_b^{im} is the image score computed using brother nodes,
- W_a^{im} is the image score computed using ancestor nodes,

The total image score is then expressed as follows:

$$W_{im} = p_1.W_d^{im} + p_2.W_b^{im} + p_3.W_a^{im} \tag{5}$$

where p_1, p_2 and p_3 are parameters used to emphasize some weights types and $p_1 + p_2 + p_3 = 1$.

With this method, all the images of the relevant documents are evaluated and will have a score > 0. Indeed, they will inherit at least of the root node score W_a^{im}. We detail the evaluation of each score in the following paragraphs.

To evaluate the score of an image using its descendant nodes, we use the score 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 ($|\overline{d}|$).

$$W_d^{im} = f(W_{rdi}, |d|, |\overline{d}|) \tag{6}$$

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 score evaluation. Using this intuition, we apply the following formula in our experiments.

$$W_d^{im} = (\frac{|d|+1}{|\overline{d}|+1}) * \sum_{i=1}^{|d|} W_{rdi} \tag{7}$$

To evaluate the score of an image using its brother nodes, we use the score of each relevant brother node 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 $|\overline{b}|$

$$W_b^{im} = f(W_{rbi}, dist(im, b_i), |b|, |\overline{b}|) \tag{8}$$

Figure 2 shows how distances between an image node and its brother nodes are calculated:

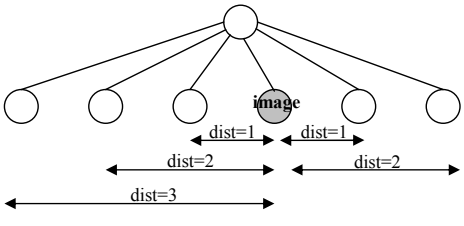


Fig. 2. distance between image node and brother nodes

The formula used in experiments presented here is :

$$W_b^{im} = (\frac{|b|+1}{|\overline{b}|+1}) * (\sum_{i=1}^{|b|} \frac{W_{rbi}}{dist(im, b_i)}) \tag{9}$$

To evaluate the score of an image using its ancestor nodes, we add the scores of relevant ancestor nodes 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 scores 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:

$$W_a^{im} = \sum_{i=1}^{|a|} \frac{log(W_{rai} + 1)}{dist(im, a_i) + 1} \tag{10}$$

where $|a|$ is the number of relevant ancestor nodes according to the XFIRM model.

MMFragments topics processing (MMF method): For MMFragments topics, we adapted the XFIRM CO+S method: we decomposed the query into sub-queries (figure 3). For each sub-query, if its element is different from "image", we applied the XFIRM COS method and if the subquery element is "image", we applied the MMI method. Then, we propagated scores of sub-queries to the target element using the XFIRM CO+S method.

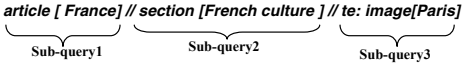


Fig. 3. Query decomposition in sub-queries

4.2 Results

MMFragments task results are based on 9 topics, whereas MMImages task results are based on 13 topics.

MMImages task results. Two metrics are used to evaluate MMImages topics: MAP (Mean Average Precision) and BPREF (Binary PREFerence). Results are presented in table 9:

We used 4 methods: XFIRM CO method, XFIRM CO+S method, MMI method and MMF method. Grayed boxes are results of our official runs. Best MAP is 0.2254 using XFIRM COS method with parameters α=0.9 and "image" as query target element.

Results for the MAP metric with the XFIRM CO+S method are better than results with the MMI method. This can be explained by the structure of the images collection (Figure4). The MMI method is based on the place of textual information in the structure (ascendants, descendants, brothers), whereas in the MMImages collection, images do not have decendant nodes and the ancestor nodes scores are only calculated with brother nodes. All the textual content of the document is thus in the brother nodes.

Best BPREF is 0.2225 using MMF method with parameter α=0.6, $p_1 = 0 p_2 = 1 p_3 = 0$ and "image" as target element.

Table 9. MMImages task results

Method	α	ρ	p_1	p_2	p_3	MAP	BPREF
CO	0.6	1	-	-	-	0.1140	0.1394
CO+S	0.9	-	-	-	-	**0.2254**	0.2060
MMI	0.6	1	0.33	0.33	0.33	0.2122	0.2065
MMI	0.6	0.9	0.33	0.33	0.33	0.2159	0.2078
MMF	0.6	-	0.33	0.33	0.33	0.2114	**0.2221**
MMF	0.6	-	0	1	0	0.2142	0.2225
MMI	0.6	0.9	0	0	1	0.2112	0.2078

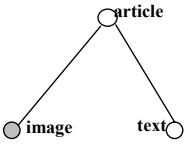


Fig. 4. MMImages collection structure

MMFragments task results. Table 10 shows results of the MMFragments task. They are based on the Mean Average Precision (MAP) metric. We tested Multimedia methods and XFIRM methods where no specification of images is done.

Table 10. MMFragments task results

Method	α	ρ	dict	p_1	p_2	p_3	MAP
CO	0.6	1	2	-	-	-	0.008599
CO	0.6	0.9	2	-	-	-	0.015488
CO+S	0.9	-	1	-	-	-	0.010142
CO+S	0.9	-	2	-	-	-	0.01079
MMF	0.5		2	0.33	0.33	0.33	0.01596
MMI	0.1	0.9	2	0.33	0.33	0.33	**0.01765**
MMI	0.1	0.9	2	1	0	0	0.000124
MMI	0.1	0.9	2	0	1	0	0.0084
MMI	0.1	0.9	2	0	0	1	0.01575

Grayed boxes are results of our official runs. Runs using the CO+S query processing model of the XFIRM system use two different dictionary indexes: *dict*1 contains simple tag equivalencies (for example, *image* and *figure* are considered as equivalent tags), whereas *dict*2 contains very extended equivalencies between tags.

The best MAP is obtained with the MMI method, where the target element is always an image node. We tested some values of the 3 parameters: $p_1, p_2 etp_3$. We can observe that using only descendant nodes gives worst results whereas using only ancestor nodes gives good results. All document context seems thus

to contribute to the image relevance. This can be explained by the wealth of other nodes information belonging to the document that are likely to share the same subject as images.
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. Boughanem, M., Dkaki, T., Mothe, J., Soule-Dupuy, C.: Mercure at TREC-7. In: Proceedings of TREC-7 (1998)
2. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. SIGIR Forum (2006)
3. Robertson, S.E., Sparck-Jones, J.: Relevance weighting of search terms. Journal of the American Society for Information Science, pp. 129–146 (1976)
4. Fuhr, N., Lalmas, M., Malik, S., Kazai, G.: In: INEX 2005 workshop proceedings (2005)
5. Hlaoua, L., Pinel-Sauvagnat, K., Boughanem, M.: Relevance feedback for xml retrieval: using structure and content to expand queries. In: Proceedings of RCIS 2007, Ouarzazate, MOROCCO, to appear, (April 2007)
6. Mass, Y., Mandelbrod, M.: Experimenting various user models for XML retrieval. In: [4] (2005)
7. Rocchio, J.: Relevance feedback in information retrieval. Prentice Hall Inc, Englewood Cliffs, NJ (1971)
8. Ruthven, I., Lalmas, M.: Selective relevance feedback using term characteristics. CoLIS 3. In: Proceedings of the Third International Conference on Conceptions of Library and Information Science (1999)
9. Sauvagnat, K., Boughanem, M., Chrisment, C.: Why using structural hints in XML retrieval? In: Flexible Query Answering (FQAS) 2006, Milano, Italia. Advances in Artificial Intelligence, 7–10 (June 2006)
10. Sauvagnat, K., Hlaoua, L., Boughanem, M.: Xfirm at inex 2005: ad-hoc and relevance feedback track. In: INEX 2005 Workshop proceedings, pp. 88–103 (2005)
11. Thijs, W.: Image retrieval: Content versus context. In: RIAO, pp. 276–284 (2000)

The Interactive Track at INEX 2006

Saadia Malik[1], Anastasios Tombros[2], and Birger Larsen[3]

[1] University Duisburg-Essen, Germany
[2] Queen Mary University of London, UK
[3] Royal School of Library and Information Science, Denmark
saadia.malik@uni-due.de, tassos@dcs.qmul.ac.uk, blar@db.dk

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 INEX 2006 workshop, the track will continue into 2007. Special emphasis is put on comparing XML element retrieval with passage retrieval, and on investigating differences between multiple dimensions of the search tasks.

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 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 Methodology[1]). Finally, we have attempted to ease the burden of experimenters and searchers by an online experimental 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

[1] See http://www.cs.otago.ac.nz/sigirmw/ for the proceedings and presentation slides, and in particular the paper by Trotman & Geva.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 387–399, 2007.

per searcher. Due to, among other things, a complex system setup, the track has been delayed and data collection was not completed before the INEX 2006 workshop. The track will therefore continue into 2007.

1.1 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 four 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 8 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 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 are tested: one based on a Passage retrieval backend[2] and one on an Element retrieval backend[3]. 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

[2] The Passage retrieval backend runs on CSIRO's Panoptic™/Funnelback™ platform. See http://www.csiro.au/csiro/content/standard/pps6f,,.html for more information.

[3] The Element retrieval backend runs on Max Planck Institute for Informatics' TopX platform. See [6] for more information.

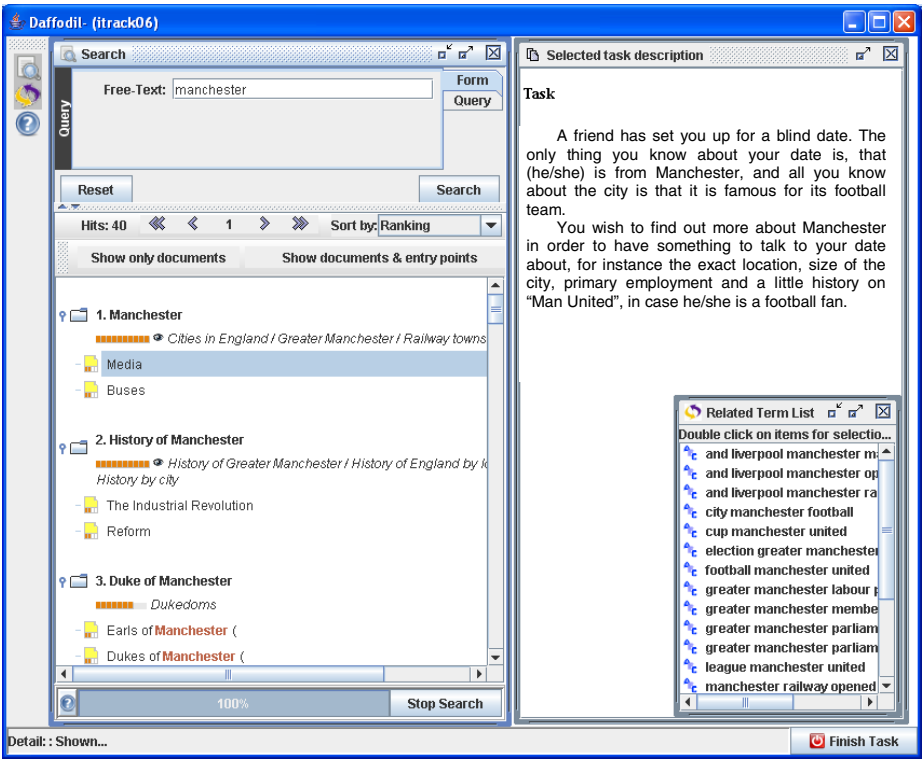


Fig. 1. Selected task, query box and result list from the INEX2006 interactive track system. (Modified version of the Daffodil Digital Library system [2]).

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.

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 highlighting 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 ( ).

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 on suggestions put forward by Gheorghe Muresan from Rutgers University, USA (and also used in Task C of last

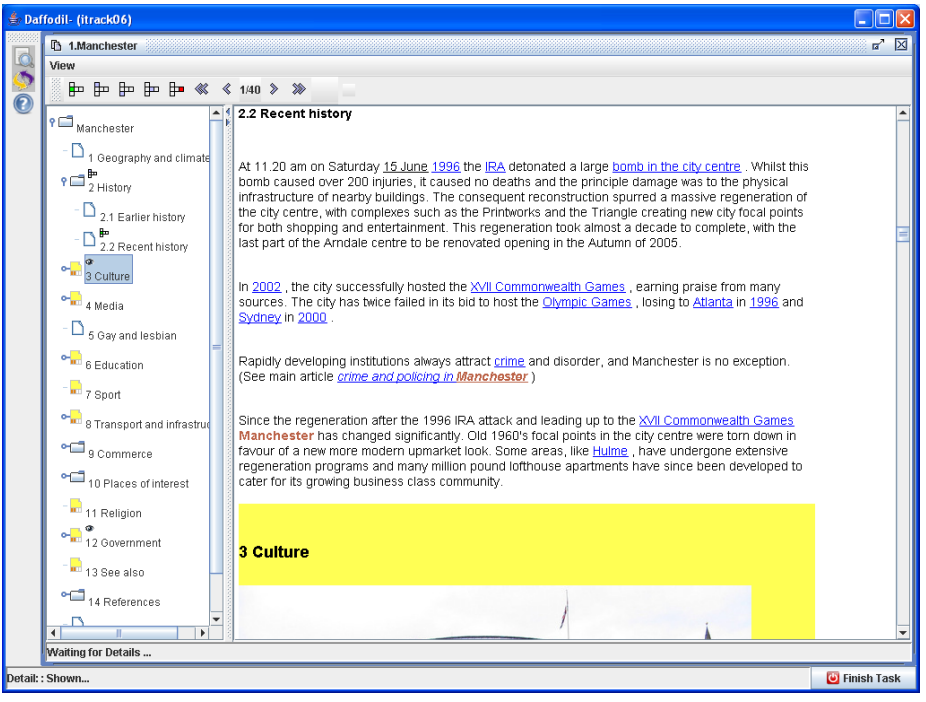
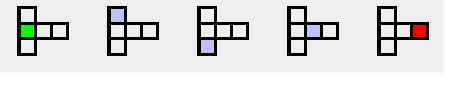


Fig. 2. Detail view of a document from the INEX2006 interactive track system. (Modified version of the Daffodil Digital Library system [2]).

year's interactive track). This scale, and its visualisation in the T-shape, balance 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 *just right*, *too large* or *too small*.

This is combined into the five scores illustrated as:

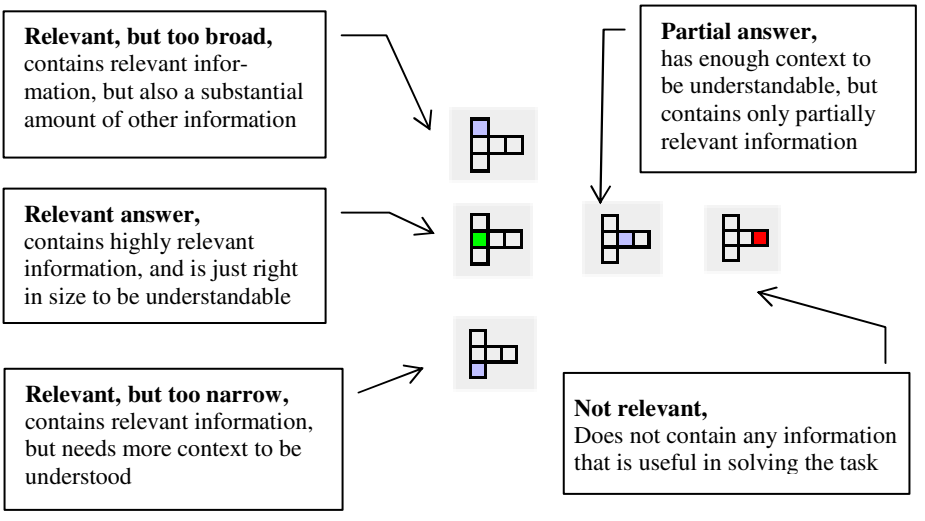


Fig. 3. INEX2006 interactive track relevance assessment scale based on Pehcevski and Thom [5]

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]. Note that in contrast to the assessments made for the ad hoc track, there is no requirement for 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 for 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 comprises 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 to participants 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.

2.5 Search Tasks

For the 2006 interactive track we have chosen to put more emphasis on investigating the effect of different search task types [7,8]. 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 twelve 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 twelve tasks are split into four categories allowing the searchers a choice between two tasks, and at the same time ensuring that each searcher will perform at least 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 15 minutes will apply. Sessions can finish before this if searchers feel they have completed the task.

2.6 Experimental Control System

We 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.

2.7 Experimental Matrix

A minimum of 8 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 8 searchers.
For information, the basic experimental matrix looks as follows:

	S1	**S1**	**S2**	**S2**
Rotation 1	C1	C2	C3	C4
Rotation 2	C2	C1	C4	C3
Rotation 3	C3	C4	C1	C2
Rotation 4	C4	C3	C2	C1

	S2	**S2**	**S1**	**S1**
Rotation 5	C8	C7	C6	C5
Rotation 6	C7	C8	C5	C6
Rotation 7	C6	C5	C8	C7
Rotation 8	C5	C6	C7	C8

Where:
Element (S1) vs. Passage (S2) systems
C1 to C8 are categories of three tasks each.

The tasks are distributed in categories as follows (see Appendix A for the tasks themselves):

Category	Tasks
C1	1,2,3
C2	5,6,7
C3	9,10,11
C4	4,8,12

Category	Tasks
C5	2,3,4
C6	6,7,8
C7	10,11,12
C8	1,5,9

These rotations are related to the searcher logins and the control system handles their administration.

2.8 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 15 minutes, with reminder by the system.*
8. *After-each-task questionnaire administered*
9. Steps 5-8 repeated for the three 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 the study is carried out smoothly. 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

In the INEX2006 interactive track we put special emphasis on comparing XML element retrieval with passage retrieval, and on investigating differences between multiple dimensions of the search tasks. In addition, a major change is the move from the corpus of IEEE CS journal articles to the Wikipedia XML corpus of encyclopaedia articles. As the track has been delayed and data collection was not completed before the INEX 2006 workshop, the track will continue into 2007. At the time of writing 10 groups are active in the track (See appendix B) which will result in data being collected from at least 80 searchers.

Acknowledgments

We wish to express our sincere thanks to Elaine Toms (Dalhousie University, Canada), Luanne Freund (University of Toronto, Canada) and Heather L. O'Brien (Dalhousie University, Canada) for their hard and imaginative work on the search tasks, to the team at Max Planck Institute (Germany) for access to the TopX XML element retrieval engine, Martin Theobald, Ralf Schenkel and Gerhard Weikum, as well as the team at CSIRO (Australia) for access to the Panoptic™/Funnelback™ passage retrieval engine: Alexander Krumpholz, Peter Bailey, George Ferizis, Ross Wilkinson and Dave Hawking. We would also like to thank Gheorghe Muresan from Rutgers University, USA for the inventive idea of placing Pehcevski and Thom's relevance categories into the user-friendly T-shape.

References

1. Denoyer, L., Gallinari, P.: The Wikipedia XML corpus. SIGIR Forum 40(1), 64–69 (2006)
2. Fuhr, N., Klas, C.P., Schaefer, A., Mutschke, P.: 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), pp. 597–612 (2002)

3. Klas, C.P., Albrechtsen, H., Fuhr, N., Hansen, P., Kapidakis, S., Kovács, L., Kriewel, S., Micsik, A., Papatheodorou, C., Tsakonas, G., Jacob, J.: A logging scheme for comparative digital library evaluation. In: Proceedingss of the 10th European Conference on Research and Advanced Technology for Digital Libraries (ECDL), pp. 267–278 (2006)
4. Larsen, B., Tombros, A., Malik, S.: 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, pp. 39–42 (2005)
5. Pehcevski, J.: Relevance in XML Retrieval: The User Perspective. In: Trotman, A. Geva, S. (eds.) Proceedings of the SIGIR, Workshop on XML Element Retrieval Methodology: Held in Seattle, Washington, USA, 10 August 2006. Dunedin (New Zealand): Department of Computer Science, University of Otago, pp. 35–42 visited 15-3-2007 (2006), http://www.cs.otago.ac.nz/sigirmw/
6. Theobald, M., Schenkel, R., Weikum, G.: An efficient and versatile query engine for TopX search. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB), pp. 625–636 (2005)
7. Toms, E.G., Freund, L., Kopak, R., Bartlett, J.C.: The effect of task domain on search. In: Proceedings of the 2003 Conference of the Centre for Advanced Studies on Collaborative Research, pp. 303–312 (2003)
8. Toms, E.G., Freund, L., Jordan, C., Mackenzie, T., Toze, S., O'Brien, H.: Evaluating Tasks by Type and Form. In: INEX2006 Workshop Proceedings (This volume) (2007)

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 clarification, and to find a good example of a castle and fortress in Canada and Denmark.	History-Travel	Fact Finding	Parallel
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

Appendix B – Participating Groups

Research Group	Task
City University London, England	Task A
Dalhousie University, Canada	Task B
Kyungpook National University, Korea	Task A
Oslo University College, Norway	Task A
Queen Mary University of London, England	Task A
Robert Gordon University, Scotland	Task A
Royal School of LIS, Denmark	Task A, Task B
Rutgers University, USA	Task A, Task B
University of Duisburg-Essen, Germany	Task A
University of Otago, New Zealand / Microsoft Research Cambridge, England / RMIT University, Australia	Task A

XML-IR Users and Use Cases

Andrew Trotman[1], Nils Pharo[2], and Miro Lehtonen[3]

[1] Department of Computer Science, University of Otago, Dunedin, New Zealand
andrew@cs.otago.ac.nz
[2] Faculty of Journalism, Library and Information Science, Oslo University College, Norway
nils.pharo@jbi.hio.no
[3] Department 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.

1 Introduction

Many of the teething troubles in setting up realistic XML-IR experiments have been blamed on the lack of user grounding. Trotman [19] 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 Use Case Studies Track [22]. 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 [7] identified the use of XML-IR in book search and separately Lehtonen [13] 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 [20] and emphasizing the importance of Passage Retrieval. We do not

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 400–412, 2007.

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.

2 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 [12] prior to the first INEX, and the task for which it could be easiest for the established search engines to adopt on the web.

2.1 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 [20]. 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.

2.2 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 [18]. 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 [16] – 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[1]. 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 cannot be certain of the accuracy of the content. The context is important, and is needed.

2.3 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.

2.4 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 [12]. There they asked assessors to identify the best entry points from which a reader should start reading in order to satisfy their information need.

2.5 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?

[1] The counter argument is that focused retrieval is about finding elements and not about displaying elements, and as such there is no context from which to be free.

3 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 [25]). 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 [23], and the latter are simple keyword lists.

Several other methods of specifying queries were tested at INEX (see Trotman [24] for details), but it is the loose semantics and the simplicity that make NEXI an appealing language for XML-IR.

3.1 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.

3.2 Other Models

Van Zwol examined a graphical user interface for choosing structural constraints for CAS queries [25]. In his interface the user selects structures from lists and is able to specify keywords to be constrained to those structures. These graphical queries are easily converted to CAS queries in the NEXI language. Baeza-Yates [1] used block-like graphical language to represent the nesting of structures. Query by fragment, the specification of XML fragments containing keywords was examined by Carmel *et al.* [3].

3.2 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.

3.4 Result Granularity

Two models for specifying result granularity exist within INEX, either it is specified explicitly (or vaguely in a CAS query) or it is left to the search engine to determine (in a CO query or a CAS query targeting //*). Previous justification of the vague interpretation has been based on the user not knowing the DTD, or alternatively not having an expectation of the best size of a result to answer their questions.

The best size of a result may be dictated not only by the user but also by equipment used to render the information. Some web sites provide different interfaces depending on the surfing device being used. Yahoo, for example, provides an interface for mobile phones as well as desktop computers. The search engine could take advantage of the user context as part of the search process. In the case of XML-IR the search engine could interrogate the user's display characteristics and specifically target elements that are of a granularity to fit the device. In the case of a desktop computer this might be a section of a book whereas on a palmtop it might be several consecutive paragraphs, but on a mobile phone only a single paragraph.

4 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.

4.1 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. [16]. 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.

4.2 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*. [8] 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 [15] are used by Newsblaster to summarize a collection of documents 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.

4.3 Question Answering (Focused)

Question Answering has been examined at TREC, CLEF and NTCIR, but not yet at INEX. In this case the search engine must identify a very short (certainly less than a paragraph) snippet from a document that answers a user question – a focused retrieval

strategy. Although it is unusual to see sentence and entity level markup in a document, applications that insert them do exist (for example, POS taggers).

4.4 Book Search (Relevant in Context or Fetch & Browse)

Dopichaj [7] 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 [21] 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 [14] 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 [14] 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.

4.5 Extending Web Search (Best in Context)

Lehtonen [13] 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 [9] 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.

5 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.

5.1 Keyword Only

Examples of keyword only interfaces abound the Internet (for example Google) and before this they abounded CD-ROMs.

5.2 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 [17] 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.

5.3 Keywords Limited to Structures with Target Elements Specified

Although now defunct, BioMedNet once used an SGML [11] 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).

5.4 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 [13] 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.

6 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 [6] 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 [6] 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.

6.1 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. These are chosen to best fit the rendering capacity of a handheld computer.

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.

6.2 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.

6.3 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 [10] (but previously under different names).

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.

6.4 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.

7 Conclusions

There are currently four *ad hoc* tasks at INEX: thorough, focused, relevant in context, and best in context. For each of these tasks 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 an application is 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 these four tasks are presented.

Accepting that it is reasonable to rank these 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.: 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 (1998)
[2] BioMedCentral, Biomedcentral. (November 2006) Available: http://biomedcentral.com
[3] Carmel, D., Maarek, Y.S., Mandelbrod, M., Mass, Y., Soffer, A.: Searching XML documents via XML fragments. In: Proceedings of the 26th ACM SIGIR Conference on Information Retrieval, pp. 151–8 (2003)
[4] Clark, J., DeRose, S.: XML path language (XPath) 1.0, W3C recommendation. The World Wide Web Consortium. Available (1999), http://www.w3.org/TR/ xpath
[5] Clarke, C., Kamps, J., Lalmas, M.: INEX 2006 retrieval task and result submission specification. In: Proceedings of the INEX 2006 Workshop to appear (2006)
[6] Cockburn, A.: Writing effective use cases, 1st edn. Addison-Wesley, London, UK (2000)
[7] Dopichaj, P.: Element retrieval in digital libraries: Reality check. In: Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology pp. 1–4 (2006)
[8] Evans, D., Klavans, J.L., McKeown, K.R.: 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) (2004)
[9] Google. Google. (November 21, 2006) Available: http://google.com
[10] Guinness: Guinness book of records. 22nd edn. Guinness Superlatives Ltd, London (1975)
[11] ISO8879: 1986. Information processing - text and office systems - standard generalised markup language (SGML) (1986)
[12] Kazai, G., Ashoori, E.: What does shakespeare have to do with INEX. In: Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology, pp. 20–27 (2006)
[13] Lehtonen, M.: Designing user studies for XML retrieval. In: Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology, pp. 28–34 (2006)
[14] Mass, Y., Mandelbrod, M.: Component ranking and automatic query refinement for XML retrieval. In: Proceedings of the INEX 2004 Workshop, pp. 73–84 (2004)
[15] McKeown, K.R., Barzilay, R., Evans, D., Hatzivassiloglou, V., Kan, M.-Y., Schiffman, B., Teufel, S.: Columbia multi-document summarization: Approach and evaluation. In: Proceedings of the Document Understanding Workshop (DUC 2001) (2001)
[16] O'Keefe, R.A.: If INEX is the answer, what is the question. In: Proceedings of the INEX 2004 Workshop, pp. 54–59 (2004)

[17] PubMed, Pubmed. (November 21, 2006) Available: http://ncbi.nlm.nih.gov
[18] Tombros, A., Larsen, B., Malik, S.: The interactive track at INEX 2004. In: Proceedings of the INEX 2004 Workshop, pp. 410–423 (2004)
[19] Trotman, A.: Wanted: Element retrieval users. In: Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, Second Edition, pp. 63–69 (2005)
[20] Trotman, A., Geva, S.: Passage retrieval and other XML-retrieval tasks. In: Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology, pp. 43–50 (2006)
[21] Trotman, A., O'Keefe, R.A.: Identifying and ranking relevant document elements. In: Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX) (2003)
[22] Trotman, A., Pharo, N.: User case studies track. INEX. (November 13, 2006) Available: http://inex.is.informatik.uni-duisburg.de/2006/usercase.html
[23] Trotman, A., Sigurbjörnsson, B.: Narrowed Extended XPath I (NEXI). In: Proceedings of the INEX 2004 Workshop, pp. 16–40 (2004)
[24] Trotman, A., Sigurbjörnsson, B.: NEXI, now and next. In: Proceedings of the INEX 2004 Workshop, pp. 41–53 (2004)
[25] van Zwol, R., Baas, J., van Oostendorp, H., Wiering, F.: Query formulation for XML retrieval with bricks. In: Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, Second Edition, pp. 80–88 (2005)

A Taxonomy for XML Retrieval Use Cases

Miro Lehtonen[1], Nils Pharo[2], and Andrew Trotman[3]

[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 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

Research on XML IR methodology often lacks a clearly defined scope. Some researchers claim that their methods are general and applicable to arbitrary XML documents while others are accused of fine-tuning their methods to a single document collection. Despite the often supportive evidence presented with the results, these claims have no common theoretical grounding. Thus, we cannot describe how general each method is or whether a method is fine-tuned to a single collection, document type, document genre, or something else. The original contribution of this paper is the first proposal for a common taxonomy for the use cases of XML Retrieval. The taxonomy should help us see which aspects of XML retrieval are addressed and which use cases are potentionally involved in each research effort. Knowing those, it is quite straightforward to describe how methods generalise and how they are fine-tuned.

There are several approaches to creating a hierarchical classification for the use cases, depending on which features or facets we consider important. Therefore, we need to analyse the properties and characteristics of various use cases in order to find the right level of description. 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

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 413–422, 2007.

presentation-specific names towards metadata-like content descriptors as we move marked-up text towards the other end of the continuum.

Content type. Books and other monographies, chapters, articles, web pages, messages, document fragments and other incomplete documents.

Size. A continuum from very small to very big. There is often interdepency between the content type and document size. The size usually makes a difference when presented to the user. Some documents 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 or value–based inventory of the document collection.

Presentation of search results. Links to standalone documents optionally with relevant passages, best entry points, aggregated or assembled documents, heatmaps.

A similar effort was put forth by Andrei Broder who presented a taxonomy of web search [1]. He classified web queries into three classes according to their intent: navigational, informational, or transactional. While this classification of web searches is closely related to the user's task, the use cases of XML retrieval are tightly coupled with what kind of searching is possible, given certain XML documents. The contrast between the two classifications has a clear explanation: Anything can be searched on the web, whereas XML retrieval only concerns searching the contents of XML documents. For example, finding the URI of an XML document is not considered XML retrieval.

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 [2]. 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, `<b>` 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 usually 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. So far, we have identified three subcategories in Class A: Book search (A.1), Article search (A.2), and Fragment search (A.3).

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. Although book search has not been explored by INEX researchers in the past, there might be a new track for book search at INEX 2007.

Example document types:

- Docbook[1].

Presentation of search results: Links (with summaries) to relevant sections and subsections or best entry points.

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.

[1] http://www.docbook.org/

A.2 Article Search

The strength of XML shows in the ways the queries can be refined. For example, the set of matching articles can be limited to those that were published during a given time period, or those that were written by certain authors. Conditions can be set on any metadata included in the documents. Article search was on the agenda of the INEX initiatives 2002–2005.

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

Having to search document fragments instead of whole documents is often the natural consequence of adopting an XML-based publishing environment. For example, the Franklin content management system of IBM decomposes information into reusable XML fragments [3]. Corresponding systems have become mainstream for enterprises. What is common to these environments is that the source documents are typically too small or too dependent on other documents (incomplete) to be published on their own. Whether the search application assembles the retrieved fragments into coherent answers is case-dependent.

Considering the number of operational systems, fragment search is one of the most common use cases of XML Retrieval. It is thus unfortunate that no INEX track has yet been devoted to it. Future prospects do not look any brighter as the realistic and often proprietary document collections are beyond reach.

Example documents:

- Single-sourced documentation [4].

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

Adding metadata to documentation and making the documents self-contained were some of the major incentives for developing the XML standard. Those goals are fulfilled in content-oriented XML documents.

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, key values, 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* [5] 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

Semantic search is much like article search (A.2) to the user. The biggest difference between semantic search and article search is that the XML documents for the former contain semantic markup which the search application is aware of.

Example documents:

- Documentation with descriptive element names (metadata) and full-text content,
- The Guideline Elements Model (GEM) [6].

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, learning and reading about the topic.

System tasks: Identifying relevant elements, vague matching of content and structural search conditions, organising the matches into whole documents.

B.2 Entity Search

Full-text content where certain entities are marked up is a typical environment for 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

Although the process-oriented XML documents are searched more often than any other types of XML documents, they are not paid much attention in the IR-related literature. Operational search applications have been available online for several years, though. The systems typically index and retrieve only XML documents, however, they have been sadly neglected by the XML Retrieval community. 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).

The applications that process the XML of Class C are highly aware of the XML markup of the documents. Good search applications are likely to follow the practice which is discouraging to the development of more general methods.

XML: Most of the tag names describe the technical interpretation of the content. Common tag names include `<sequence>`, `<input>`, `<operation>`, and `<port>`. Search applications typically specialise in a single set of tag names (a single document type), which shows in fine-tuned methods and tailored user interfaces.

Queries: Keywords, keyphrases, and structural constraints with domain-specific interpretation.

Presentation of search results: The relevant answers are listed as links pointing to the actual answers. Some answers such as Web services (C.3) are naturally viewed in a web browser, whereas others may need separate applications. For example, newsfeeds (C.2) are opened with a feed reader.

C.1 Multimedia Search

As XML has also become a format for multimedia, it is natural to query and search such XML documents. However, the XML representation of multimedia is often the source format, whereas systems are more accustomed to querying the derivative formats such as jpg and pdf. Whether search applications will actually specialise in querying the XML of the multimedia documents is thus uncertain.

Example document types:

- SVG[2], SMIL[3].
- Other documents that may contain multimedia.

User tasks: Finding multimedia.

System tasks: Ranking multimedia by metadata descriptions, possibly interpret the XML structures, as well.

[2] http://www.w3.org/Graphics/SVG/
[3] http://www.w3.org/AudioVideo/

C.2 Feed Search

Information filtering has long been and IR application [7], but it was not until XML came around that searching newsfeeds (including blogs) online had a real user demand. The first operational implementations of XML feed search date back to 2003 when Feedster launched their online system[4].

Example document types:

– RSS[5], OPML[6].

User tasks: Finding relevant postings.

System tasks: Indexing and querying streaming XML, monitoring and filtering XML feeds.

C.3 Web Services Search

Finding the most relevant web services has become an application area of XML Retrieval, thanks to the XML-based Web Services Description Language (WSDL) [8]. Woogle[7] [9] is a decent implementation of a such a search.

Example document types:

– WSDL.

User tasks: Finding good web services that are compatible with their demand.

System tasks: Matching keywords with both service descriptions and operations, as well as input and output parameters. Compositions of operations with similar functionality to the search operations may also be returned.

C.4 Message Search

When XML is sent in an XML envelope, it becomes an XML message. Systems that send and receive such messages also store them for archival purposes. Searching the archived XML messages is a potential application area of XML Retrieval.

Example document types:

– SOAP [10].
– Other transactional documents.

[4] http://www.feedster.com/
[5] http://web.resource.org/rss/1.0/
[6] http://www.opml.org/spec
[7] http://www.cs.washington.edu/woogle

User tasks: Finding relevant messages.

System tasks: Matching strings and values with the relevant parts of the message, indexing messages.

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. Although the descriptions are still incomplete, we can learn a few key points from them:

- Both the user tasks and system tasks are different for each use case, which implies that no system can have a single search interface for all the use cases.
- Only few use cases are included in the INEX-related research, possible because of some diversity in the interpretation of "XML Retrieval".

The two-level classification described in this paper is the result of at least one round-table meeting, some email correspondence, literature review, a few interviews with two graduate students who independently develop systems for XML retrieval, and a few interviews with commercial representatives of the XML server industry. This work is on-going and the authors appreciate any feedback and contributions that advance this work.

References

1. Broder, A.: A taxonomy of web search. SIGIR Forum 36, 3–10 (2002)
2. Chidlovskii, B., Fuselier, J.: Supervised learning for the legacy document conversion. In: DocEng '04: Proceedings of the 2004 ACM symposium on Document engineering, pp. 220–228. ACM Press, New York, USA (2004)
3. Weitzman, L., Dean, S.E., Meliksetian, D., Gupta, K., Zhou, N., Wu, J.: Transforming the content management process at ibm.com. In: CHI '02: Case studies of the CHI2002/AIGA Experience Design FORUM, pp. 1–15. ACM Press, New York, USA (2002)
4. Clark, D.: Rhetoric of present single-sourcing methodologies. In: SIGDOC '02: Proceedings of the 20th annual international conference on Computer documentation, pp. 20–25. ACM Press, New York, USA (2002)
5. 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, pp. 445–452. ACM Press, New York, USA (2006)
6. ASTM International: E2210-02 Standard Specification for Guideline Elements Model (GEM)-Document Model for Clinical Practice Guidelines (2003)
7. Belkin, N.J., Croft, W.B.: Information filtering and information retrieval: two sides of the same coin? Commun. ACM 35, 29–38 (1992)

8. W3C: Web Services Description Language (WSDL) Version 2.0, W3C Candidate Recommendation (March 27, 2006) Latest version available at http://www.w3.org/TR/wsdl20
9. Dong, X., Halevy, A.Y., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: VLDB '04: Proceedings of the 30th International Conference on Very Large Data Bases, pp. 372–383 (2004)
10. W3C: SOAP Version 1.2 Part 0: Primer (Second Edition), W3C Proposed Edited Recommendation (December 19, 2006) Lates version available at http://www.w3.org/TR/soap12-part0/

What XML-IR Users May Want

Alan Woodley[1], Shlomo Geva[1], and Sylvia L. Edwards[2]

[1] School of Software Engineering and Data Communication
[2]School 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 retrieval on a granularity lower than documents XML-IR systems will better satisfy users' information need than traditional IR systems. Participants 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 users. The purpose of the User-Case Studies track is to explore the application of XML-IR systems from the users' perspective. This paper outlines QUT's involvement in this track. 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 system for, what extra information they would need to interact with an XML-IR system 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.

1 Introduction

XML-IR systems differ from traditional IR systems by returning results to the user at the sub-document (that is element) level. The assumption is that XML-IR systems will be able to better fulfil users' information needs since they only return the relevant parts of documents to users, rather than whole documents that will undoubtedly contain both relevant and irrelevant material. Most of the INEX tracks and tasks have been developed based upon this assumption, each which a slightly different user model in mind. Participating INEX systems are evaluated in the Cranefield tradition involving sets of: source documents, end-user queries (topics), relevance judgements and metrics. Despite the progress made by INEX participants, debate has raised as to how applicable some of the tracks and task really are to potential end-users of XML-IR systems [3]. The aim of the User-Case Studies track is to examine this question and to investigate situations where XML-IR is suitable for end-users.

This paper details QUT's participation in the User-Case Studies track. Our participation stems from previous work in the Ad-hoc and NLP tracks. In previous years we

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 423–431, 2007.

have developed a laboratory XML-IR system [1] and natural language interface [4] for participation in both of those tracks. This year, for the first time we were able to test our systems in a user experiment. Following the experiment we interviewed some of the participants asking them two sets of questions. The first set of questions focussed on their experiences using the natural language interface and an alternative template-by-query interface to formulate structured queries. The second set of the questions were more general, asking them how they felt about XML-IR overall and if there were situations where XML-IR would be more beneficial than traditional IR. The answers to the second set of questions form the basis of this paper.

The rest of this paper is organised as follows. It begins with a description of the user experiment. Then it discusses the interviews with the participants following the experiment. Finally, it outlines how, based upon the information gathered from the interviews, ways that INEX can facilitated user-centred XML-IR tasks.

2 The Experiment

The experiment simulated the task of users interacting with an academic retrieval system. Sixteen participants took part in the experiment. The participants acted as academic researchers, for example: post-graduate research students, corporate researchers or academics. The participants searched the INEX IEEE collection, a set of academic IEEE journal articles from 1995 to 2002. The journals had a broad range of focus, ranging from general journals such as Computing to specific journals such as Neutral Networks.

The participants were post-graduate information technology students who were uninitiated in the domain of XML-IR. While this may not a representative sample of possible XML-IR users, it was necessary to have such participants since understanding the technical nature of the information needs and source collection is beyond casual users. Also since the participants were uninitiated in the domain of XML-IR, it is valid for us for us to extrapolate the results of this experiment into the wider area of XML-IR. The participants were given six information needs that simulated those of a real user. The information needs contained both a detailed explanation of the information sought and a condition of relevance that described the motivation behind the information need. The information needs were sampled from the narrative elements of INEX Topics 253 – 284.

The system used in the experiment was separated into two parts: the front-end interfaces and the backend retrieval system. Two different interfaces were used: NLPX, that accepted queries written in natural language (English) [4], and Bricks, a query by template interface that allowed users to enter queries via a graphical user interface [5]. Examples of the input screen used for both interfaces appear in Figures 1 and 2. These examples capture the type of queries entered by the participants. The same backend search engine, GPX, was used for both interfaces. For each result retrieved by GPX, users were presented with the option of selecting to view the entire document or just the element. Since GPX only accepted formal language queries, both interfaces translated their user input into NEXI before submitting them to GPX. Below we describe NLPX, Bricks and GPX in more detail.

2.1 Interface A – NLPX

NLPX accepts natural language queries (NLQs) and produces formal queries written in the NEXI language. The NLPX translation process involves four steps. First, NLPX tags words either as special connotations (for instance structures) or by their part of speech. Second, NLPX divides sentences into atomic, non overlapping segments (called chunks) and then classifies them into grammatical classes. Third, NLPX matches the tagged NLQs to query templates that were derived from the inspection of previous INEX queries. Finally, NLPX outputs the query in NEXI format. Batch testing of a single backend search engine that used both natural language queries parsed through NLPX and formal NEXI queries has shown comparable results [4]. This is the first time that NLPX has been tested in a usability experiment.

2.2 Interface B – Bricks

Bricks is a query-by-template interface that allows users to input structured queries via a graphical user interface (GUI). Users enter their content needs via text boxes and their structural needs via drop-down boxes. To aid users, structural needs are indicated via conceptual rather than physical names, for example “a section” rather than sec. Bricks allows users to develop queries in several steps (”blocks”) starting with their desired unit of retrieval and then by adding any additional information needs. Blocks are also added as the user traverses the hierarchy of the documents. Upon completion of input, the data in the Bricks GUI is translated to formal NEXI expression, however, due to the constraints of the GUI, users are unable to enter malformed expressions. Usability testing has shown that users find Bricks superior to keyword only and NEXI interfaces [5].

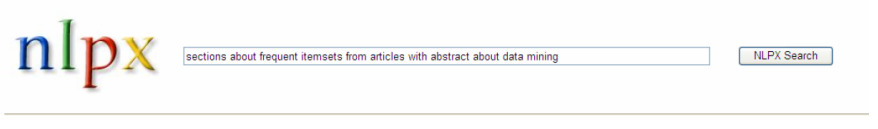


Fig. 1. The NLPX search interface

2.3 Backend Retrieval System – GPX

The backend retrieval system for this experiment was Gardens Point X (GPX) [1]. GPX was chosen since it has performed strongly at the annual INEX conference since 2002 - consistently among the top three systems. GPX stores the information about each leaf element in the collection as an inverted list. Upon retrieval, GPX matches query terms to all leaf elements that contain the term and then dynamically creates their ancestors. Elements are ranked according to their predicted relevance in GPX’s ranking scheme. GPX rewards leaf elements that contain phrases and specific, rather than common, terms. It also rewards ancestors with multiple relevant children, rather than a single relevant child. For this experiment, the results list was filtered so that

"overlapping elements" (that is, elements whose ancestors or descendants appear higher ranked on the results list) were removed before been being presented to users. This decision was made because users have been known to react negatively to overlapping elements [2].

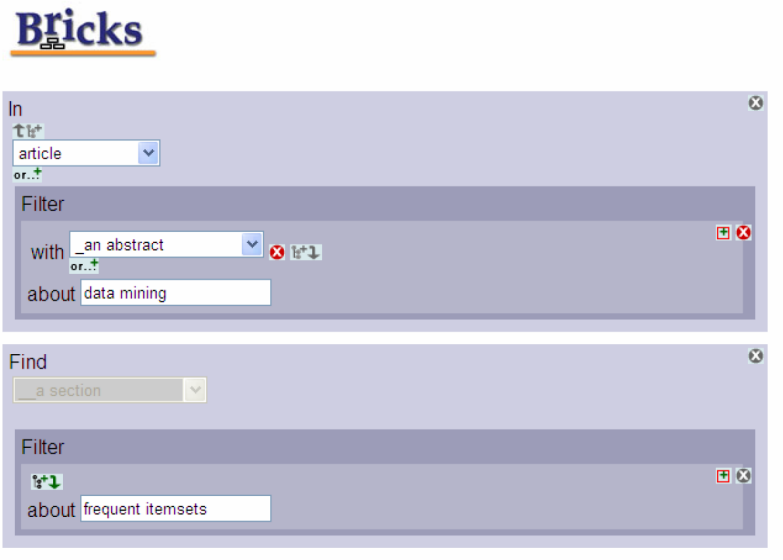


Fig. 2. The Bricks search interface

3 Interviews

After the experiment 12 out of the 16 participants were interviewed. Some of the questions asked were specifically about the experiment, in particular their experience using the query formulation interfaces. A discussion on these questions is outside the scope of this paper. However, another set of questions were about their thoughts on the area of XML information retrieval as a whole in comparison traditional information retrieval. These questions, and some of their responses are presented here.

The main difference between XML-IR systems and traditional IR systems is that XML-IR systems returns elements rather than documents. We assume that since elements are more specific than documents that they will be more useful to users. However, this assertion has only been very limitedly tested with users, mainly in the context of INEX's interactive track. Here we ask our participants in which situation would element-retrieval be more useful than document retrieval.

The first observation was that element retrieval would be more useful than document retrieval in situations where there was a lot of, largely irrelevant, information in the source documents. Or alternatively, situations where the user was searching for very precise or specific information. This is summarised in the responses made by participants 8,1 and 12 shown in Figures 3 – 5.

Participant 8: "Technical forum, and you want to find a solution, and sometimes has hundred of pages, and each page has hundred of discussion, in that one it might help, just look at one of them and might help you to find the one you need."

Fig. 3. Participant 8's response regarding XML-IR uses

Participant 12: "So any thing that is not free from, anything that has these logical sections would be beneficial. Specifically if you're just looking for, to just focus your search results on these specific categories, which obviously you can't do in an unstructured manner."

Fig. 4. Participant 12's response regarding XML-IR uses

Participant 1: "You get much more precise searches by using markup. And all the information is all in the one place in the document, or rather gotten easily from the document, which can't be done with free text. If you take free text with no mark up and then you take text with all the author details, abstracts, bibliography all marked up you're going to be able to find stuff a lot quicker on the marked up one than the free text one. So XML would be a great improvement on free text."

Fig. 5. Participant 1's response regarding XML-IR uses

Participant 11: "So yeah, I could say I wanted to particularly search for these keywords in the abstract of the paper or been able to express that kind of thing. Or saying that I needed a paper that was generally about a certain topic, but then I wanted a section in that paper that was about a particular kind of subtopic so I didn't end up with these other papers that were about the right kind of general topic but not about the specific subtopic that I wanted."

Fig. 6. Participant 11's response regarding complex queries

Others felt that the use of structure enabled them to write very detailed queries, particular those that may be about more than one topic. These types of queries are common in the CAS tasks, where a query may specify to a retrieve a particular about a topic, but also wishes the document to contain information about another topic. This opinion was expressed by participant 11.

An observation made by some participants was that the users of XML-IR system would need to know the structure of the document that they were searching, and possibly be domain experts. This was a point raised by participants 1 and 5.

Participant 1: "I'm guessing that every time you open up a new document, there's different ways of representing the structure, so I think that would make it quite difficult to use on a daily basis. If you were using the same file structure then Bricks would be great, but if you were using different structures or DTD then it would be really difficult to figure out how to use it."

Fig. 7. Participant 1's response regarding document knowledge

Participant 5: Its probably true that you need a bit of experience with the domain or at least in research to know where you have to look for a particular research type document. Similarly if your looking for a publicity or news type article you might want to have some idea how they're structured, and that's obviously a bit of domain knowledge that you need to have, but once you've got it, it makes a lot to sense to use it because if I want the title to have something in it that I'm searching for then its good to be able to query that way.

Fig. 8. Participant 5's response regarding document knowledge

Participant 2: "Why not just group those different components together so that the user can have a choice. I see this document, and this document has 3 or 5 components relevant to the information need. And the other document has 2 parts [related to the] information needs."

Fig. 9. Participant 2's response regarding highlighting

Participant 6: "I mean its [element retrieval is] ok , as long as you can retrieve the actual whole document when you get the section back, if you could retrieve the whole document and see where it fits in that would be fine, and if you could retrieve the whole document and then see the section that you pulled up, that would be fine as well. But as long as you could see both the document and section that would be fine... so the whole document presentation with the section highlighted would work for me."

Fig. 10. Participant 6's response regarding document knowledge

Another point made by some participants was that element retrieval could be used in conjunction with document retrieval. Specifically, when documents are retrieved their most relevant elements could be highlighted. This opinion was expressed by participants 2 and 6 shown in Figures 9 and 10.

The next section discusses how the outcomes of these interviews can be used to developed more user-orientated tasks at INEX.

4 Discussion

The results of the user experiment and interviews are of great value for the INEX community. Particularly pleasing was that the participants found merit in the use of XML-IR systems to fulfil their information need. The challenge for INEX participants and organisers is to use the information derived from these types of experiments to help focus our research efforts. An immediate way that we can put this into practise is by re-examining the tasks we preform each year, particularly in the Ad-hoc track, to see how well they correspond to tasks that users want. As a reference the following tasks are currently performed by INEX participants:

1. **Thorough Retrieval**: the aim of the thorough retrieval task is to retrieve all relevant elements matching an information request.
2. **Focussed Retrieval**: the aim of the focussed retrieval task is to retrieve the most relevant elements along an XPath matching an information request.
3. **All in Context**: the aim of the all in context retrieval task is to first, retrieve the most relevant documents matching an information request and second, to highlight all relevant information within those documents.
4. **Best in Context**: the aim of the best in context retrieval task is to first retrieve the most relevant documents matching an information request and second, to find the best entry point for those documents.

In this experiment results were returned as a single ranked list of relevant elements. Overlapping elements were removed, and therefore, the presentation is analogous to the output of systems in focussed retrieval task. Users were, at least initially, confused by the presentation of results as a single list of ranked elements. However, this is not too surprising since the users' experience with retrieval systems has been solely with document retrieval systems, and hence the idea of receiving back elements would have appeared "unnatural" to some. Users seemed to find the retrieval of elements with little or no context particular confusing. It is important to note that overlapping elements were removed from the presentation since previous experiments have shown that users react adversely to them [2]. If they were included in this experiment then the user reaction may have been even more negative. At first, this seems alarming for proponents of the Thorough and Focussed tasks since these tasks are based on returning lists of elements. However, even if the tasks are not suitable for end-users they might still be worthwhile perusing since they could be used a precursor for other XML-IR tasks, such as Best in Context or All in Context, or other information seeking tasks, such as question and answering.

During the post-experiment interviews it was discovered that users reacted positively to the idea of highlighting relevant elements within a document. This is a positive sign for INEX since it correlates well to the All in Context task. The users felt that highlighting passages would be beneficial to deciding if the document they are browsing is relevant. It would also help them when browsing large documents, particularly for documents that contain a lot of irrelevant information. This presents an interesting opportunity for INEX since it opens the possibility of having a document retrieval task at further workshops. This would allow participants to examine if the if techniques specifically designed for XML-IR are able to find more relevant documents (not elements) or even documents that are more relevant than traditional IR techniques. This task could be run in conjunction with one of the other document evaluation forums such as TREC or CLEF.

Expanding on this issue users also commented that they liked the idea of a best "entry point" into documents. Again, this is pleasing for INEX since it directly correlates to the Best in Context task. Some users also commented that they would like to see the elements within the documents ranked according to relevance. This presents the opportunity to extend the All in Context task to measure the retrieved elements within each document as a ranked rather than unordered list.

We have already discussed how XML-IR could help users when they wish determine if their document is relevant. However, there are other information seeking tasks where XML-IR could be useful. One such task is when a single user query has multiple information requests. Often, in this scenario the user wish to retrieve a particular item for instance *sections about information retrieval* inside of articles that will have a second information item such as *paragraphs about compression* even if they don't wish to retrieve items matching the second request. This type of "complex" information request would be encapsulated in the NEXI expression **//article[about(//p,compression)]//sec[about(.,information retrieval)]** and is typical of one of the more complex CAS queries. This is a validation that this type of query is suitable for users, particularly when accessing documents that are about multiple topics, and that INEX should continue to use these types of queries in the future

A final comment made by interviews was the XML-IR system enabled them to find more specific results than traditional IR systems. INEX could capitalise on this situation in several ways. First, it strengthens the motivation for INEX's named entity task, since information need for that task is very specific, and inherently requires some sort of sub-document retrieval. Another interesting area of research that the INEX community would be to examine how users' information needs change as they interact with the retrieval system. One could assume that their information needs would start vague and then become more specific as they interact with the system. And as their needs become more specific one could assume that an XML-IR system would become more useful than a traditional IR system. Some of the INEX tracks, particularly the interactive track, could examine if this is true.

5 Conclusion

This paper outlined QUT's involvement in this years User-Case Studies track. Our participation stems from our work in two of the other INEX tracks, namely, the Ad-hoc and NLP tracks. This paper detailed an experimentation we performed, and

user interviews following the experiment. It then discussed, using information derived from the interviews, ways in which the INEX community can focuses on user-centred research.

Acknowledgements

The authors would like to acknowledge Utrecht University especially Jeroen Bass and Roelof van Zwol for the development of and permission to use Bricks, as well as the anonymous participants of the experiments, in particular those participants who were interviewed following the experiment. Without their contribution this work could not been conducted.

References

1. Geva, S.: GPX - Gardens Point XML Information Retrieval INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 221–222. Springer, Heidelberg (2005)
2. Pehcevski, J., Thom, J.A., Tahaghoghi, S.M.M., Vercoustre, A-M.: Hybrid XML Retrieval Revisited, In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 153–167. Springer, Heidelberg (2006)
3. Trotman, A.: Wanted: Element Retrieval Users. In: Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology, Glasgow, Scotland, pp. 58–64 (July 30, 2005)
4. Woodley, A., Geva, S.: NLPX at INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 358–372. Springer, Heidelberg (2005)
5. van Zowl, R., Bass, J., van Oostendorp, H., Wiering, F.: Query Formulation for XML Retrieval with Bricks. In: Fuhr, N., Lamas, M., Trotman, A. (eds.) In: Proceedings of INEX 2005 workshop on Element Retrieval Methodology, Glasgow, Scotland, pp. 75–83 (July 30, 2005)

Report on the XML Mining Track at INEX 2005 and INEX 2006

Categorization and Clustering of XML Documents

Ludovic Denoyer[1], Patrick Gallinari[1], and Anne-Marie Vercoustre[2]

[1] LIP6 - University of Paris 6
firstname.lastname@lip6.fr
[2] INRIA Rocquencourt
anne-marie.vercoustre@inria.fr

Abstract. This article is a report concerning the two years of the XML Mining track at INEX (2005 and 2006). We focus here on the classification and clustering of XML documents. We detail these two tasks and the corpus used for this challenge and then present a summary of the different methods proposed by the participants. We last compare the results obtained during the two years of the track.

1 Introduction

The XML Document Mining track[1] was launched for exploring two main ideas: first identifying key problems for mining semi-structured documents and new challenges of this emerging field and second studying and assessing the potential of machine learning techniques for dealing with generic Machine Learning (ML) tasks in the structured domain i.e. classification and clustering of semi structured documents.

This track has run for two editions: 2005 and 2006, and the present report summarizes the work done during these two years [2]. The track has been supported through INEX by the DELOS Network of excellence on Digital Libraries and by the PASCAL Network of excellence on Machine Learning.

Among the many open problems for handling structured data, we have focused in this track on two generic ML tasks: *classification* and *clustering*. The goal of the track was therefore to explore algorithmic, theoretical and practical issues regarding the classification and clustering of XML Documents. Note that one new task - Structure mapping[3] - has been proposed in the 2006 edition of the track but since only two submissions was made for this more complex task ([1],[2]), we will only discuss here the results obtained for classification and clustering. In the following, we first describe the mining problems addressed at INEX in

[1] http://xmlmining.lip6.fr
[2] the challenge will continue one year more in 2007.
[3] Structure Mapping was defined as learning from examples how to map documents in one or several formats onto a predefined mediated schema.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 432–443, 2007.

section 2, we then describe in section 3 the instances of the mining tasks at INEX 2005 and 2006. Finally in section 4.1 we summarize the different contributions of participants.

2 Categorization, Clustering of XML Documents

Dealing with XML document collections is a particularly challenging task for ML and IR. XML documents are defined by their logical structure and their content (hence the name semi-structured data). Both types of information should be addressed in order to effectively mine XML documents. Compared to other domains, where the structured data consists only of the "structure" part with no content this is much more complex and this had been addressed only for very specific problems in the ML community. Note that most existing ML methods can only deal with only one type of information (either structure or content). Structure document information is described through a labelled tree where labels do correspond to XML tags which may or may not carry semantic information. In the used document collections, content information is composed of text and images, but only the textual part was considered in the mining track of INEX. The textual content is usually contextually dependent of the logical structure (e.g. section content vs header or bibliography information). It should be stressed that XML documents usually come in large collections and that scalability is a fundamental issue for mining semi-structured data.

When dealing with semi-structured documents, according to the application context and on the prior information available on the collection, it may be relevant to consider the structure information alone or to consider both structure and content information. Two types of tasks were then defined corresponding to these two conditions: "Structure Only" (SO) and "Structure and Content" (SC).

Dealing with structure alone measures the ability to recover or classify structural classes corresponding to different document sources. The structure + content tasks are more challenging and encompass many different generic tasks in the document domain. In order to define more precisely the classification and clustering problems, let us consider the two following dimensions: structure (S) and thematic content (T). The former dimension characterizes the document structure generation and the latter its content class. Figure 1 illustrates a partition of a document collection composed of different thematic areas and structures. The goal of classification and clustering will be to identify different classes or groups of documents corresponding for example to different information sources (each source or "class" corresponds to a color on Figure 1). Classification will aim at discriminating between the different sources, clustering will try to recover some hidden source information. In this example, each source may be considered as a mixture of one or more themes and structures in different proportions. The different sources may overlap in different proportions leading to tasks with different complexity.

For a classification problem for example, the models will have to classify specific mixtures of content and structure information components corresponding to each source.

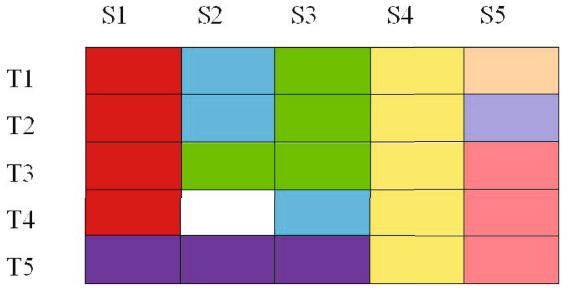


Fig. 1. The structural and thematic dimensions do respectively correspond to columns and rows in the figure. All documents in column S1 for example have structure S1 - or follow schema S1 - while all documents in row T1 have thematic T1. The different information sources are identified by colors. In this example, there are 9 distinct sources, each identified by a given color, among a maximum of 25 potential sources. Each source is defined here as a mixture of several structure and themes, for example, on the left bottom, documents share thematic content T5 and may have structure S1, S2 or S3.

Depending on the application in mind, one may distinguish different generic problems like:

- identify common structures across different content types or themes (structure oriented classification and clustering) - this is illustrated in Figure 2 where each class corresponds to a specific structure and deals with all the collection themes. The theme may appear in different proportions for each class
- identify different content types across structures (content oriented classification and clustering). Classes would then correspond to the horizontal lines in the figure.
- identify homogeneous classes for both structure and content information (mixed clustering and classification) - this corresponds to Figure 1 and is the more general task. It may come in many different variants.

From this enumeration, it should be clear that many different classification and clustering problems may occur in real situations and structural and content information will play different roles for these two generic tasks.

3 Tasks and Evaluation

We will now describe the different corpora and tasks proposed during the two years of the XML Document Mining track. During the first year of the track

Fig. 2. Each source corresponds to a specific structure (5 sources here - corresponding to columns) - the different thematic (rows) will appear in different proportions in each source

both the *structure only (SO)* classification/clustering tasks and *structure and content (SC)* tasks where considered, while the second year was more focused on the *structure and content* tasks.

3.1 Corpora

The different tasks proposed during the XML Mining Track were evaluated on three different XML corpora whose characteristics are provided in Table 1. The movie corpus is an artificial corpus based on the IMDB[4] Corpus while IEEE and Wikipedia are real world document corpora. In some cases the classes or clusters correspond to a natural partition of the corpus, while in other cases, several classes were generated artificially from an homogeneous corpus as detailed below. Three corpora have been used for the track:

1. The movie corpus is composed of about 9,500 XML documents that describe movies.
2. The IEEE corpus is composed of 12,000 scientific articles from IEEE journals in XML format. This has been the reference corpus for INEX from year 2002 to 2005 [3].
3. The XML Wikipedia corpus is composed of 150,000 english documents from Wikipedia, formatted in XML. This corpus is a subpart of the official corpus for INEX 2006 where each document belongs to exactly one category. A complete description of this corpus is available in [4] under the name of *classification corpus*.

3.2 Structure Only Task

Structure Only tasks correspond to classification or clustering using the structural description of XML documents alone, i.e. the XML ordered labelled tree

[4] http://www.imdb.org

Table 1. Description of the different corpora used. Corpora were splitted using 50% of the documents for training and 50% for testing.

Corpus	Nb Docs	Nb node labels	USed for SO	Used for SC
Movie	9,463	≈ 190	YES	NO
IEEE	12,108	≈ 150	YES	YES
Wikipedia	≈ 150,000	≈ 10,000	YES	YES

providing the relations between the document elements. The input space in this case corresponds to the tag alphabet, and to the relations between these tags. Note that the tag space is of limited size here, much less than the size of the dictionary for CS tasks, but can be quite high (up to 10 000 different tag names for Wikipedia).

Dealing with structure alone measures the ability to identify information sources in the context of XML data - e.g different Web servers, XML databases, ... Note that in many other domains (e.g. biology) data also come as ordered labelled trees so that investigating classification and clustering methods for such trees is a generic problem, with many applications outside the field of XML documents.

For the *structure only* task, we have proposed a set of five different classification/clustering subtasks. Four are based on the movie corpus, one on the IEEE corpus. For each subtask, the goal is to recover by classification or by clustering the different classes, i.e. the different structural sources in the corpus.

- **Movie N SO task**: This task is based on the *m-db-s-N*[5] corpora that are composed of XML documents describing movies expressed in eleven different schemas. These schemas were defined by hand and the documents from the corpus were mapped onto each schema using XSLT scripts. Number *N* quantifies the difficulty of the task: the higher N is, the more overlap there is among the 11 classes. We have defined 4 tasks identified by N = 0 to 3.
- **IEEE SO task**: This task consists in identifying two different families of structures coming from a same source. The *INEX SO* corpus used here corpus corresponds to the reference INEX IEEE corpus [3] where content information has been removed. Documents in this corpus come from two structural sources *Transactions on ...* journals and other IEEE journals. The two sources are composed of documents following the same DTD but with different writing rules. The goal here is to recover this class information.

3.3 Structure and Content Tasks

In these tasks, the goal is to identify categories defined corresponding to structural and thematic characteristics. Classes or clusters are then to be identified using both the structure and the content of the XML documents. We have used two different corpora here for defining two different tasks:

[5] http://xmlmining.lip6.fr

– **IEEE SC**: This task amounts at identifying 18 categories of the IEEE corpus that correspond to the 18 different IEEE journals present in the collection. These categories are both structural and thematic. As said before, there are two broad structural sources in this corpus (Transaction journals and other journals), while the same thematic can be covered in two different journals.
– **Wikipedia SC**: This task proposes to identify 60 categories in a corpus of about 150,000 XML documents that comes from the wikipedia XML corpus. In this corpus each document belongs to exactly one category and each category corresponds to a *portal* of wikipedia - see [4] for more informations. This collection allows us to evaluate the capacity of the different models to deal with large scale XML corpora. Note that the categories information that appear in the wiki document have been removed from the XML files.

3.4 Evaluation Measures

Different measures have been used during the first year of the track and the second year. In all experiments, each document belongs to exactly one category or one cluster (see part 3.1).

Precision, Recall and F1 measure for classification. In the field of classification, for a given category, *recall* is the ratio of the number of correctly classified documents in the category to the total number of documents from this category. *precision* is the ratio of the number of correctly classified retrieved in the category to the total number of documents assigned to this category. F_1 measure reflects both the precision and the recall (see Table 2).

$$Precision = \frac{C}{C+F}, Recall = \frac{C}{C+M}, F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (1)$$

Table 2. Precision, Recall and F_1. C is the number of correctly classifed documents, F is the number of falsely classified documents, M is the number of documents that are not correctly classified

Note that the recall, precision and F_1 presented in the result part are the mean of the recall, precision and F_1 over all the categories.

Purity for clustering. For clustering, we can also define a *precision* and a *recall* (also called *purity*) for each of the clusters. These measures are computed as expressed in the previous section considering that a cluster is assigned to the category that corresponds to the majority of its documents. Note that these measures are only informative but do not really allow us to compare the quality of two methods. Moreover, these measures completely depend on the number of clusters found by the different models. The only way to really know if a clustering has a good quality is to look at the details of the clusters found. Note that the track allowed participants to freely choose the number of clusters - but the real number of clusters was known for each corpus.

4 Models and Results

4.1 Models Submitted

We present here the nine different methods used for clustering and classification of XML documents using *structure only* or *structure and content* information. Figure 3 shows a typology of the different works submitted to the track and the models used. We present here a small summary of the different methods.

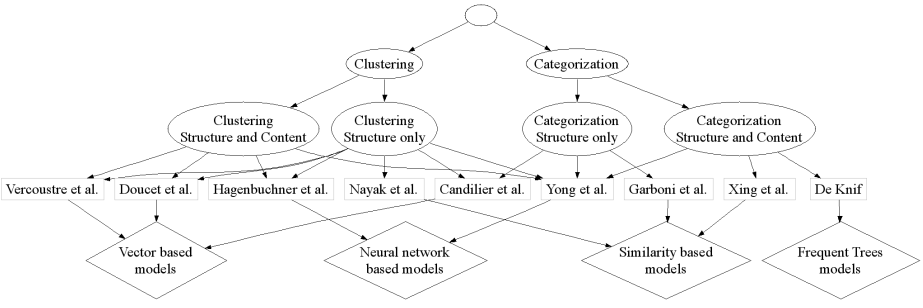


Fig. 3. The typology of the different methods proposed during the XML Mining track. This figures shows the different articles submitted to the track and the tasks concerned by the different models. We also present here the different ideas underlying each article. **Vector based models** are models that first transform XML documents to a vector (or a set of attributes-values) and then use classical vectorial models for clustering or classification. **Similarity based models** define a similarity measure over XML documents - or XML clusters - and then use this measure for clustering or classification. **Neural network based models** are models based on Self Organizing Map or Recursive Neural Networks. **Frequent Trees models** concern models that use the extension of frequent item sets to XML trees.

Vercoustre et al. - Clustering - 2005. The method presented [5] models each XML tree by a set of elements that basically correspond to the different sub-paths of the XML tree. Each of these elements is characterized using different criterions: the length of the path considered, the root node label, the number of nodes in the path, etc. The models then projects each document into a vectorial space where each feature of the space corresponds to one of the possible element i.e each document is transformed into the frequential vector of its elements. The clustering problem is then considered as a classical vectorial clustering problem. This model is used for *structure only* clustering but can be used also with *structure and content* if the author model the sub-path with some content information. The model is evaluated on the **Movie SO** and **IEEE SO** clustering tasks.

Garboni et al. - Classification - 2005. This model [6] was developped for the classification of *structure only* XML documents. It proposes a similarity measure between an XML document and an XML cluster. Each XML document is transformed into a sequence of its node labels. Then, each cluster is transformed into a set of sequences that are extracted from the set of all the sequences of the cluster using a classical sequential pattern extraction measure. The similarity measure between an XML document and a cluster of documents is thus computed as the longest common subsequence between the sequence of the document and the sequences that characterize the cluster. The model has been tested on the **Movie SO** classification task.

Candilier et al. - Classification and Clustering - 2005. [7] proposed to transform each XML tree into a set of attributes-values. The attributes-values sets are built using different relations between the nodes of the input tree (parent-child relations, next-sibling relations, set of distinct paths,...). Classification and clustering of these attributes-values sets are then made using the *Subspace clustering algorithm (SSC)* and the *C5* classification algorithm. The experiments are made on both the **Movie SO** and **IEEE SO** corpora.

Hagenbuchner et al. - Clustering - 2005 and 2006. The two papers [8] and [9] propose a method for the clustering of *Structure only* and *Structure and Content* XML documents. The method is based on an extension of *Self Organizing Map (SOM)* to *SOM-SD (SOM for Structured Data)* and *Contextual SOM-SD* that can take into account complex structures like labelled trees. This method was used on the **Movie SO**, **IEEE SO** and **IEEE SC** corpus.

Doucet et al. - Clustering - 2006. The article by Doucet et al. [10] introduces a method that transforms an XML document to a vector and then uses a K-means algorithm for the clustering. The transformation that projects a document to a vector takes into account both the structure and the content. The paper also proposes to integrate a *textitude* measure to the document description process that basically measures the ratio between the weight of the structural information and the weight of the content information. This method is used on the **IEEE SC**, **IEEE SO**, **Wikipedia SC** and **Wikipedia SO** corpora.

De Knijf - Categorization - 2006. The model proposed in [11] makes classification of XML document using Frequent attributes Trees. The algorithm is composed of 4 steps:

1. Each class is characterized by a set of Frequent Attributes Trees discovered using the *FAT-miner* algorithm
2. Emerging trees are selected for each category
3. Each document is then transformed into a vector where each component indicates if a particular emerging tree appear into the document
4. Last, a classical classification algorithm is used on these vectors (Binary decision tree)

This model is used on the **Wikipedia SC** corpus.

Nayak et al. - Clustering - 2005 and 2006. The papers by Nayak[6] et al. ([12] and [13]) defines a similarity measure between an XML document and a cluster of XML documents. This similarity called CPSim (for Common Path Similarity) is computed during a matching step and is based on different criterions that take into account:

- the number of common nodes between the document and the documents of the cluster considered,
- the number of common nodes paths,
- the order of the nodes of the XML document,
- ...

This measure is then used by an incremental clustering algorithm called PCXSS. This model is applied on the **IEEE SO** and **Wikipedia SO** corpora.

Xing et al. - Categorization - 2006. In this paper ([14]), the auhors proposes to use both a tree edit distance and a Minimum Description Length criterion (MDL) for the classification of *content and structure* XML documents. The method models each class with a normalized regular hedge grammar (NRHG). This grammar is extracted from a set of document using the MDL principle. The distance between a document and a category is then computed by a tree edit distance between the XML tree and the grammar computed for the category. The model is tested on the **IEEE SC** corpus.

Yong et al. - Categorization and Clustering - 2006. This article [15] proposes to categorize XML documents using Graph Neural Networks (GNN). A GNN is an extension of the formalism of Recurrent Neural Network designed for graphs and trees. The intuitive idea behind GNN is that nodes in a graph represent objects or concepts and edges represent their relationships. A *state* vector is attached to each node which collects a representation of the object represented by the node, where the state is naturally specified using the information contained in the neighborhood of a node. The model is used for the **IEEE SO** and **IEEE SC** classification tasks.

4.2 Results

We present below the results obtained by the participants for the classification in Tables 3 and 4, and clustering in Tables 5 and 6. The different results are only indicative of the task difficulty and/ or of the potential of a method for a given task. The track was not designed as a benchmark for comparing finely different approaches and methods, but rather as a forum for investigating classification and clustering on structured documents. Participants were free to use any preprocessing for the data and to participate to any task. More results are provided in the full papers of each participant.

[6] the two papers of 2005 and 2006 do not use exactly the same algorithm but are based on the same idea

Table 3. Classification results using a recall measure during INEX 2005

Method	Movie S (0 to 3)	IEEE S	Wikipedia S	IEEE SC	Wikipedia SC
Candilier et al.	96.8 % , 96.6 % , 94.7 % , 94.2 %	94.1 %	-	-	-
Garboni et al.	95 %, 94 %, 89 %, 80%	-	-	-	-

Table 4. Classification results using a F1 measure during INEX 2006

Method	Movie S (0 to 3)	IEEE S	Wikipedia S	IEEE SC	Wikipedia SC
Yong et al.	- , - , - , -	48%	-	72 %	-
Xing et al.	- , - , - , -	-	-	60%	-
De Knijf	- , - , - , -	-	47%	-	-

Table 5. Clustering results using a purity measure during INEX 2005

Method	Movie S (0 to 3)	IEEE S	Wikipedia S	IEEE SC	Wikipedia SC
Vercoustre et al.	45% , 71 % , 66 % , 53 %	70%	-	-	-
Candilier et al.	78%,-,-,-	-	-	-	-
Hagenbuchner et al. 2005	97%, -,-,-	-	-	-	-
Nayak et al. 2005	60%,60%,59%,59%	65%	-	-	-

Table 6. Clustering results using a purity measure during INEX 2005

Method	Movie S (0 to 3)	IEEE S	Wikipedia S	IEEE SC	Wikipedia SC
Hagenbuchner et al. 2006	-, -,-,-	36 %	13%	-	-
Nayak et al. 2006	-,-,-,-	18%	12.5%	-	-
Doucet et al.	-,-,-,-,	13%	22%	34 %	42 %

These results show some tendencies. Classification on the Movie S (N) datasets, each composed of 11 classes appears quite easy for all the values of parameter N. As said before, in order to create these corpora, artificial classes were generated from a Movie description dataset. The classes have different structures with different degrees of overlapping. Candilier et al. [7] also obtained excellent results for the classification of IEEE document structures into two classes. The content and structure classification task is more complex since the number of classes here is more important (18). For clustering, here too, some participants were able to recover the hidden classes in the Movie S (N) corpora sometimes with a high accuracy. The structure of the IEEE collections (transactions and non transactions) was also recovered up to 70 % by Vercoustre et al. [5]. For all the other tasks, performance was rather poor. Real sources like IEEE or Wikipedia collections are more challenging to mine than artificially built collections like Movie S. Note that in the literature, a majority of the experiments for evaluating structured data clustering methods have been performed on artificial data created

via random tree generators. It is clear from the above experiments that they are not representative of real situations. The SO and SC tasks were investigated as separate tasks, and the respective influence of structure and content cannot be inferred from these results for the SC tasks. This should be investigated in the future.

5 Conclusion

We have presented here the different models and results obtained during the two years of XML Document Mining Track at INEX for both the classification and clustering tasks. The performances obtained show that the structure only task seems quite easy and simple models work very well on this task. This is why we have focused during the second year to the *structure and content* tasks. Concerning the SC tasks, the results obtained can certainly be improved with more sophisticated models. The structure and content tasks on both the IEEE and the Wikipedia corpus will continue next year during INEX 2007.

For INEX 2007, we will also propose the XML Mining track and we will focus on classification/clustering of content+structure XML documents on both the IEEE corpus and the Wikipedia Corpus. The experience of these two years of XML Mining track showed us that we have to define a more strict context and evaluation measures in order to really compare the different methods, and try to encourage participants to submit categorization results that are easier to analyze. For the next year, we will first preprocess all the data - it will allow participant to concentrate on the models - and propose a set of results obtained by classical flat categorization/clustering models in order to have some baseline models as comparison.

Acknowledgments

We are grateful to Remi Gilleron (INRIA, University of Lille), Marc Tommasi (INRIA, University of Lille), Marie Christine Rousset (LIG, Grenoble) and Nathalie Pernelle (LRI, Orsay) for their help on the definition of the different tasks and the construction of the different corpora. We would like to thank all the participants for their efforts and hard work.

References

1. Maes, F., Denoyer, L., Gallinari, P.: XML structure mapping application to the pascal INEX 2006 XML document mining track. In: Workshop of the INitiative for the Evaluation of XML Retrieval (2006)
2. Gilleron, R., Jousse, F., Tellier, I., Tommasi, M.: XML document transformation with conditional random fields. In: INEX 2006 (2007)

3. Fuhr, N., Gövert, N., Kazai, G., Lalmas, M., (eds.): Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX), Schloss Dagstuhl, Germany, December 9-11, 2002. In: Fuhr, N., Gövert, N., Kazai, G., Lalmas, M., (eds.) Workshop of the INitiative for the Evaluation of XML Retrieval (2002)
4. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. SIGIR Forum (2006)
5. Vercoustre, A.M., Fegas, M., Gul, S., Lechevallier, Y.: A flexible structured-based representation for XML document mining. In: Workshop of the INitiative for the Evaluation of XML Retrieval, pp. 443–457 (2005)
6. Garboni, C., Masseglia, F., Trousse, B.: Sequential pattern mining for structure-based XML document classification. In: Workshop of the INitiative for the Evaluation of XML Retrieval, pp. 458–468 (2005)
7. Candillier, L., Tellier, I., Torre, F.: Transforming XML trees for efficient classification and clustering. In: Workshop of the INitiative for the Evaluation of XML Retrieval, pp. 469–480 (2005)
8. Hagenbuchner, M., Sperduti, A., Tsoi, A.C., Trentini, F., Scarselli, F., Gori, M.: Clustering XML documents using self-organizing maps for structures. In: Workshop of the INitiative for the Evaluation of XML Retrieval, pp. 481–496 (2005)
9. Kc, M., Hagenbuchner, M., Tsoi, A., Scarselli, F., Gori, M., Sperduti, A.: XML document mining using contextual self-organizing maps for structures. In: Workshop of the INitiative for the Evaluation of XML Retrieval (2006)
10. Doucet, A., Lehtonen, M.: Unsupervised classification of text-centric XML document collections. In: Workshop of the INitiative for the Evaluation of XML Retrieval (2006)
11. Knijf, J.D.: Fat-cat: Frequent attributes tree based classification. In: Workshop of the INitiative for the Evaluation of XML Retrieval (2006)
12. Tran, T., Nayak, R., Raymond, K.: Clustering XML documents by structural similarity with pcxss. In: Workshop of the INitiative for the Evaluation of XML Retrieval (2006)
13. Nayak, R., Xu, S.: XML documents clustering by structures. In: Workshop of the INitiative for the Evaluation of XML Retrieval, pp. 432–442 (2005)
14. Xing, G., Xia, Z.: Classifying XML documents based on structure/content similarity. In: Workshop of the INitiative for the Evaluation of XML Retrieval (2006)
15. Yong, S.L., Hagenbuchner, M., Tsoi, A., Scarselli, F., Gori, M.: XML document mining using graph neural network. In: Workshop of the INitiative for the Evaluation of XML Retrieval (2006)

Classifying XML Documents Based on Structure/Content Similarity

Guangming Xing[1], Jinhua Guo[2], and Zhonghang Xia[1]

[1] Department of Computer Science, Western Kentucky University, Bowling Green, KY 42104
guangming.xing@wku.edu, zhonghang.xia@wku.edu
[2] Computer and Information Science Department, University of Michigan - Dearborn, Dearborn, MI 48128
jinhua@umich.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 precision based on the user's specification. Thirdly, classification of XML documents is discussed. Representation of XML documents based on the structures and contents is also studied. 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 [12,13].

Various methods [11,4] 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 [10,6] 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 [11] proposed a method

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 444–457, 2007.

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 offer 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].

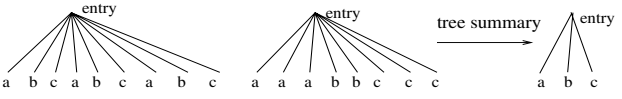


Fig. 1. Trees of Different Structure with the Same Structural Summary

As the structure of an XML document is defined by a schema, it is ideal 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 an 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 discusses the representation of the content information in an XML document, and the combination with the structure information. Section 5 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 6, and the conclusion remarks are given in section 7.

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 an 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 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 (a sequence of trees which corresponds to a 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 an NRHG if it can be generated by the grammar. When a tree doesn't conform to an 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 an NRHG* is the minimum cost of a sequence of edit operations transforming the tree to conform to the grammar.

2.1 The Edit Distance Between an XML Document and a Schema

In this section, we present the recursion to calculate the distance between an ordered labeled tree and an NRHG using top down edit distance [2]. It should be noted that other distances between XML documents and schemata may be used in place of this algorithm.

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 an 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 an 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 $T[i]$:*

$$C[v_t, T[i]] = \min \left\{ \begin{array}{lll} v_t \rightarrow x & \delta(x, T[i]) & (1) \\ v_t \rightarrow a\langle v_f \rangle & \delta(\lambda, T[i]) + C[v_t, \lambda] & (2) \\ v_t \rightarrow a\langle v_f \rangle & C[v_f, F[i]] + \delta(a, t[i]) & (3) \end{array} \right\}$$

and for each $v_f \in V_F$ and sub-forest $F_s[i] = T[i]F_s[n(i)]$:

$$C[v_f, F_s[i]] = \min \left\{ \begin{array}{lll} v_f \rightarrow v_t & C[v_t, T[i]] + \delta(\lambda, F_s[n(i)]) & (4) \\ v_f \rightarrow v_t & \delta(\lambda, T[i]) + C[v_f, F_s[n(i)]] & (5) \\ v_f \rightarrow v_t v_f^{'} & C[v_t, T[i]] + C[v_f^{'}, F_s[n(i)]] & (6) \\ v_f \rightarrow v_t v_f^{'} & \delta(\lambda, T[i]) + C[v_f, F_s[n(i)]] & (7) \\ v_f \rightarrow v_t v_f^{'} & C[v_t, \lambda] + C[v_f^{'}, F_s[i]] & (8) \\ v_f \rightarrow v_f^{'} & C[v_f^{'}, F_s[i]] & (9) \end{array} \right\}$$

The correctness of the above theorem can be found in [2].

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). This precludes direct use of dynamic programming. We may use the same modification given in [9] to circumvent this problem.

Firstly, the other three cases that lead to smaller cases of the problem can be computed by the following formula:

$$known[v_f, F_s[i]] = \min \left\{ \begin{array}{lll} v_f \rightarrow v_t & C[v_t, T[i]] + \delta(\lambda, F_s[n(i)]) & (4) \\ & \delta(\lambda, T[i]) + C[v_f, F_s[n(i)]] & (5) \\ v_f \rightarrow v_t v_f^{'} & C[v_t, T[i]] + C[v_f^{'}, F_s[n(i)]] & (6) \end{array} \right\}$$

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: $C[n..F[1.. \mid T \mid]]$ 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. **for**[1] $s = 0$ **to** n **do**
13. **for**[2] all tree $T[i]$ of size s **do**
14. **for**[3] all $v_t \in V_T$ **do**
15. calculate $C[v_t, T[i]]$
16. **for**[2] all forest $F_s[i]$ of size s **do**
17. **for**[3] all $v_f \in V_F$ **do**
18. $C[v_f, F_s[i]] = known[v_f, F_s[i]]$
19. H $\leftarrow$ heap of $C[v_f, F_s[i]]$
20. **while not** H.empty()
21. C[v_f,$F_s[i]$] $\leftarrow$ H.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.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 documents. 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 6.

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 $\{d_1, d_2, ..., d_n\}$, find a schema s, such that d_1, d_2, ..., 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 [15,14]. 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.

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 precision and generalness 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 $\{abcab, abdab, abcabab, abcababab, abdababababab\}$ 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.

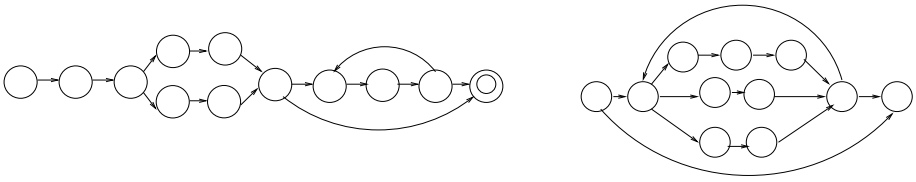


Fig. 2. $NFAs$ for $ab(c|d)(ab)*$ and $(ab|c|d)*$

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.

Table 1. Transition Vectors for NFA_1 and NFA_2

Step	1 2 3 4 5 6 7 8 9 10 11 12	cost for NFA_1	1 2 3 4 5 6 7 8 9 10 11	cost for NFA_2
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

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 in 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 $d_i = (w_1, w_2, \ldots, w_{|T|})$, where $0 \leq w_j \leq 1$ is the weight of the jth 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

$$A_{|T|\times|D|} = U_{|T|\times r} \Lambda V^T_{r\times|D|}$$

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.

$$\hat{q} = \Lambda^{-1} U^T_{r\times|T|} q_{|T|}.$$

After the transformation, the similarity between query q and document d_i can be evaluated in the LSI space.

After the structural and content distances have been computed, the distance vectors can be concatenated for representing an XML document. The relative significance between the structure and content can be adjusted by changing the relative weight of the distance vectors.

5 Document Classification

In this section, we will first show how to classify XML documents based on their structures using the edit distance between XML documents and schemas, and then combine the structure and the content information for classifications.

The design of the classification system can be illustrated by Figure 3. When content information is used in classification, the system can take some minor changes as explained in step 2 in the following paragraph.

According to our approach, there are three steps to get a classifier.

– The first step is representative selection and schema extraction. Schema extraction has been covered in detail in Section 3. The representatives of a group are chosen to minimize the intraclass distance.

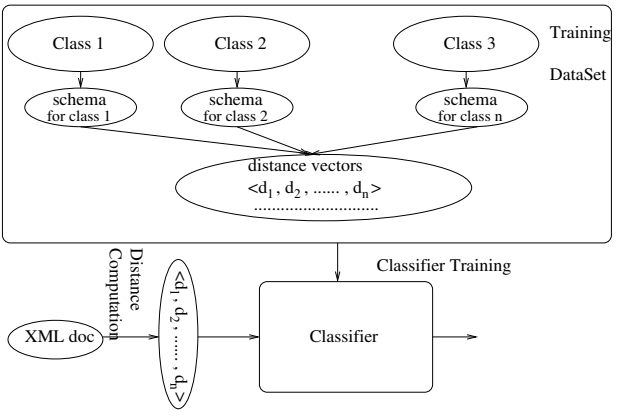


Fig. 3. System Architecture for Document Classification Based on Structure (Content)

- 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 $\langle d_1, d_2, ..., d_n \rangle$, 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. When the content information is needed, a separate vector is computed using the method presented in Section 4. The vector for the structure and content can be concatenated and used for representing an XML document.
- 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. An SVM classifier is trained using the Weka [16] software package in our studies.

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.

6 Implementation and Experimental Results

We have fully implemented the algorithms described in the above sections, and developed a prototype system for document classification. 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 on various datasets to show:

1. The distance between an XML document and a schema is a effective similarity measure for XML documents.
2. The feasibility of XML document classification based on this new distance metric.

The following three datasets are used in our experiments:

- The first dataset is from Sigmod collection: a collection of documents that conform to either one of OrdinaryIssuePage.dtd, IndexTermsPage.dtd or ProceedingsPage.dtd.
- The second dataset is from data generated by using three DTDs bookstore1.dtd, bookstore2.dtd and bookstore3.dtd from [4] which are very close to each other.
- The third dataset is the MovieDB dataset from XML Mining Challenge [12,13], which is a collection of documents that have very similar structure and some noise information.

To evaluate the time efficiency, we compared our method with the classification method using edit distance between trees, and the classification method using edit distance between structural summaries. The time for structural summary method includes time the for computing the tree summaries, and the time needed for edit distance between tree summaries. The time for our method includes the time for tree size reduction, schema extraction, and the time needed for computing the edit distance between the tree and the schema. Fig. 4 shows the time performance on a pair of documents of variable sizes from MovieDB dataset.

From Fig. 4, we know that the original tree distance method is the slowest one, and our method is slower than the structural summary method, but the difference is not significant compared with the original method.

The classification quality is evaluated by following the same procedure as described in [4]. The number of true positive, false positive, false negative for each group, and overall *Precision* P are used to compare different methods.

For a collection of documents that are clustered into n groups, $C_1, ...C_i, ...C_n$ with corresponding DTDs $D_1, ...D_i, ...D_n$, let:

- a_i be true positive for cluster C_i, which is the number of documents in C_i that are indeed a member in D_i;
- b_i be false positive for cluster C_i, which is the number of documents in C_i that are not a member in D_i;
- c_i be false negative for cluster C_i, which is the number of documents not in C_i although they should be a member in D_i.

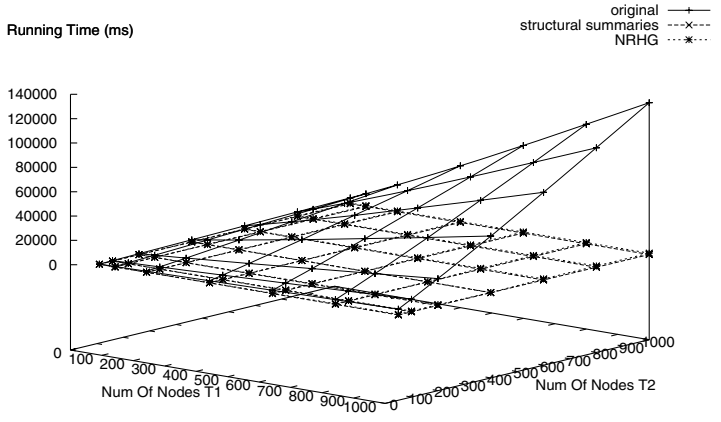


Fig. 4. Time performance for original, structural summaries, NRHG

We have:

$$P := \frac{\Sigma_i a_i}{\Sigma_i a_i + \Sigma_i b_i}.$$

The classification result for the Sigmod, Bookstore, and MovieDB data set based on the structure is presented in Tables 2, 3, and 4.

Table 2. Clustering Results: Sigmod Data

Cluster No	w/o summaries			w summaries			tree grammar		
	a	b	c	a	b	c	a	b	c
1	19	1	1	20	0	0	20	0	0
2	19	1	1	20	0	0	20	0	0
3	20	0	0	20	0	0	20	0	0
P	96.7%			100%			100%		

Notice that in our algorithm, the values of P reach an 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.

We have also evaluated the feasibility of classifying Web documents using the combination of the structure and content information. The inex data set was used in our experiments for this task. The classification result for the inex data set based on the structure and the content is presented in Table 5.

Table 3. Clustering Results: Bookstore Data

Cluster No	w/o summaries			w summaries			tree grammar		
	a	b	c	a	b	c	a	b	c
1	12	15	8	20	8	0	20	2	0
2	5	8	15	12	0	8	18	0	2
3	20	0	0	20	0	0	20	0	0
P	61.7%			86.7%			96.7%		

Table 4. Classification Result: MovieDB Data Based on Structure

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	41.6%			71.6%			95%		

Table 5. Classification Result: Inex Data Based on Structure and Content

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	

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).

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 [11] 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.

We have identified the following challenges that we like to continue to investigate in the future:

1. The selection of the documents for schema extraction: The schema to represent a group of documents depends on not only the schema extraction method, but also the documents selected for schema extraction. How to select documents that can capture the structural properties of each class is a challenging task.
2. The handling of classes with limited documents: When the number of documents of a class is very small, the extracted schema tends to be less accurate, which may contribute to the false negatives in the classification. To make the problem even worse the extracted schema may have smaller distances with documents in other classes, and introduces false positives.
3. Large number of classes: Our system performs very well (for two class classification, it is nearly 100%), however, the performance of the system significantly degrades when the number of classes becomes large (for example 60). The efficiency of the system also becomes significantly degraded when the number of classes becomes large as the feature vector for the structure of a document is linearly increasing.
4. Structure and content combination: How to balance the representation of the textual content and structure information of the document if both types of information are used for classification purpose. The presentation of XML documents in this paper is based on the tree structure and the textual content, however, the combination is a simple concatenation which may omit important semantic information. An improved combination may improve the precision of classification results.
5. Edit distance: The distance between an XML document and a schema used in this paper is very restrictive, which may not be very precise for some applications. We plan to study the distance with less restrictive edit operations (for example the constrained edit operations as presented in [7]) while computing remains efficient.

References

1. Suzuki, N.: Finding an Optimum Edit Script between an XML Document and a DTD. In: Proceedings of ACM Symposium on Applied Computing, Santa Fe, NM pp. 647 - 653 (March 2005)

2. Xing, G.: Fast Approximate Matching Between XML Documents and Schemata. In: Zhou, X., Li, J., Shen, H.T., Kitsuregawa, M., Zhang, Y. (eds.) APWeb 2006. LNCS, vol. 3841, pp. 425–436. Springer, Heidelberg (2006)
3. Canfield, R., Xing, G.: Approximate XML Document Matching (Poster). In: Proceedings of ACM Symposium on Applied Computing, Santa Fe, NM (March 2005)
4. Dalamagas, T., Cheng, T., Winkel, K.-J., Sellis, T.K.: A methodology for clustering XML documents by structure. Information Systems 31(3), 187–228 (2006)
5. Thompson, K.: Regular Expression Search Algorithm. Communications of ACM 11(6), 419–422 (1968)
6. Shasha, D., Zhang, K.: Approximate Tree Pattern Matching. In: Apostolico, A., Galil, Z. (eds.) Chapter 14 Pattern Matching Algorithms, Oxford University Press, Oxford (1997)
7. Zhang, K.: Algorithms for the constrained editing distance between ordered labeled trees and related problems. Pattern Recognition 28(3), 463–474 (1995)
8. Murata, M.: Hedge Automata: A Formal Model for XML Schemata, http://www.xml.gr.jp/relax/hedge_nice.html
9. Myers, G.: Approximately Matching Context Free Languages. Information Processing Letters 54(2), 85–92 (1995)
10. Chen, W.: New Algorithm for Ordered Tree-to-Tree Correction Problem. J. of Algorithm 40, 135–158 (2001)
11. Nierman, A., Jagadish, H.V.: Evaluating structural similarity in XML documents, WebDB 2002, Madison, Wisconsin (June 2002)
12. XML Document Mining Challenge http://xmlmining.lip6.fr/
13. Denoyer, L., Gallinari, P.: Report on the XML Mining Track at INEX 2005 and INEX 2006. In: Proceedings of INEX (2006)
14. Chidlovskii, B.: Schema Extraction from XML Data: A Grammatical Inference Approach, KRDB'01 Workshop, Rome, Italy, (September 15, 2001)
15. Garofalakis, M.N., Gionis, A., Rastogi, R., Seshadri, S., Shim, K.: Xtract: A System for Extracting Document Type Descriptors from XML Documents, SIGMOD Conference 2000, Dallas, Texas, USA pp. 165-176 (May 16-18, 2000)
16. WEKA Project, http://www.cs.waikato.ac.nz/ml/weka/
17. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
18. Karypis, G.: CLUTO A clustering toolkit Technical Report 02017, University of Minnesota, Department of Computer Science, Minneapolis, MN 55455 (August 2002)

Document Mining Using Graph Neural Network

S.L. Yong[1], M. Hagenbuchner[1], A.C. Tsoi[2], F. Scarselli[3], and M. Gori[3]

[1] University of Wollongong, Wollongong, Australia
{sly56,markus}@uow.edu.au
[2] Monash University, Melbourne, Australia
act@hkbu.edu.hk
[3] University of Siena, Siena, Italy
{franco,marco}@dii.unisi.it

Abstract. The *Graph Neural Network* is a relatively new machine learning method capable of encoding data as well as relationships between data elements. This paper applies the Graph Neural Network for the first time to a given learning task at an international competition on the classification of semi-structured documents. Within this setting, the Graph Neural Network is trained to encode and process a relatively large set of XML formatted documents. It will be shown that the performance using the Graph Neural Network approach significantly outperforms the results submitted by the best competitor.

1 Introduction

Neural networks are popular machine learning methods that can be trained on a set of examples. Multilayer perceptron networks [4] 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: 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, or equivalently the difference between the output of the network and the desired output is minimised. 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 and for which there is no known desired value. The network will be able to provide a desired output value to this input. 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

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 458–472, 2007.

objects depicted in photographs that can differ vastly in size, pose or quality. Many computer methods on the other hand cannot handle this vast variation in size, pose or quality. Most methods involve the hard-coding of information or rules to produce acceptable 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 of an object to fly, then such applications may be unable to identify an apple as it is being thrown into the air. Neural networks on the other hand are a generic method that can learn from possibly noisy samples and produce a desired output. As such, a neural network would be able to recognize an apple as an apple regardless of its surroundings or physical condition.

Traditionally, neural networks are trained on constant-length vectorial inputs. One extension, called Recurrent Neural Networks [6], allows the method to be used in characterizing continuous time 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 [3] 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, before feeding it to e.g. a multilayer perceptron for 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 [9].

A recent development produced the *Graph Neural Network* (GNN) [8] which can process much more general types of graphs. This has been a breakthrough development since it has become possible for the first time to process general types of graphs without first "squashing" the graph structured data into a vectorial form.

Numerous examples of graph structured information can be found 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 a 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 encoded in the topological nature of the Web would be lost in the process.

The intuitive idea underlining GNNs is that nodes in a graph represent objects or concepts and edges represent their relationships. To each node n a vector $\boldsymbol{x}_n \in I\!R^s$, called *state*, can be attached which collects a representation of the object denoted by n. $\boldsymbol{x}_n$ is naturally specified using the information contained in the neighborhood of n. More precisely, let $h_{\boldsymbol{w}}$ be a feed-forward neural network, called *transition network*, that expresses the dependence of a node on its neighborhood and is parametrized by a set of parameters $\boldsymbol{w}$. The states $\boldsymbol{x}_n$ are defined as the solution of the following system of equations:

$$\boldsymbol{x}_n = \sum_{u \in \mathrm{ne}[n]} h_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{x}_u, \boldsymbol{l}_u),\ n \in \boldsymbol{N}, \tag{1}$$

where $\boldsymbol{N}$ is the set of nodes of the graph and ne$[n]$ is the set of neighbors of n. The existence of a solution is ensured by appropriately constraining the network parameters and the solution is computed by Jacobi algorithm for nonlinear systems [8]. For each node n, an output $\boldsymbol{o}_n \in I\!R$ is also defined which depends on the state $\boldsymbol{x}_n$ and label $\boldsymbol{l}_n$. The dependence is described by a parameterized *output network* $g_{\boldsymbol{w}}$

$$\boldsymbol{o}_n = g_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{l}_n),\ n \in \boldsymbol{N}. \tag{2}$$

Thus, Eqs. (1) and (2) define a method to produce an output $\boldsymbol{o}_n$ for each node, i.e. a parameterized function $\varphi_{\boldsymbol{w}}(\boldsymbol{G}, n) = \boldsymbol{o}_n$ which takes in input a graph $\boldsymbol{G}$, one of its nodes n and predict a property of the object represented by n. The corresponding machine learning problem consists of adapting the parameters $\boldsymbol{w}$ such that $\varphi_{\boldsymbol{w}}$ approximates the data in the learning data set $\mathcal{L} = \{(n_i, \boldsymbol{t}_i) | \, 1 \leq i \leq q\}$, where each pair $(n_i, \boldsymbol{t}_i)$ denotes a node n_i and the corresponding desired output $\boldsymbol{t}_i$. In practice, the learning problem is implemented by the minimization of a quadratic error function [8]

$$J_{\boldsymbol{w}} = \sum_{i=1}^{q} (\boldsymbol{t}_i - \varphi_{\boldsymbol{w}}(\boldsymbol{G}, n_i))^2 . \tag{3}$$

This paper applies the GNN to the task of classifying XML formatted documents to investigate its suitability. 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. Machine learning methods are known to generalize well over unseen data and that they are insensitive to noise. This renders such methods particularly useful for many data mining tasks. GNN is the only supervised machine learning method for data which is represented by general types of graphs. For the first time is a supervised machine learning method that is capable of encoding graphs applied to this kind of learning problem.

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. Furthermore, it will be demonstrated that the performances obtained by processing either the document in a structure only mode, or in the structure and content mode that the results are vastly superior to those presented by the competitor at INEX 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

The Initiative for the Evaluation of XML Retrieval (INEX) [2] runs an annual international competition on XML document mining, classification, and categorization. For the most recent round in 2006, a training dataset is provided which 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.

Table 1. XML document classes in the training dataset. The associated numbers are the class IDs.

	Computer	Graphics	Hardware	Artificial Intelligence	Internet	Parallel
Transactional	tc(13),ts(18)	tg(15)		tp(17),tk(16)		td(14)
Non Transactional	an(1),co(3), cs(4),it(8),so(12)	cg(2)	dt(5),mi(9)	ex(6),mu(10)	ic(7)	pd(11)

Information on which XML document belongs to which class (the target labels) is provided for the training dataset. The distribution of the training data is shown in Figure 1. It can be observed 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 unbalanced[1] for the number of documents in each class; and
2. The dataset is unbalanced between "Transaction" and "Non-Transaction" types of articles.

There are two classification tasks specified in this competition:

Structure only: Classification of documents by considering the XML structure only. Any textual content (other than the XML tags) is ignored.

[1] By "unbalanced" we mean that there are uneven distributions of training data.

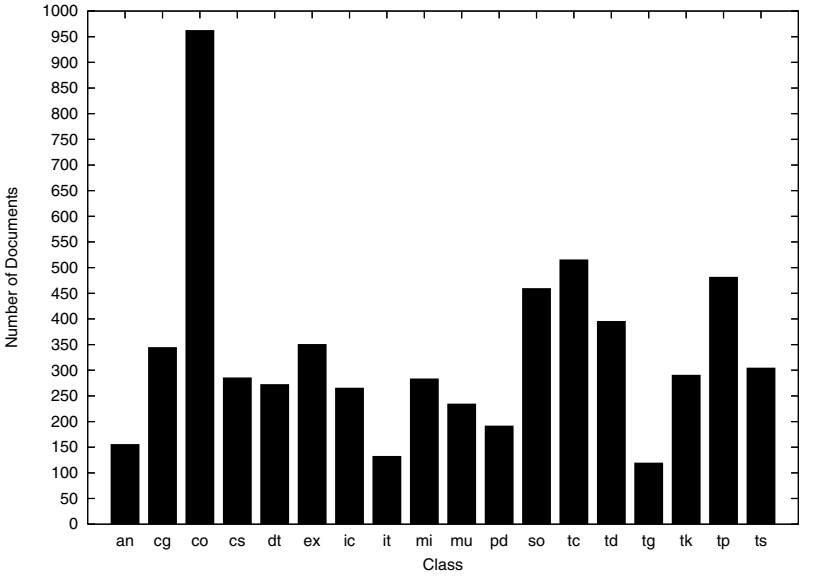


Fig. 1. Distribution of INEX XML documents in their respective class

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 competition[2]. 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 models.

For both tasks, the performance is measured using Macro and Micro-F_1. F-measure[5] is the weighted harmonic mean of precision and recall defined as:

$$F = \begin{cases} 0 & \text{if } \alpha R + (1-\alpha)P = 0 \\ \frac{PR}{\alpha R + (1-\alpha)P} & \text{else} \end{cases} \tag{4}$$

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 average of F_1 weighted according to class dimensions. The task is to maximise F.

[2] The script can be obtained from http://xmlmining.lip6.fr/Results

3 Experiments

3.1 Initial Experiments

Test data were not available until shorty 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[3]:

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 in which the data is not used in the training of the models, and the outputs are assumed to be unknowns, mimicking the real life situation in which no output information is not available on test data sets.

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. The MLP results will form a base upon which the results obtained from GNN can be compared. 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 MLP is used. This is to validate if the inclusion of structural information into the machine learning method produces a noticeable advantage. When using the MLP, the structured input data needs to be "squashed" into a corresponding vector. This is performed as illustrated in Figure 2. Thus, the training dataset 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 $1,000$ iterations required 30 minutes of CPU time on a 3GHz Intel

[3] 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.

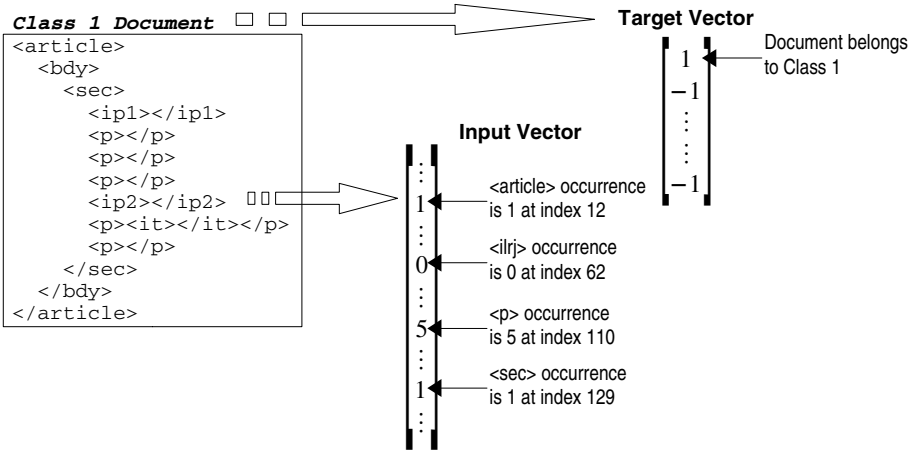


Fig. 2. Conversion of an XML document to a MLP input and target. Each input pattern is a vector $p = [p_1, ..., P_{165}]$, where element e_i equals the number of the occurrences of the i-th XML tag in the document.

based CPU. The performance of the trained MLP on the test dataset is as shown in Table 2. It can be observed that the performance meassured P and R can vary greatly across the various classes and tend to produce better results for the larger classes.

Table 2. MLP training results

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
R	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
P	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																	

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 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 observed 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 preprocessing. However for real world tasks, in general, training GNN without preprocessing is not feasible due to the limitation of computer hardware memory and processing speed.

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
R	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
P	0.05	0.09	0.05	0.09	0.12	0.11	0.02	0.02	0.010	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																	

Without pre-processing, the resulting graphs have a maximum size of 14,047 nodes. The GNN requires each graph to be mapped to a fixed size matrix[4]. 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 internal nodes in the transition network, which is an important contribution to memory usage. 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. Since GNN has to store several data structures for each node, it is not feasible to process such large data sets with current hardware capabilities, 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 as depicted in Figure 3. In Figure 3 it can be observed 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 its neighbour nodes iteratively. Referring to Figure 3, 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 3, 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 3, we note further that there are four <p> nodes in the same configuration. These can be grouped 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 3 is reduced to a graph as shown in Figure 4.

With the pre-processing method, the graph size is reduced to a maximum of 78 nodes per XML document. Each node is then given a unique numeric label according to the XML tag (as shown in Figure 5) to differentiate from other nodes. Ideally these node labels should be multi-dimensional (i.e. 165 dimension in the

[4] This is a limitation with the current software implementation. No such limitation is imposed by the underlying theory of GNN.

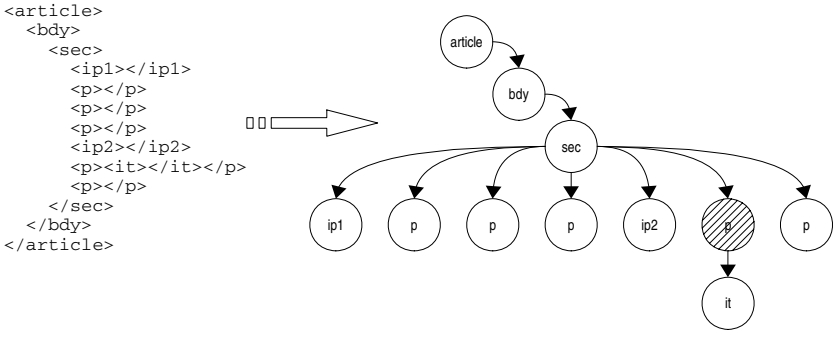


Fig. 3. Conversion of an XML document to a corresponding graph structure

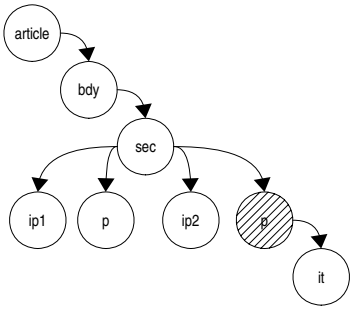


Fig. 4. The reduced XML Graph

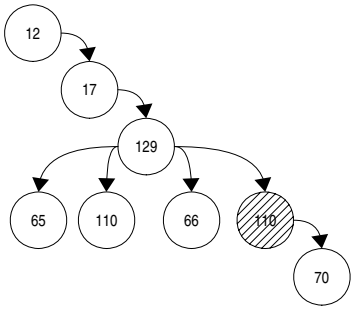


Fig. 5. XML graph with node labels

INEX XML since there are 165 different tags) to maximize the Euclidean distance. For the experiments we use a one hot label representation $l = [l_1, \ldots, l_{165}]$, where $l_i = 1$ if the node contains the i-th XML tag and $l_i = 0$, otherwise. 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[5];

[5] The root node is the root of the XML tree. It is possible for the GNN to compute an output at every node. However, this approach should not be necessary when processing tree structured data.

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.
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.

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

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 interactively, 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 and they may 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.

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
R	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
P	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																	

For the INEX XML Classification using structure only results, it is observed that GNN is capable of performing better than the baseline MLP method. This 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 that of 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 practice, the experiments in GNN typically finished within 12 hours on a 3GHz Intel machine with 2GB RAM. Thus, the GNN requires considerable more time for training when compared to that of the MLP.

During the training of GNN, it was noted that the overfitting of GNN does not appear to occur, and that convergence to a stable mean square error value occured at about $1,000$ to $1,500$ training epochs. Since GNN does not appear to overfit in this learning problem, there appears to be no particular need for a validation dataset. Thus, in Section 3.2, the experiments are performed using all of the data in the original training dataset.

3.2 Further 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 dataset 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 a balanced training dataset obtained using the procedures indicated in the previous section.

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.

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	

GNN, Classification Using Structure and Content: A method is needed to compute a numerical representation of the text contained in a document. As a first approach, we extracted the textual content from each document in the training set, applied the Porter stemming algorithm [7] to obtain a numerical vector representing the text, then trained a naïve Bayes classifier [1] on these data. The Porter stemmer algorithm implements a preprocessing procedure that reduced the words of the documents to their radices. After such a preprocessing, a bag of word representation was used to encode the documents, i.e. a document was a vector $d = [d_1, d_2, \ldots]$, where $d_i = 1$ if the document contains the i-th word of the dictionary and $d_i = 0$, otherwise. The naïve Bayes classifier produced an 18-dimensional binary vector indicating the class membership of the document (which is computed based on the document's textual content only). Thus, to generate the labels for each node in a graph, we applied the following procedure for each document in the dataset:

Step 1: remove the XML tags;
Step 2: remove the stop words;
Step 3: perform Porter stemming algorithm [7] 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 an 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. 5 would now have a node label of <12 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0> (assuming that the naïve Bayes classifier output is class 1) instead of just <12> as was shown in Fig. 6.

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

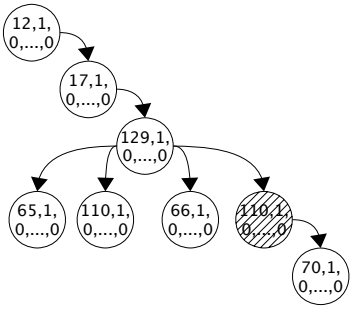


Fig. 6. Graph of XML Document with multi-dimensional node labels

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
R	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
P	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																	

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 observed that the classifier produces a performance which is somewhat better when compared to the results obtained on structure only approaches.

The result of a GNN trained on the data labeled by this classifier is illustrated in Table 8. It can be observed 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 results on the test dataset show that this method achieved Micro F1 of 0.721440 and Macro F1 of 0.713969, which is the best result obtained in the INEX XML classification competition[6]. 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 observed 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

[6] 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 8. Test results for GNN Classification using both structure and content

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
R	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
P	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																	

in the training set by the response of the advanced classifier, and re-trained the GNN accordingly. The results are shown in Table 10. It is observed that the incorporation of structural information has again helped to significantly improve the classification of the GNN.

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
R	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
P	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																	

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
R	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
P	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																	

4 Conclusions

This paper demonstrated for the first time that a supervised machine learning method capable of processing structured data is a suitable approach for classifying possibly 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 alone 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 datasets 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 would 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 provide 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. Crnkovic-Dodig, L., Elkan, P.: Classifier showdown http://blog.peltarion.com/-2006/07/10/classifier-showdown/
2. Denoyer, L., Gallinari, P.: Report on the xml mining track at inex 2005 and inex 2006. In: proceedings of INEX 2006 (2006)
3. Frasconi, P., Francesconi, E., Gori, M., Marinai, S., Sheng, J., Soda, G., A., S.: Logo recognition by recursive neural networks. In: Kasturi, R., Tombre, L.K. (eds.) Second International Workshop on Graphics Recognition, GREC'97, pp. 104–117. Springer, Heidelberg (1997)
4. Haykin, S.: Neural Networks, A Comprehensive Foundation. Macmillan College Publishing Company, Inc. 866 Third Avenue, New York, New York 10022 (1994)
5. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge, Massachusetts (1999)
6. Pineda, F.J.: Generalization of back-propagation to recurrent neural networks. Physical Review Letters 59(19), 2229–2232 (1987)
7. Porter, M.F.: An algorithm for suffix stripping, pp. 313–316. Morgan Kaufmann Publishers, San Francisco (1997)
8. Scarselli, F., Yong, S., Gori, M., Hagenbuchner, M., Tsoi, A., Maggini, M.: Graph neural networks for ranking web pages. In: Web Intelligence Conference, pp. 666–672 (2005)
9. Sperduti, A., Starita, A.: Supervised neural networks for the classification of structures. IEEE Transactions on Neural Networks 8(3), 714–735 (1997)

Evaluating the Performance of XML Document Clustering by Structure Only

Tien Tran and Richi Nayak

Faculty of Information Technology, Queensland University of Technology
Brisbane, Australia
t4.tran@qut.edu.au, r.nayak@qut.edu.au

Abstract. This paper reports the results and experiments performed on the INEX 2006 Document Mining Challenge Corpus with the PCXSS clustering method. The PCXSS method is a progressive clustering method that computes the similarity between a new XML document and existing clusters by considering the structures within documents. We conducted the clustering task on the INEX and Wikipedia data sets.

Keywords: Clustering, XML document mining, Structural mining, INEX, XML, Structural similarity.

1 Introduction

With the emergence of XML standard, XML documents are widely accepted by many industries such as business, education, entertainment and government [2]. 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 a number of applications such as improved information retrieval, document classification analysis, structure summary, improved query processing [1, 8] 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 [7] that is developed to deal with the heterogeneous XML schemas to cluster the INEX 2006 Document Mining Challenge Corpuses [3]. 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 document and existing clusters of XML documents, 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.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 473–484, 2007.

Our philosophy is based on the common usage of XML that is, XML is mainly used for representing the text data in the structured format. Based on this, we assume that a clustering algorithm should group the documents that share a similar structure. For example, documents from the publication domain would have different structure from the documents from the movie domain. Our initial work has shown that the structure of the documents plays a prominent role in grouping the similar XML documents [6]. The semantic difference in tag names can be avoided during the clustering process. In these experiments, we also have not included the instances. The inclusion of instances (the contents within the tag) incurs an additional computing cost. We would like to test the hypothesis such as how important is the structure of the XML documents when categories of documents are mainly based on theme such as the INEX 2006 Document Mining Challenge Corpuses.

The next section gives an overview of the PCXSS methodology. Interested readers can read [7] for a more detailed discussion on this methodology. Phases of the PCXSS method are then described further in Sections 3 and 4. Section 5 reports the results, experiments and data analysis performed on INEX and Wikipedia data sets. The paper is then concluded and further work is outlined in Section 6.

2 The PCXSS Method: Overview

Fig. 1 illustrates a high level view of the PCXSS method. The pre-processing phase decomposes every XML document into the 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 XML document and other objects (the existing 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 the common path coefficients (*CPSim*) between the document and all existing clusters. The second stage of the clustering phase groups the XML document 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 the 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. Other information such as data type and constraints are ignored. Secondly, the structure matching of the clustering phase measures the structural similarity between X_Paths of a document and of clusters considering only the exact match between element names. We do not consider the various semantic and syntactic meanings that an element name can have during the structure matching. We have shown elsewhere that semantics of an element name (such as person vs. people) in XML documents do not make any significant contribution when determining similarity between two XML documents [6].

3 PCXSS Phase 1: Pre-processing

All documents in the INEX collection or in the Wikipedia collection conform to only one DTD schema. As a result, we do not perform the pre-processing of element

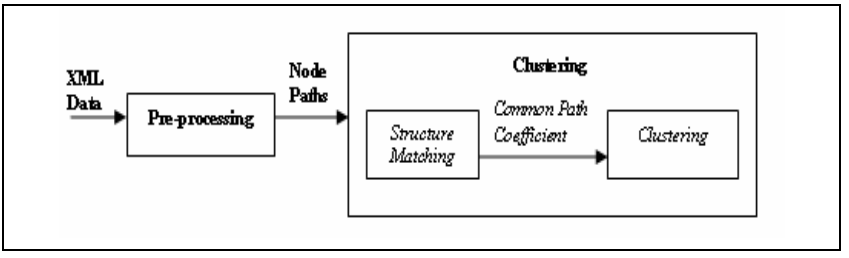


Fig. 1. The PCXSS Methodology

names while inferring the structure of the documents. Only a simple pre-processing step has been applied on the XML documents. An XML document is first parsed and modelled as the labelled tree (Fig. 2). The attribute of an element is modelled exactly the same way as its child elements. The tree is then decomposed into X_Paths to represent the structure of the XML document.

An X_Path is formally defined as an ordered sequence of tags 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 the XML document structure. Moreover, duplicated X_Paths in a document structure are eliminated. After the pre-processing of XML documents, documents are represented as a collection of distinct X_Paths.

4 PCXSS Phase 2: Clustering

The clustering phase consists of two stages: structure matching and clustering. At structure matching stage, the similarity between a XML document and existing clusters is measured. The output of this stage is 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 (1) the clustering number has not yet exceeded and (2) *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 process measures the similarity between the nodes in node paths by considering the name similarity only. While clustering XML schemas, PCXSS also includes the data type similarity (Tsim) and constraints similarity (Csim). As the INEX 2006 documents follow the same schema, neither semantic nor syntactic similarity computation is needed on the element name matching. Additionally, the exact matching process on element names saves a significant computation effort. Consequently, node matching depends on the exact match of the node names. For example, the last node at level 2 in Fig 2 is ‘bdy’. Consider another tree that contains

a node named as 'body'. If we compare these two trees, these two nodes will not be considered similar; however, they are syntactically similar. In a similar fashion, a node named as 'person' in one tree and a node named as 'people' in another tree will not be considered similar, although, they are semantically similar. The *NodeSim* value between element names is equal to 1 if they have an identical name else it is assigned with a 0.

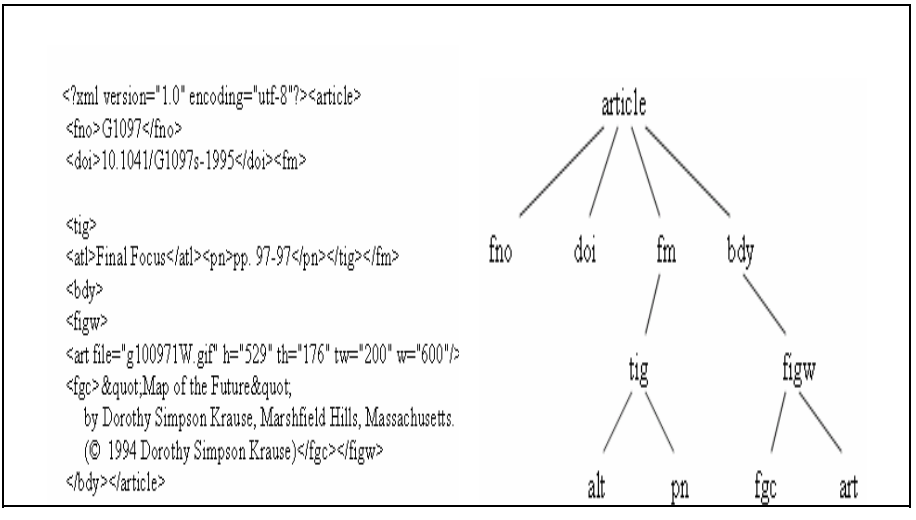


Fig. 2. An XML Document (article) & its Tree Representation

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 relationships between elements. The order of where the element resides in the structure is important in determining the structural similarity between the XML document and existing clusters.

The structural similarity between two XML documents 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 Fig. 3. *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:

$$Psim(P_1, P_2) = \frac{Max(CNC(P_1, P_2), CNC(P_2, P_1)}{Max(|P_1|, |P_2|)} \quad (1)$$

```
Function: CNC ( P1 , P2 )
Sim:= 0; for each n_i ∈ P1
  while j not end of P2 length
    if (NodeSim( n_i , n_j )) ==1
        Sim += NodeSim( n_i , n_j )
        j--
                  break from 'while' loop
    else
        j--
    end if
  end while
end for
return Sim
```

Fig. 3. The CNC function

Fig. 4 shows an example of traversing through the *CNC* function.

Consider two paths: Path1 (1/2/3/4/5/6) and Path2 (1/2/4/5/6). Path1 contains 6 element names that are showed as numbers for convenience. 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 equals to 1 (a match) then increase Sim by NodeSim coefficient 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 equals to 1 (a match) then increase Sim by NodeSim coefficient and repeat step 2 else move 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.

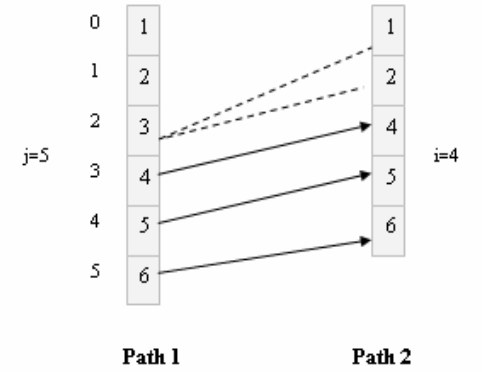


Fig. 4. Example of CNC Matching

The CNC function is not transitional. It means that $CNC(P_1, P_2)$ is not equal to $CNC(P_2,P_1)$. This is due to the fact that if the leaf element from P_1 can not be found in P_2 then no further matching is required. However, 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. As a consequence, 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:

$$CPSim(Tree_1,Tree_2) = \frac{\sum_{i=1}^{|TPath_1|} \max(\int_{j=1}^{|TPath_2|} Psim(P_i,P_j))}{Max(|TPath_1|,|TPath_2|)} \quad (2)$$

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:

$$CPSim(Tree,Cluster) = \frac{\sum_{i=1}^{|TPath|} \max(\int_{j=1}^{|CPath|} Psim(P_i,P_j))}{Max(|\,TPath\,|)} \quad (3)$$

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 a 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).

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.

Table 1. Test Data Sets

Test Data	No. of Classes	No.of XML documents	Size (MB)
INEX	18	6054	259
Wikipedia	60	75047	530

Evaluation methods. For the INEX XML Mining Challenge 2006, the clustering solutions are measured using the 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. For example, for binary classification, the precision (p), recall (r) and f1-measure are defined below, where A stands for the number of positive samples which are predicted as positive, B stands for the number of false negative samples which are predicted as positive, and C stands for the number of false positive samples which are predicted as negative

$$p = A / A + B \qquad r = A / A + C \qquad f1 = 2pr / p + r$$

In a multi-label classification, summing up A, B and C values from all binary classifications respectively and then these values are used to calculate f1 value is called micro-average f1 measure. The macro-average f1 is derived from averaging the f1 values from all binary classifications. Refer to paper [5] for more information on f1 measure for multi-label classification. Micro and macro f1 measures are

applied directly on multi-label classification solutions for evaluation. 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 3 results for the INEX test data and 1 result for the Wikipedia test data to the INEX XML Document Mining track 2006. The varied submissions were made due to the results obtained by setting different thresholds during experiments. The results of the clustering solutions performed by PCXSS are shown in Table 2.

Table 2. Results from INEX XML Mining Track 2006

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

The F1 measure of the clustering solutions obtained with PCXSS on the 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 is a reason for poor performances. The results do not seem to improve much by varying the threshold values.

Secondly, we eliminate attributes of an 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 somewhat improve the clustering results using the same thresholds. However, the results are not yet satisfactory. The reason for 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.

Table 3. Clustering Solution without the Attributes

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

The clustering solution using a 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 18 clusters that have been generated by PCXSS and the true categories.

Table 4. Mapping of 18 Clusters Discovered by PCXSS to its True Category

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

It shows that the documents in category 3 are widely spread out over the 18 clusters that have been discovered by PCXSS. This can 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 differences in structure and size. The PCXSS algorithm mainly derives the solution based on structure similarity. Moreover, the contents within tags play a significant role in measuring the similarity between documents of the INEX corpus in which documents conform to only one schema. We have ignored the contents within tags in our experiments.

To achieve some success, we tried another modification to the clustering algorithm. 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 somewhat better results than those in Table 3. It shows that the clustering on common paths on these kinds of data may not be sufficient enough without including the contents within tags.

PCXSS with the Iteration Phase. The XML documents are grouped according to CPSim between an XML document and existing trees. We do not include further iterations to refine the clustering process. Due to the absence of iteration phase, the clustering process highly depends upon the order of the data set and the clustering threshold. Consider this scenario: the clustering threshold is firmly set as 0.8 in the

Table 5. Results from the Modification of the Clustering Alogrithm in PCXSS

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

experiment. CPsim between two documents from the same domain is measured as 0.75 while processing. These documents are not considered to be grouped together according to this predefined threshold.

With the current PCXSS clustering process when the desired number of cluster is reached, for the remaining data set, PCXSS will not use the predefined threshold but will find the best similarity from the existing cluster that this remaining data set can be grouped into. This in turn creates a problem at the start when two documents belong to the same group are split into two different clusters. Due to this problem, the experiment is then extended the PCXSS clustering process by including the iteration phase.

The iteration works as follows: after the PCXSS clustering process ends (with the clustering number greater than the predefined one), the iteration phase starts by going through all the existing clusters and merging clusters together if their similarity is greater than the clustering threshold until the desired number of cluster is reached. At the end of the iteration phase if the number of existing clusters is still greater than the desired number of cluster, the iteration phase starts again and the clustering threshold will be decremented by 0.1 until the number of desired cluster is reached. Decrementing the clustering threshold can help to identify two clusters that contain documents from the same domain but have the similarity values lower than the rigid predefined clustering threshold. These two clusters can be merged together.

The experiment uses 0.7 for the clustering threshold and runs the PCXSS with the iteration phase on INEX 6054 test data. The micro and macro F1 of the clustering solution are 0.095 and 0.057 respectively, which are lower than PCXSS with no iteration phase shown in Table 2. We can argue here that the iteration phase proposed in this experiment is not suitable. The reasons are twofold: (1) XML documents from different categories contain many overlapping tags and (2) XML documents from the same category greatly vary in size. For example, XML documents from the 'an' category have an XML document that is 1KB and another document is 276KB in document size, where there is a big gap difference in both tags and content. These two documents surely can never be grouped together if XML documents from different categories have many overlapping tags and content.

During the testing and analysis of the INEX data set, it has been ascertained that even if PCXSS is extended by including contents in the clustering process, the clustering solution will not be that much better if no training or learning is done on

the INEX data set because two documents from the same category may contain different content and keywords (where semantic learning of the content or keywords may require). Thus, the INEX test data is more suitable for the classification task rather than for the clustering task.

Based on all the experiments above, it can be ascertained that 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 contents of documents such as vector space or neural networks may have been more appropriate here. The structure overlapping in the documents of the corpus due to deriving from the same schema and the large variations in the sizes and structures of documents from the same category also downplay the PCXSS clustering process.

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. Furthermore, INEX documents are data-centric based where the structure of the document plays a small role in determining the similarity between INEX documents.

The development of the PCXSS clustering algorithm originally meant to cluster the heterogeneous schemas. Use of PCXSS on the XML documents may need a number of extensions such as the learning of instance and data type for a more efficient clustering solution.

For future work, PCXSS will be extended to include the learning of content and to develop a more suitable iteration phase for the clustering process so that it is not highly depended on the predefined threshold. The effect of the size and of the order of the XML documents will also be thoroughly investigated in PCXSS. The PCXSS method will be appropriately modified to reduce those effects.

References

1. Boukottaya, A., Vanoirbeek, C.: Schema matching for transforming structured documents. In: 2005 ACM symposium on Document engineering. Bristol, United Kingdom (November 02-04, 2005)
2. Bray, T., et al.: Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation (2004)
3. Denoyer, L., Gallinari, P.: Report on the XML Mining Track at INEX 2005 and INEX 2006. In: INEX 2006 (2006)
4. Han, J., Kamber, M.: Data Mining. In: Concepts and Techiques, Morgan Kaufmann, Seattle, Washington, USA (2001)
5. Luo, X., Zincir-Heywood, N.: Evaluation of two systems on multi-class multi-label document classification. In: ISMIS05, New York, USA (2005)

6. Nayak, R.: Investigating Semantic Measures in XML Clustering. In: The 2006 IEEE/ACM International Conference on Web Intelligence. Hong Kong (December 2006)
7. Nayak, R., Tran, T.: A Progressive Clustering Algorithm to Group the XML Data by Structural and Semantic Similarity. To be published in International Journal of Pattern Recognition and Artifical Intelligence (Data of Acceptance: 9th October 2006)
8. Nayak, R., Witt, R., Tonev, A.: Data Mining and XML documents. In: The 2002 International Workshop on the Web and Database (WebDB 2002) (June 24-27, 2002)

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 [2,13,14]. 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 [11] 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 [15], Geamsakul et al. [9] and Bringmann and Zimmermann [5]. The first method computes frequent trees, orders these based upon some rule strength measures, and uses a decision list

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 485–496, 2007.

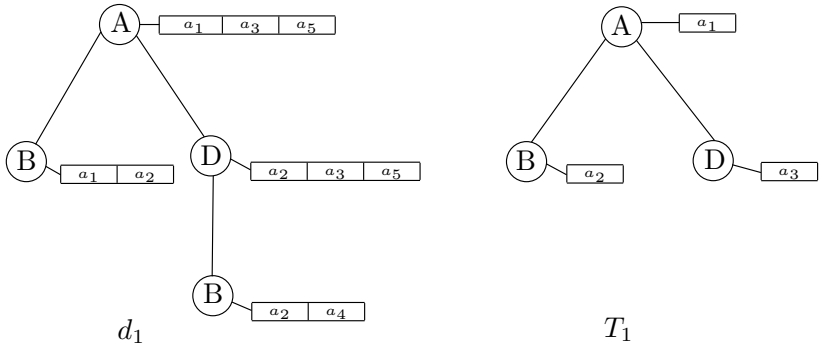


Fig. 1. T_1 is an induced subtree of the rooted ordered attribute tree d_1

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 [12].

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 [10] 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 [7] 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 : $\mathcal{A} = \{a_1, \ldots, a_n\}$, 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: $M : V \rightarrow \mathcal{P}(\mathcal{A})\backslash\{\varnothing\}$.

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 it's subtrees. Let $D = \{d_1, \ldots, d_m\}$ denote a database where each record $d_i \in D$, is a labeled rooted ordered attribute tree. Let $C = \{c_1, \ldots, c_k\}$ 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 $D_{\overline{c_i}}$ the set of records in the database is denoted that has a class label different form 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

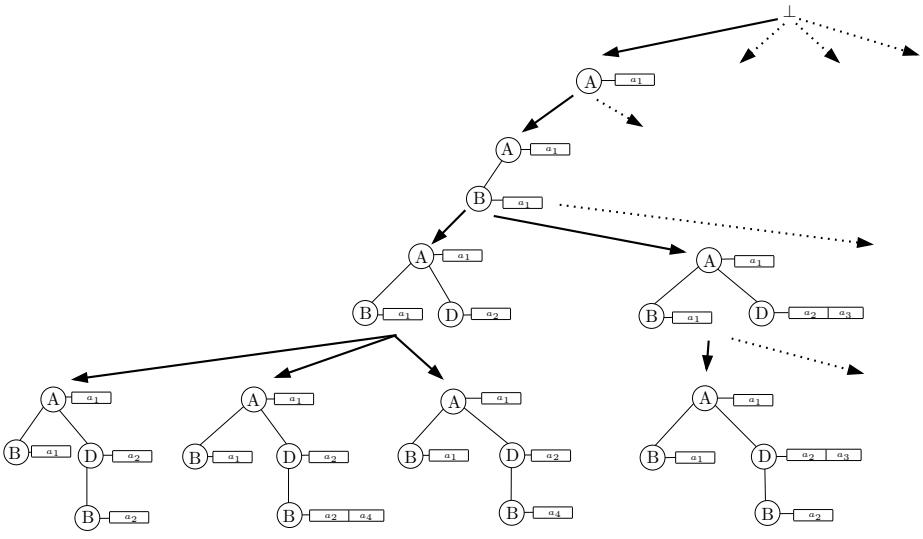


Fig. 2. Part of the enumeration tree for patterns on the tree d_1 (shown in figure 1)

mining 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 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,

```
Function LocMine ( X, OCL)
    if X = ∅
    then
        l ← 1
    else
        l ← k + 1
    do
        while l ≤ n
            if support(X ∪ a_l) ≥ minsup
            then
                (X ← X ∪ {a_l})
                return(X,ComputeOcc(X, OCL))
            else
                l ← l + 1
        Y ← X
        if X ≠ ∅
        then
            X ← X \ {a_k}
            l ← k + 1
    while Y ≠ ∅
return (∅, ∅)
```

```
Function ComputeOcc( X, OCL)
    out ← ∅
    for each d_i ∈ OCL
        for each Φ^j_{d_i} ∈ d_i
        if X ⊆ M(Φ^j_{d_i}(v_{k+1}))
        then
            out ← out ∪ Φ^j_{d_i}
return out
```

Fig. 3. The local mining algorithm

```
Algorithm FAT-miner(database D)
    out ← ∅
    C_1 ← candidate one patterns
    for each T ∈ C_1
        out ← out∪ Expand-Trees(T, D)
return out
```

```
Function Expand-Trees(T, D)
    out ← ∅
    do
        (Aset, Nocc) ← LocMine(M(T), occ(T, D))
        if |Nocc| ≥ minsup
        then
            M(v_{k+1}) ← Aset
            occ(T, D) ← Nocc
            out ← out ∪ T
            C_{k+1} ← candidates generated from T
            for each c_{k+1} ∈ C_{k+1}
                out ← out∪ Expand-Trees (c_{k+1})
    while |Nocc| ≥ minsup
return out
```

Fig. 4. The global mining algorithm

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 then 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 then or equal to 0.6. This property is also known as the confidence of the rule $T \rightarrow$ 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 [8], 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

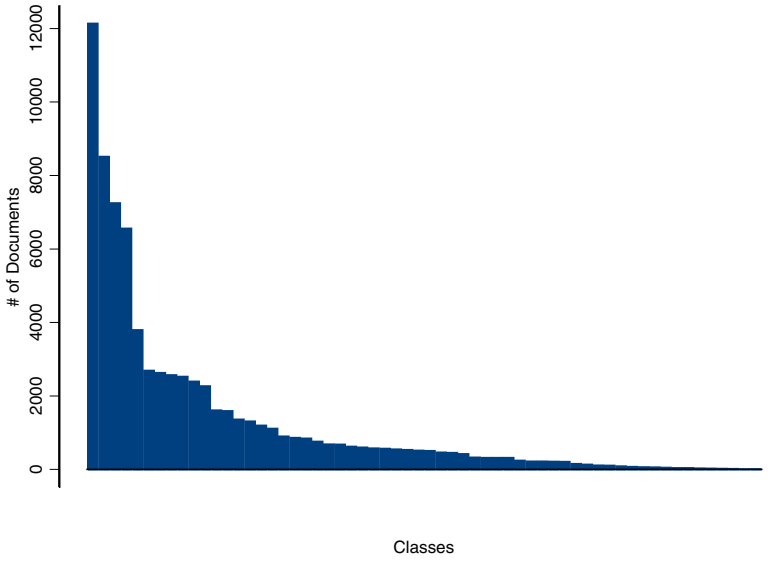


Fig. 5. Distribution of the XML documents over the different classes, classes are sorted according to the number of documents they contain

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 $T \rightarrow$ class. Another commonly used measure is the confidence of the rule $T \rightarrow$ 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 decision trees [12] to learn a classification model, more specifically the implementation provided by Borgelt [4]. This implementation uses a top down divide and conquer technique to build the decision tree. At each node in the decision tree, a number of attributes is selected—in a greedy manner—that best discriminates between the classes. As a selection criterion, information gain was used. Additionally, after construction of the decision tree we used confidence level pruning to reduce over fitting. The parameter used with confidence level pruning was set to 50%.

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 [7] as provided for the document mining [6] 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.

	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 ATR classifier was constructed with the usage of 4,762 distinct binary features of which 1966 were used in the resulting decision tree. Likewise for the NOATR classifier, these numbers were respectively 5,267 and 2,595. 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 then 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 values 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 insures 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 F1 measure per class are plotted, for the ATR and the NOATR classifier respectively. In both cases the classes were sorted according to the number of documents they contain. When comparing the performance of the classifiers per class, it is noteworthy that the best performance is achieved by classes of moderate size; for the classes that contained very few documents (the smallest six classes) both the classifiers were not able to retrieve any document at all. Furthermore, 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 an F1 score of 0.2853 while, for class 2257163 ("Portal:Pornography/Categories") an F1 score of 0.9345 was achieved. In order to accomplish a higher performance for the first class, we experimented 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 for all used classes.

Looking at the difference in F1 score per class for ATR and NOATR classifier(shown in figure 11), the ATR classifier substantially outer performs the NOATR classifier for almost every class. However, in two classes the score for the ATR classifier was lower than the score for the NOATR classifier. Hence, for

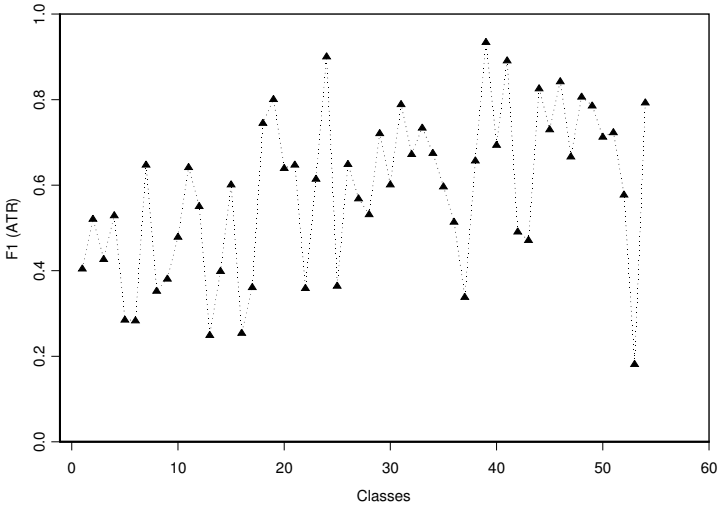


Fig. 9. The F1 score per class, for the experiments where we used the attributes of the data

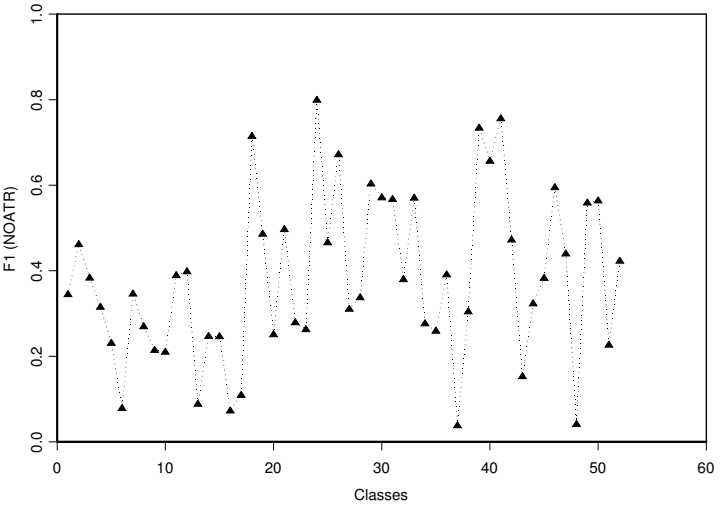


Fig. 10. The F1 score per class, for the experiments where the attributes of the data were left out

some classes the inclusion of attributes has a negative influence on the classification performance. A closer investigation of the emerging patterns for this class is needed to give a possible explanation.

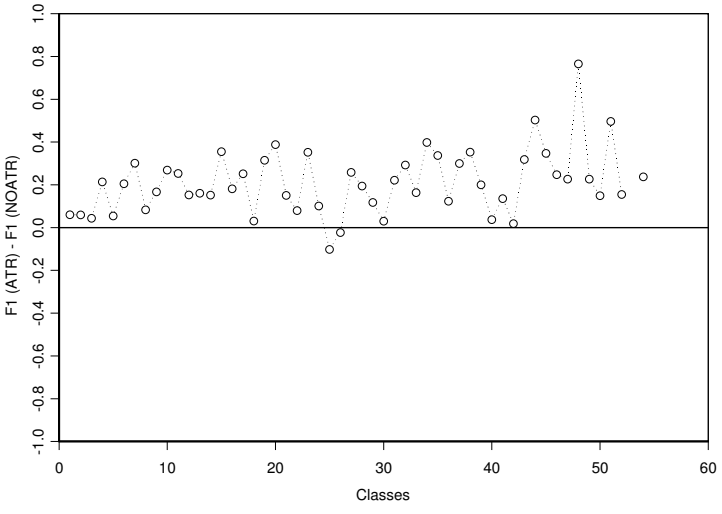


Fig. 11. The difference between the F1 score per class for the ATR classifier and the NOATR classifier. A positive value, corresponds to a higher F1 score for the ATR classifier compared with the NOATR classifier and vice versa.

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 generally greatly 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. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. 20th Int. Conf. Very Large Data Bases, VLDB, pp. 487–499 (1994)
2. Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H., Arikawa, S.: Efficient substructure discovery from large semi-structured data. In: SIAM Symposium on Discrete Algorithms (2002)
3. Bayardo, R.: Efficiently mining long patterns from databases. In: Laura, A. T., Haas, M. (eds.) SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, pp. 85–93 (1998)
4. Borgelt, C.: A decision tree plug-in for dataengine. In: Proc. 6th European Congress on Intelligent Techniques and Soft Computing (1998)

5. Bringmann, B., Zimmermann, A.: Tree^2 - decision trees for tree structured data. In: European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 46–58, (2005)
6. Denoyer, L., Gallinari, P.: Report on the xml mining track at inex 2005 and inex 2006. In: proceedings of INEX (2006)
7. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. SIGIR Forum (2006)
8. Dong, G., Li, J.: Efficient mining of emerging patterns: Discovering trends and differences. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 43–52 (1999)
9. Geamsakul, W., Yoshida, T., Ohara, K., Motoda, H., Yokoi, H., Takabayashi, K.: Constructing a decision tree for graph-structured data and its applications. Fundamenta Informaticae. 66(1-2), 131–160 (2005)
10. De Knijf, J.: FAT-miner: Mining frequent attribute trees. In: SAC '07: Proceedings of the 2007 ACM symposium on Applied computing to appear (2007)
11. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 80–86 (1998)
12. Ross, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
13. Wang, K., Liu, H.: Discovering structural association of semistructured data. Knowledge and Data. Engineering 12(2), 353–371 (2000)
14. Zaki, M.J.: Efficiently mining frequent trees in a forest. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80 (2002)
15. Zaki, M.J., Aggarwal, C.C.: Xrules: an effective structural classifier for XML data. In: Getoor, L., Senator, T. E., Domingos, P., Faloutsos, C. (eds.) ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 316–325 (2003)

Unsupervised Classification of Text-Centric XML Document Collections

Antoine Doucet[1,2] and Miro Lehtonen[2]

[1] IRISA-INRIA
Campus de Beaulieu
F-35042 Rennes Cedex
France
antoine.doucet@irisa.fr
[2] Department of Computer Science
P. O. Box 68 (Gustaf Hällströmin katu 2b)
FI–00014 University of Helsinki
Finland
miro.lehtonen@cs.helsinki.fi

Abstract. This paper addresses the problem of the unsupervised classification of text-centric XML documents. In the context of the INEX mining track 2006, we present methods to exploit the inherent structural information of XML documents in the document clustering process. Using the k-means algorithm, we have experimented with a couple of feature sets, to discover that a promising direction is to use structural information as a preliminary means to detect and put aside structural outliers. The improvement of the semantic-wise quality of clustering is significantly higher through this approach than through a combination of the structural and textual feature sets.

The paper also discusses the problem of the evaluation of XML clustering. Currently, in the INEX mining track, XML clustering techniques are evaluated against semantic categories. We believe there is a mismatch between the task (to exploit the document structure) and the evaluation, which disregards structural aspects. An illustration of this fact is that, over all the clustering track submissions, our text-based runs obtained the 1st rank (Wikipedia collection, out of 7) and 2nd rank (IEEE collection, out of 13).

1 Introduction

Document clustering has been applied to information retrieval for long. Most of this work followed the *cluster hypothesis*, which states that relevant documents tend to be highly similar to each other, and, subsequently, they tend to belong to the same clusters [1]. Clustering was then applied as pseudo-relevance feedback in order to retrieve documents that were not good direct matches to the query, but that were very similar to the best results [2]. Documents have to be clustered before querying, so as to form document taxonomies.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 497–509, 2007.

The quantity of data organized with an XML structure grows drastically. While XML document collections have essentially been data-centric, there are now more and more text-centric document collections. The necessity for tools to manage these collections has grown correspondingly. Clustering is one way to automatically organize very large collections into smaller homogeneous subsets.

In this paper, we explore a number of ways to exploit the structural information of XML documents so as to improve the quality of unsupervised document classification. 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, which was enhanced with different types of textual and structural features. We propose to combine them at once or or in a 2-step approach, first using the structural features, and then the textual ones.

We present the corresponding results in the context of the INEX 2006 document mining track. We extend our contribution with the integration of a measure of the "textitude" of a structured document.

Because we require no document markup description (such as a document type definition — DTD), our techniques are particularly suited for experiments with several different collections, such as the ones used in the mining track 2006: the IEEE journals collection and the Wikipedia collection.

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 there may be many ways to split a document collection that are 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 by using textual features exclusively. The paper discusses this issue and elaborates on why the results should be analyzed carefully.

Section 2 covers related work. Our experimental setting and the methods to be evaluated are presented in Section 3. The performance of these techniques in the context of the INEX mining track 2006 are presented in Section 4, where we also discuss a number of issues and difficulties that are to be encountered when evaluating XML clustering. We draw conclusions and present the future directions of our work in Section 5.

2 Related Work

Until recently, most of the research on structured document processing was focused on data-centric XML (see for example [3] and [4]). One early motivation for XML document clustering was to gather documents that were structurally similar, so as to generate a common DTD for them. Nierman and Jagadish notably proposed a tree-edit distance as a structural similarity measure of XML documents [5].

The birth of the INEX mining track in 2005 [6] provided an experimental framework very much needed for the case of text-centric document collections [7]. This triggered research at the crossroads of information retrieval, machine learning and XML databases.

There are currently two main approaches to text-centric XML document clustering. One of them is to build models naturally close to the XML tree structure, such as neural networks [8], including self-organizing maps [9]. The other approach relies on a transformation of the document structure into a flat vector space representation, before applying well-known clustering techniques [7,10,11,12]. Previous work has proposed to use element labels as the structural features, and to combine them into word term features in a common $tfidf$ framework [7]. Candilier et al. [12] proposed more advanced structural features, such as parent-child or next-sibling relations. Vercoustre et al. [11] proposed to represent an XML tree by its different sub-paths, with features such as the path length, or the number of nodes it contains. An open problem for such techniques is to find a good way to combine the structural and textual features.

3 Procedure of the Experiments

The document model we used was the vector space model. In other words, we represented documents by N-dimensional vectors, where N is the number of document features in the collection.

Using this document model and the k-means algorithm, we performed our clustering experiments with various feature sets in one and two steps. We will now describe the clustering algorithm and then present the different ways we used it.

3.1 Clustering Technique

We chose to use the k-means algorithm for our experiments. *K-means* is a commonly used partitional clustering technique, where k is the number of desired clusters, either given as input, or determined in the loop. In the experiments, for simplicity and to allow easier comparison, we set k to be equal to the desired number of classes. The algorithm relies on a initial partition of the collection that is repeatedly readjusted, until a stable solution is found.

In these experiments, we mainly decided to use k-means because of its linear time complexity and the simplicity of its algorithm.

Given k desired clusters, k-means techniques provide a one-level partitioning of the dataset in linear time ($O(n)$ or $O(n(log\ n))$ where n stands for the number of documents[13]). The *base* algorithm presented in Figure 1 takes the number of desired clusters as input.

3.2 Run Descriptions

As our aim is to take into account both the semantics of the text and its structural markup, we naturally build two corresponding feature sets. Therefore, we

1. *Initialization*:
   - k points are chosen as initial centroids
   - Assign each point to the closest centroid
2. *Iterate*:
   - Compute the centroid of each cluster
   - Assign each point to the closest centroid
3. *Stop condition*:
   - As soon as the centroids are stable

Fig. 1. Base k-means algorithm

need to use two baseline runs: one relies on a text-only representation, and the other on a structure-only representation.

- **Text features only**: These features are the result of a typical (unstructured) text representation. We removed the stop words, and then stemmed the remaining words using the Porter algorithm. The dimensions of the vector space are the remaining single word terms, in a canonical form.
- **Tag features only**: This representation uses the XML element labels as the dimensions of the vector space (stopwords are not removed and labels are not stemmed).

The rest of our runs are tentative ways to combine the information of unstructured text to that of the structural indicators. A simple way to do so is to combine the text and tag features into a single vector space. In other words, this approach consists in merging the bag of words and the bag of tag names. We name this representation "**text+tags**". This naïve approach serves as a baseline combination of textual and structural data. Note that we prevent the confusion between word features and tag features (the "art" element name should not be confused with the word "art").

We will now present two more advanced techniques. The first one was originally presented in 2002, at a time when no formal evaluation framework existed for XML mining experiments. We decided to revisit it in the framework of the INEX mining track. The second technique is new, it introduces a structural indicator in the context of the unsupervised classification process: the T/E measure [14].

The 2-step approach. Previous experiments suggested that a 2-step approach, "**tags → text**" (read "tags then text"), permits to obtain better results, by putting aside structural outliers before running the textual (semantic) classification [7]. The algorithm is described in Figure 2. To use tag features exclusively is very noisy when most of the XML elements have a purely stylistic role, as is the case in the IEEE collection. The technique presented here permits to benefit from the structural information of documents, with the internal similarity threshold as a safe-guard. Only the most cohesive tag-based clusters will be kept, while the rest of the clustering process is achieved based on text content.

a *Input*:
  - A document collection
  - n, the final number of desired clusters
  - σ, the internal similarity threshold

b *Step one, tag-based clustering*:
  - Based on tag-features only, perform k-means with $k = n$
  - Keep the m clusters with an internal similarity higher than σ

c *Step two, text-based clustering*:
  - Based on text-features only, perform k-means with $k = n - m$

d *Finally*:
  - The m tags-based clusters and the $(n - m)$ text-based clusters are combined to form the final n-clustering

Fig. 2. The 2-step approach: tags then text

In practice, this algorithm is as fast as a text-based n-clustering (often faster). This is due to the fact that tag-based clustering is very efficient thanks to a representation with a very small number of features.

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 [14]. 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. The T/E measure is a quite reliable indicator when the XML fragment is relatively small, e.g. a paragraph of text or a small section. As the size or the heterogeneity of the fragment increases, a single T/E value starts to shift from being an exact indicator towards being an approximation.

Integrating the T/E measure into the vector space model. We integrate the T/E measure as an additional dimension of the vector space. Because our weighting scheme is based on inverted document frequency, the inclusion of the T/E value for every document would have a null effect. Therefore, we only gave a non-null value to the T/E dimension if the T/E measure was greater than 1.

4 Evaluation

In this section, we will define the experimental settings. We will first describe the document collections and the evaluation measures, and then present our results in details.

4.1 Collection Description

The INEX mining track 2006 provides two separate XML document collections. The first one is a collection of scientific journal articles from the IEEE Computer Society[1]. The second one is a collection of English documents from Wikipedia [15]. Each collection further includes a set of categories C. Every document is assigned to a subset of categories of C that describe it best. The goal of the clustering task is to automatically produce a categorization that matches these (*ideal*) assignments as closely as possible.

A specificity in the INEX mining track 2006 is that there is exactly one category corresponding to every document. In other words, each collection is partitioned into category-wise subcollections.

Let us now describe the two collections in further details.

IEEE. The IEEE collection has long been known as the "INEX collection", because it was the only collection in use in the main INEX track from the first INEX initiative in 2002 until 2006. It contains approximately 12,000 articles published in 18 different IEEE journals. They are mainly marked up with hierarchical and stylistic elements. The hierarchical markup typically indicates the beginning and end of sections, subsections and paragraphs, and possibly their titles, as well as figures and bibliographical references. Stylistic elements, for instance, are used to mark bolded text or mathematical formulas.

The categories that were used to partition the collection are the journals in which a document was originally published. Hence, every document is assigned to exactly one category.

As we pointed out earlier [7], we believe that these categories are not fully satisfying, as the fact that a paper was published in a given journal does not necessarily mean that it is entirely irrelevant to every other journal. Among other things, such a strict interpretation means we should assume that a paper published in "Transactions on Computers" cannot possibly have anything in common with a paper published in "Transactions on Parallel&Distributed Systems".

Moreover, the IEEE collection contains documents of different types. The most common document type is scientific articles, but the collection also contains calls for papers, book reviews, keyword indices, etc. Regardless of their nature, documents published in the same journal are assigned to the same category. An intuitive issue with this "ideal" categorization is that any clustering assigning documents by their nature will be penalized in the evaluation process.

Wikipedia. The Wikipedia collection is new to INEX 2006. It contains 150,094 English documents from Wikipedia. The collection used in the mining track is a subset of the "main" Wikipedia collection as described by Denoyer and Gallinari [15]. The main collection contains 659,388 documents and covers a hierarchy of 113,483 categories. It contains about 5,000 different tags, with an

[1] http://www.computer.org/

average number of 161 XML nodes per document and an average element depth of 6.72.

The subset of the Wikipedia collection used in the INEX mining track consists of the 150,094 documents to which only one semantic category corresponds. These categories were extracted from the Wikipedia portals, which include 72 semantic categories (the 113,483 categories mentioned earlier come from a different source, check [15] for details). After the removal of documents to which more than 1 category was attached, only 60 non-empty classes remained. This partition is used as the evaluation gold-standard.

Naturally, we may express similar concerns as the ones we expressed earlier about the IEEE collection. The assumption that a document should belong to one and only one category does not seem right when we are handling text. To use a partition as our ideal classification implies the assumption that no two categories have anything in common. This can hardly be right when those categories are based on semantics.

4.2 Evaluation Measures

As we mentioned earlier, the evaluation of the clustering track relies on the comparison of a given run to an ideal classification. The theoretical gold standards for each collection were described in the previous subsection. We shall now introduce the metrics of this comparison. In the INEX mining track, two official measures were used to compare an ideal classification and an experimental run: the micro- and macro-average F1 measures. We define these measures below.

Recall and precision. For a given category, we define the positives (respectively, negatives) as the set of documents assigned to that category (respectively, not assigned to that category).

When we compare a submission to the ideal classification, we define the true positives (TP) as the positives that were assigned to the right category. The false positives (FP) are the documents that were wrongly assigned to that category. Similarly, the true negatives (TN) were duly assigned to another category, while the false negatives (FN) should have been assigned to the category being considered.

Precision and recall are defined as follows:

$$Precision = \frac{TP}{TP + FN}, \; Recall = \frac{TP}{TP + TN}.$$

F1-measure. Precision and recall complement each other. For instance, it is easy to obtain very high scores with one, at the expense of the other. To get perfect recall, one can simply assign every document to every category. In a symmetrical fashion, one may obtain high precision by limiting the number of answers. Hence, we rather utilize a measure that combines precision and recall, such as the F1-measure, defined as the harmonic average of recall and precision:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}.$$

Micro- and macro-average. To obtain a single measure for the evaluation of a classification, the F1 measure needs to be averaged over all the classes. There are two ways to do this. *Macro-average F1* is the non-weighted average of the F1 measure over all the classes, while *micro-average F1* is weighted by the number of documents in each corresponding class. Clearly, the latter is more strongly influenced by larger classes.

4.3 Experimental Results

We submitted the runs previously described to the INEX mining track 2006. The document features were weighted with inverted document frequency, and the Cluto software[2] was used to perform the k-means clusterings.

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.

Our results are summarized in Table 1 for the IEEE collection and in Table 2 for the Wikipedia collection. The notation "Tags→Text, 0.8" means that we used the "tags then text" approach, and kept the tag-based clusters with an internal similarity higher than 0.8.

Table 1. INEX - IEEE Collection

Features	Micro F1	Macro F1	Overall rank (out of 13)
Text	.348789	.290407	3rd
Tags	.132453	.081541	7th
Text+Tags	.253369	.203533	5th
Tags→Text, 0.8	.270350	.222365	4th
Text + T/E	.348828	.290379	2nd

4.4 Result Analysis

Overall observations. Looking at the results of our submissions, we can make a number of observations. An obvious disappointment is the fact that the exclusive use of text features beats all the other alternatives by far. Even worse, when we look at "text", "tags", "text+tags" and "text+T/E", it seems like the performance decreases as the number of structural features increases.

On the positive side, we could confirm that tag-based clustering is very fast, and that using the "tags then text" approach performs just as fast as using text features only.

Another positive result is that "tags then text" outperforms "text+tags". This result is especially satisfying because both approaches use exactly the same features. Consequently, we get confirmation that the 'tags then text" technique is a better way to integrate structural features into the clustering process.

[2] CLUTO, http://www-users.cs.umn.edu/$\sim$karypis/cluto/

Table 2. INEX - Wikipedia Collection

Features	Micro F1	Macro F1	Overall rank (out of 7)
Text	.444455	.210621	1st
Tags	.221829	.072834	6th
Text+Tags	.372376	.128239	5th
Tags→Text, 0.8	.406129	.155034	4th
Tags→Text, 0.9	.413439	.159473	3rd
Text + T/E	.427438	.183567	2nd

The explanation is fairly simple. The structural clustering can detect and put aside what we may call "structural outliers". Typically, in the IEEE collection, they are tables of contents and keyword indices of journal issues as well as calls for papers. In the Wikipedia collection, the outliers include lists (lists of counties by area, list of English cricket clubs, etc.). However, to count on a small number of element names as unique document descriptors is obviously risky. This is why we ensure that only the most cohesive tag-based clusters are kept, by using a high internal similarity threshold.

Comparison with other participants. In the INEX mining track, a total of 7 clustering runs were submitted for the Wikipedia collection and 13 for the IEEE collection.

On the Wikipedia collection. As shown in table 2, our 6 Wikipedia submissions rank at the first 6 places of the INEX clustering track 2006. Only one other team submitted a run for the Wikipedia collection. This is mostly due to scalability issues. Several approaches are indeed based on XML tree operations, which are computationally complex and may become intractable with a shift from $12,017$ documents (IEEE) to $150,094$ (Wikipedia), combined with the fact that the structure of the Wikipedia documents is much deeper and much more unpredictable (from 163 distinct elements to $7,208$). Another reason that might have discouraged potential participants is the lack of a DTD. This is, however, a very common feature of real-life collections.

One strong point in our approach is that it does not use anything but the documents themselves. From a computational point of view, clustering is the costliest operation with a linear time complexity of $O(n)$ or $O(n \log n)$.

Hence we had no problems shifting from one collection to the other and we do not expect difficulties in applying this work to new collections, whatever their structure and size is, since our techniques scala well.

On the IEEE collection. Two other teams submitted clustering runs for the IEEE collection. The overall ranking of our runs is given in table 1. The best-performing method is based on contextual self-organizing maps [9]. Its performance is fairly close to our own best, with a micro-average F1 of 0.365079 and a macro-average of 0.326469. The technique proves to be efficient. However, its complexity makes

it hardly scalable (the supervised learning actually needs to be restricted to a subpart of the IEEE document trees: the content of the "fm" element).

Conclusion. We should be quite happy to see our runs in the top ranks for both collections. However, the fact that our best run is always the one that actually ignores the structural information is rather worrisome. We believe that this is not necessarily due to a weak state of the art of the systems presented in the INEX clustering task, but for a big part to a semantic bias of the evaluation system.

4.5 Discussion on Evaluation

Evaluation of clustering. There are two ways to evaluate clustering experiments. The first one is to use *internal quality measures*, such as entropy, purity, or cohesiveness. For instance, the cohesiveness of a cluster is the average similarity between each two documents in the cluster. The problem of these measures is that the computation of document similarities is strongly dependent on the document model. Internal quality measures are useful to compare clustering techniques based upon the same document model, but they are meaningless in most other cases. In particular, they cannot help as we wish to compare techniques based on the same algorithm but different feature sets.

In such situations, we must rely on *external quality measures*, such as recall, precision, or F1-measure. The latter were the official evaluation metrics for the clustering task of the INEX mining track in 2006.

Gold-standard. External measures are meant to compare every submitted clustering to a "gold-standard" classification. The more similar a run is to that standard, the better. Defining such a gold-standard is a great challenge.

Indeed, we are not convinced that the gold-standard classifications that were used for the evaluation of the INEX mining track are optimal. The consequence of this is very important, because to improve a system's performance with respect to the F1-measure means to produce a classification closer to the gold-standard. If the gold-standard is weak, improving the performance of a system might actually require that a number of reasonable assumptions be compromised.

What is a good clustering? The main issue with the current "gold" classifications is the use of disjoint clusters. This is an excessive simplification when we are dealing with text and thematic classes, as is the case currently.

In fact, having to deal with thematic classes can also be seen as a problem. Since the motivation of XML clustering is to take structural information into account, we should also consider categories that are not solely based on semantics.

An empirical analysis of our clusters show that the technique "tags then text" manages to put aside outliers, such as tables of contents or call for papers in the IEEE collection. Our technique stores these into clusters of their own and performs text-based clustering with the remaining documents. We do believe that this is a good result for most uses of the document collection. Thinking of

information retrieval, it is likely that a user performing a search on a scientific journal is looking for articles (or fragments thereof) rather than keyword indices or calls for papers. However, with respect to the current evaluation metrics, the effectiveness of a system taking this fact into account is weakened, because the gold-standards were built the opposite way: each call for papers belongs to the journal in which it was published. Hence, they are spread out uniformly in the ideal classification, while we actually built a system that puts all of them in the same category.

The problem is that if the gold-standards were solely based on the document structure, separating calls for papers, tables of contents and regular articles, one would as well be able to argue that it does not make sense to have to categorize the call for papers for a data mining conference is in a different class from that of a data mining article. The key issue is there. Given a document collection, there are numerous "perfect" ways to classify the contents. The classification needs to be related to a certain need, but it may still be totally inappropriate in other situations.

This leads us to the conclusion that the intended use of XML document clustering needs to play a larger role in its evaluation, and hence a prior question needs to be answered: why do we do it? If the goal of XML clustering is to build a semantic-based disjoint taxonomy, then the current gold-standards are suitable. In order to detect DTDs automatically, we would need a structure-oriented gold standard. For information retrieval, we might use the per topic relevance judgements as the classes, or perform indirect evaluation through the ad hoc XML IR runs.

5 Conclusion - Future Work

Our conclusive remarks and suggestions concern two topics, our XML clustering approach and the general problem of the evaluation of XML clustering. Actually, the evaluation of supervised classification is also related, even though the learning phase can help compensate for the issues aforementioned.

We have introduced the experiments with our XML clustering techniques in the context of the INEX 2006 mining track. The generality and scalability of our approach was underlined by the fact that we made no difference in the way we handled two radically different document collections, whereas many participants have been discouraged by the size and depth of the Wikipedia document collection (perhaps also by the lack of a DTD). One weakness of our techniques is their flat use of the structural information. We created a "bag of structure" and implemented advanced ways to use it as a complement of the "bag of words", but we ignored the tree structure of the elements and did not either connect the words to their path in the XML tree. This is left for future work.

For both collections, we had the satisfaction to see our runs in the top ranks. Looking at the top 5 runs for the two collections, the only one that is not ours occupies the 1st rank for the IEEE collection. The "tags then text" approach was demonstrated to be more efficient at combining semantics and structure,

than a baseline merger of the features, in spite of a tendency to contradict some of the arguable implications of the current evaluation system. Hence, in a more appropriate evaluation setting, we expect to observe the same phenomenon with an even greater margin. Evidently, this remains to be verified.

A source of concern should be the fact that our best performing runs were the ones that actually ignored the structural information. However, we feel that this only reflects the bias of the evaluation system. Indeed, micro- and macro-average F1 are measuring the closeness of a run to a theoretically ideal classification. However, the current "ideal" classifications in use are disjoint and thematic. Since there is no evidence that the classifications we use as gold-standards are related to the structure of the documents, it is natural that the best performing approaches are the ones that simply ignore that structure.

An important point is that several classifications of the same collection may be perfect, depending on the context. We hence plead for placing the applications of XML clustering in the center of the evaluation process. This may be done by creating an ideal classification for every corresponding application, and/or by evaluating XML clustering indirectly, by measuring how much we can benefit from it in another task. In 2006, an INEX "user case studies track" was created. Perhaps a comparable reflection is now needed in the XML mining track.

References

1. Jardine, N., van Rijsbergen, C.: The use of hierarchic clustering in information retrieval. Information Storage and Retrieval 7, 217–240 (1971)
2. Tombros, A.: The effectiveness of hierarchic query-based clustering of documents for information retrieval. PhD thesis, University of Glasgow (2002)
3. Guillaume, D., Murtaugh, F.: Clustering of XML Documents. Computer Physics Communications 127, 215–227 (2000)
4. Yi, J., Sundaresan, N.: A classifier for semi-structured documents. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, Massachusetts, pp. 340–344 (2000)
5. Nierman, A., Jagadish, H.: Evaluating Structural Similarity in XML. In: Fifth International Workshop on the Web and Databases (WebDB 2002), Madison, Wisconsin (2002)
6. Denoyer, L., Gallinari, P.: Report on the xml mining track at inex 2005 and inex (2006). [16]
7. 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 Dagsuhl, Germany, pp. 81–87 (2002)
8. Yong, S.L., Hagenbuchner, M., Tsoi, A., Scarselli, F., Gori, M.: Xml document mining using graph neural network. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX LNCS, vol. 3977, Springer, Heidelberg (2006)
9. Kc, M., Hagenbuchner, M., Tsoi, A.C., Scarselli, F., Gori, M., Sperduti, A.: Xml document mining using contextual self-organizing maps for structures. [16]
10. Despeyroux, T., Lechevallier, Y., Trousse, B., Vercoustre, A.M.: Experiments in clustering homogeneous xml documents to validate an existing typology. In: Proceedings of the 5th International Conference on Knowledge Management (I-Know), Vienna, Austria, Journal of Universal Computer Science (2005)

11. Vercoustre, A.M., FEGAS,: M., Lechevallier, Y., Despeyroux, T.: Classification de documents xml à partir d'une représentation linéaire des arbres de ces documents. In: Actes des 6èmes journées Extraction et Gestion des Connaissances (EGC 2006), Revue des Nouvelles Technologies de l'Information (RNTI-E-6), Lille, France (2006)
12. Candillier, L., Tellier, I., Torre, F.: Transforming xml trees for efficient classification and clustering. In: INEX 2005 Workshop on Mining XML documents (2005)
13. Willett, P.: Recent trends in hierarchic document clustering: a critical review. In Information Processing and Management 24, 577–597 (1988)
14. Lehtonen, M.: Preparing Heterogeneous XML for Full-Text Search. ACM Transactions on Information Systems 24, 1–21 (2006)
15. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. SIGIR Forum (2006)
16. Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.): Advances in XML Information Retrieval and Evaluation, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, December 18-20 2006, Revised Selected Papers. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX. Lecture Notes in Computer Science, Springer (2007)

XML Document Mining Using Contextual Self-organizing Maps for Structures

M. Kc[1], M. Hagenbuchner[1], A.C. Tsoi[2], F. Scarselli[4], A. Sperduti[3], and M. Gori[4]

[1] University of Wollongong, Wollongong, Australia
{wmc01,markus}@uow.edu.au
[2] Monash University, Melbourne, Australia
act@hkbu.edu.hk
[3] University of Padova, Padova, Italy
sperduti@math.unipd.it
[4] University of Siena, Siena, Italy
{franco,marco}@dii.unisi.it

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 in 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. These tasks are part of an international competition on XML clustering and categorization (INEX 2006). It will be shown that the proposed approaches provide a suitable tool for the clustering of structured data as they yield the best results in the international INEX 2006 competition on clustering of XML data.

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 and specifications. 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, say, a UNIX application can be viewed or edited by any other systems, e.g. a Window system, 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 considered 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>Hello<B>World</B></A> gives a document which has the string "Hello World" as its content where the word "World" shares the property <A> of the word

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 510–524, 2007.

"Hello" but also is being assigned the property <B>. 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><B></A></B>. 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 model that is capable of clustering structured information in an unsupervised fashion. The neural network model is known as the Self-Organizing Map for Structured Data (SOM-SD) [3] and its extension: Contextual SOM-SD (CSOM-SD) [5] 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 [7].

In 2005 it was shown for the first time that a SOM-SD based model can produce exceptional good performances on XML classification tasks by winning the international competition INEX 2005 by having the best overall performance compared with the performance of other approaches [6]. 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 performed 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 by modifying the connection strengths among the biological neurons to represent the information contained in the inputs (stimuli). Such an approach has the obvious advantage of not requiring a system to hard-code the information and thus is much more flexible. 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 and application) environment. For example, ML methods are popularly applied in human-machine interfaces such as speech recognition systems. No two

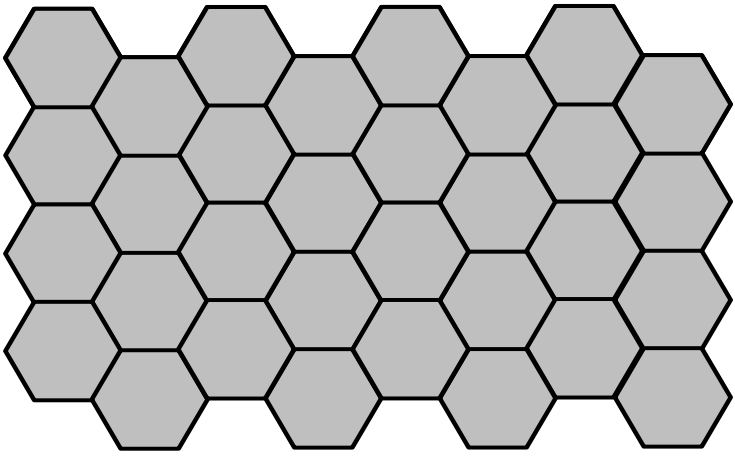


Fig. 1. The architecture of a simple Self-Organizing Map

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 input samples together with associated desired outputs. An iterative training approach is commonly applied with the aim to adjust internal system parameters such that for a given input the desired output 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 Artificial Neural Network (ANN) methods, are proven to be able to approximate any input-output relationship to an 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 which share certain properties are grouped together. The most famous and most widely used of the unsupervised neural network methods is the Self-Organizing Map (SOM) and its many variants developed originally in [7].

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 $\mathbf{m}_i^T = (m_{i1}, \ldots, m_{in})^T$, 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 [7]. 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 [7]. 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 [7]. 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 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 input 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 two 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:

$$r = \arg\min_i \|\mathbf{u} - \mathbf{m}_i\| \tag{1}$$

where $\|.\|$ is the Euclidean distance.

Step 2. Cooperative step. After the best matching codebook $\mathbf{m}_r$ has been found, all codebook vectors are updated. The winning vector $\mathbf{m}_r$ 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 a 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 [1]. The updating algorithm is given by:

$$\Delta\mathbf{m}_i = \alpha(t) f(\Delta_{ir})(\mathbf{m}_i - \mathbf{u}) \tag{2}$$

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 neighborhood function $f(\cdot)$ can take the form of a Gaussian function:

$$f(\Delta_{ir}) = \exp\left(-\frac{\|\mathbf{l}_i - \mathbf{l}_r\|^2}{2\sigma(t)^2}\right) \tag{3}$$

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-the neuron in the lattice. Other neighborhood functions are possible.

[1] 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 ([7], page 111).

Steps 1 and 2 together constitute a single training step and they are repeated until the training process ends. There are a number of ways in which the training can terminate. One way is to fix the number of iteration steps. In this case, 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 most 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 [3] produced Self-Organizing Maps which can process graph structured data without requiring the pre-processing of such data. This will be addressed in the following section.

3 Self-organizing Maps for Structures

Work on SOM methods capable of mapping structures was inspired by developments in supervised neural networks for graphs [2]. 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) [3] 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 [7]. 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:

$$r = \arg\min_i \|(\mathbf{x}_v - \mathbf{m}_i)\mathbf{\Lambda}\| \tag{4}$$

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 [3].

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:

$$d = \sqrt{\mu_1 \sum_{i=1}^{p} (u_i - m_i)^2 + \mu_2 \sum_{j=1}^{2o} (y_{ch_j} - m_{n+j})^2} \tag{5}$$

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 [3].

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 or cyclic graphs, a much more general class of graphs than tree structures.

A first solution was proposed in [4]. In [4] 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 iterations[2], 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 [5]. The method proposed in [5] 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 $\boldsymbol{y}_{ch[v]}$, and all the states of parents $\boldsymbol{y}_{pa[v]}$ 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 data label that is attached to this node, initialize $\boldsymbol{y}_{ch[v]}$ with the states of this node's offsprings (which is available from the previous iteration), the $\boldsymbol{y}_{pa[v]}$ 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 point.

Step 2. Train the SOM-SD on every node in the training set as usual ensuring that the vector components $\boldsymbol{y}_{ch[v]}$ and $\boldsymbol{y}_{pa[v]}$ are updated by using the states from the previous iteration, and ensuring that the components $\boldsymbol{l}$, $\boldsymbol{y}_{ch[v]}$, and $\boldsymbol{y}_{pa[v]}$ are weighted so as to balance their influence on the Euclidean distance measure similar to that described in Section 3.

Step 2 is iterated at least N_d times, where N_d is the 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.

[2] This is particularly the case during the final stages of the learning procedure.

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 [5] 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 Initiative for the Evaluation of XML Retrieval (INEX) [1] organized an international competition on XML clustering and categorization in 2006. This paper addresses the clustering challenge. 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. The software which was developed for the experiments on the SOM-SD and CSOM-SD is available from http://www.artificial-neural.net.

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. In particular, the input dimension grows twice as fast as the (maximum) number of children at any node in a graph. The dimension of codebook vectors grow at the same rate. Since these vectors are multiplied as in Eq.5, the computational complexity grows quadratic with the number of children. Note that the computational complexity grows linear with the number of nodes in the graphs, and hence, any optimization in the representation of graphs should address the outdegree. While in general 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 INEX-2005 challenge [6], which consolidates repetitive sub-structures. However, due to the large variation of structures and tag positions in the set of INEX-2006 documents, this approach did not significantly decrease the expected training times. Another drawback of this approach is that repeating sub-structures may be a significant feature, and compressing that feature could impact the clustering performance.
2. **Constructing tag-based connectivity graph:** 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 substructures: 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.

A common problem with Self-Organizing Maps is that training parameters need to be determined using a trial and error approach. This paper proposes a more sophisticated approach by statistically analysing the properties of the dataset which helps to make a reasonable assumption about the underling difficulty of the learning problem. Then training parameters which should be most suitable for this given task are set. Absolutely no information about target values or class memberships are used during this task, and hence, the approach remains unsupervised.

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 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 observed 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.

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

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 also computed. 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

Table 2. Training parameters used for the clustering 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

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 $y_j \in \{t_1, \ldots, t_q\}$ 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.

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.

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	Doucet and Lehtonen	K-means, Tags Text
0.18	0.12	Tran, Nayak, Raymond	PCXSS with clustering threshold 0.8
0.17	0.12	Tran, Nayak, Raymond	PCXSS with clustering threshold 0.9
0.13	0.08	Doucet and Lehtonen	K-means, Tags only
0.13	0.09	Tran, Nayak, Raymond	PCXSS with clustering threshold 0.4
0.10	0.05	Tran, Nayak, Raymond	PCXSS with clustering threshold 0.3

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 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.

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. The approaches taken by the other teams were as follows: (1)Team Doucet and Lehtonen from IRISA-INRIA, France, and from the University of Helsinky, repectively use K-means with different features (see Tables). They adopt a two step procedure, first they cluster according to a set of features, then according to another. The feature *TE* is “a structural indicator of the proportion of mixed content in an XML fragment”. (2) Team Tran, Nayak, Raymond from the Queensland University of Technology, Australia use a method called Progressively Clustering XML by Structural Similarity (PCXSS). The approach is an iterative method which uses a measure of the distance between a pair of trees which roughly counts the number of common paths from the root to the leaves. A cluster center is represented by a set of common paths. Clusters are built while

Table 4. Test results when clustering using XML structure and document content

Micro F1	Macro F1	Team	Method used
0.35	0.29	Doucet and Lehtonen	K-means, text + TE
0.35	0.29	Doucet and Lehtonen	K-means, text only
0.25	0.20	Doucet and Lehtonen	K-means, text and tags
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	Tran, Nayak, Raymond	PCXSS with clustering threshold 0.8
0.09	0.04	Tran, Nayak, Raymond	PCXSS with clustering threshold 0.7
0.07	0.04	Tran, Nayak, Raymond	PCXSS with clustering threshold 0.5

reading the input trees. At each step of this procedure the input tree is either inserted into the closest cluster or into a new cluster, if the minimum distance to the existing clusters is over a given threshold.

As the tables show, our team obtained the best performance in the structure only clustering. In fact, the performances obtained by using XML structure only also outperformed 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

SOMs have traditionally been useful for many data mining tasks due to the linear nature of the learning algorithm, their generalization ability, and due to their insensitivity to noise. With SOM-SD, this property is maintained with the exception of a quadratic dependence on the connectivity of nodes in a graph. The computational complexity grows quadratic with the outdegree a node in the dataset. In this paper it was shown that through the removal of redundancies in the data set it is possible to reduce the outdegree significantly, and hence, rendering SOM-SD practical for the given task. Thus, 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 preprocessing 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.

It is recognized that the out-degree can be an issue for certain learning tasks. For example, when processing the graph depicted by the World Wide Web, the outdegree can be very large and the removal of redundancies can be a difficult task. A project with addresses this issue is under way. The general idea is to alter the processing technique. Instead of utilizing the state of each offspring as a network input, the state space of the (finite) network is used. Such an approach keeps the input dimension constant.

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. Denoyer, L., Gallinari, P.: Report on the xml mining track at inex 2005 and inex 2006. In: Proceedings of INEX (2006)
2. Frasconi, P., Gori, M., Maggini, M., Martinelli, E., Soda, G.: Inductive inference of tree automata by recursive neural networks. In: Proceedings of the Fifth Congress of the Italian Association for Artificial Intelligence, Rome, Italy, pp. 425–432. Springer, Heidelberg (1997)
3. Hagenbuchner, M., Sperduti, A., Tsoi, A.: A self-organizing map for adaptive processing of structured data. IEEE Transactions on Neural Networks 14(3), 491–505 (May 2003)
4. Hagenbuchner, M., Sperduti, A., Tsoi, A.: Contextual processing of graphs using self-organizing maps. In: European symposium on Artificial Neural Networks, Poster track, Bruges, Belgium (April 27-29, 2005)
5. Hagenbuchner, M., Sperduti, A., Tsoi, A.: Contextual self-organizing maps for structured domains. In: Workshop on Relational Machine Learning (2005)
6. Hagenbuchner, M., Sperduti, A., Tsoi, A., Trentini, F., Scarselli, F., Gori, M.: Clustering xml documents using self-organizing maps for structures. In: Fuhr, N., (eds.) LNCS, vol. 3977, pp. 481–496. Springer, Heidelberg (2006)
7. Kohonen, T.: Self-Organizing Maps. Springer Series in Information Sciences, vol. 30. Springer, Heidelberg (1995)

XML Document Transformation with Conditional Random Fields

Rémi Gilleron, Florent Jousse, Isabelle Tellier, and Marc Tommasi

INRIA Futurs and Lille University, LIFL, Mostrare Project
first.last@univ-lille3.fr, jousse@grappa.univ-lille3.fr

Abstract. We address the problem of structure mapping that arises in XML data exchange or XML document transformation. Our approach relies on XML annotation with semantic labels that describe local tree editions. We propose XML Conditional Random Fields (XCRFs), a framework for building conditional models for labeling XML documents. We equip XCRFs with efficient algorithms for inference and parameter estimation. We provide theoretical arguments and practical experiments that illustrate their expressivity and efficiency. Experiments on the Structure Mapping movie datasets of the INEX XML Document Mining Challenge yield very good results.

1 Introduction

Semi-structured documents in XML are omnipresent in today's computer science applications, since XML has become the standard format for data exchange. The essence of XML documents is their tree structure. Machine learning tasks dealing with XML structures should account for this fact. This is why *tree labeling* has become one of the most basic tasks in the XML context. It arises in information extraction from structured documents on the Web. The basic idea is to label nodes that are selected for some role positively and all others negatively. Another application is learning-based schema matching [1], where XML data for a source schema are labeled w.r.t. a target schema. In this case, matches are to be elaborated into mappings to enable XML data translation. In this paper, we consider the problem of XML document transformation which is ubiquitous in XML document engineering. For the INEX structure mapping task, the problem is to learn to transform layout-oriented HTML documents into content-oriented XML documents. The transformation task can be modeled by a labeling task, by assigning an operational semantic (local deletion, local inversion, etc.) to every element of the labeling alphabet. Thus we reduce the problem of learning to transform in the problem of learning to label.

The labeling task can be described as follows: given an observable $\mathbf{x}$, the problem consists in finding the most likely labeling $\mathbf{y}$, of the same dimension. Solutions to this problem based on graphical generative models try to evaluate from the labeled examples the joint probability distribution $p(\mathbf{y}, \mathbf{x})$. As we are only interested in labeling, we prefer to use conditional models that model

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 525–539, 2007.

the conditional distribution $p(\mathbf{y}|\mathbf{x})$ directly. Thus, for learning to label trees, we propose to extend on Conditional Random Fields (CRFs) introduced in [2]. Until now, CRFs have mainly been applied to sequence labeling tasks occurring in computational linguistic applications such as part-of-speech tagging, shallow parsing, but also to information extraction [3,4,5,6,7]. For an overview, see [8].

In this paper, we develop XCRFs, a new instance of CRFs that properly accounts for the inherent tree structure of XML documents. As a matter of fact, in an XML document, every node has an unlimited number of ordered children, and a possibly unbounded number of unordered attributes. Independence conditions in CRFs are governed by an undirected graph over random variables for labelings. The undirected graph for XCRFs is defined by: for ordered (parts of the) trees, the maximal cliques of the graph are all triangles consisting of a node and two adjacent children; for unordered (parts of the) trees, the maximal cliques are edges consisting of a node and one child. With such an undirected graph, in XCRFs, $p(\mathbf{y}|\mathbf{x})$ can be decomposed into a product of potential functions, each applying on a one-node clique, or an edge clique or a triangular clique. And these potential functions are themselves defined thanks to feature functions and parameters.

A contribution of this paper is to adapt the technical apparatus associated with CRFs to this new kind of dependence relationships. We define efficient algorithms for the inference problem and the parameter estimation problem in XCRFs. Because of the unranked property of XML trees, algorithms for XCRFs implement two recursions: a vertical recursion following the child ordering and an horizontal recursion following the sibling ordering using both forward-backward variables and inside-outside variables.

We have implemented XCRFs in a freely available system that allows to (learn to) label XML documents. In the experiments section, we evaluate our model on the Movie datasets from the Structure Mapping task of the INEX XML Document Mining challenge. This task consists in transforming layout-oriented HTML documents into data-oriented XML documents. To perform this task, we model such a transformation by a labeling of the HTML documents. We evaluate both the quality of the labeling of the HTML documents and the complete transformation. Results show that XCRFs perform very well on this task.

Related work

The idea to define CRFs for tree structured data has shown up recently. Basically, works differ in the graphical structure of CRFs. In [9], output variables are independent. Other approaches such as [10,11] define the graphical structure on rules of context-free or categorial grammars. [12] have considered discriminative context-free grammars, trying to combine the advantages of non-generative approaches (such as CRFs) and the readability of generative ones. All these approaches apply to ordered ranked rather than unranked trees. As far as we know, their graphical models are limited to edges, not accounting for father-child-next-sibling triangles as in XCRFs.

XML data translation takes place in the domain of semantic integration. An overview can be found in [1]. For schema mapping, it has been shown [13]

that XCRFs can be compared with LSD [14]. But, applications of CRFs to more complex tasks and a comparison with recent systems for schema matching remain to be done. For information extraction, systems [15,16,17,18,19,20] deal with semi-structured input, mainly HTML data, but, to the best of our knowledge, structured output have not been considered so far by machine learning techniques.

For XML document transformation, Chidlovskii and Fuselier [21] address the problem of semantic annotation of HTML documents according to a target XML schema. They use a two-step procedure: terminals (leaves of the output document) are predicted by a maximum entropy classifier, then the most likely output tree is generated using probabilistic parsing for probabilistic context-free grammars. They suppose that leaves are in the same order in the input and the output document. The complexity of probabilistic parsing is cubic in the number of leaves, and therefore the system is not appropriate for large XML trees. Gallinari *et al* have considered generative stochastic models in [22]. Such models need to model input documents and to perform the decoding of input documents according to the learned model. The complexity of the decoding procedure could be prohibitive for large XML documents.

2 Conditional Random Fields for xml Trees

2.1 Conditional Random Fields

We refer to [8] for a complete introduction to CRFs. A CRF is a conditional distribution with an associated graphical structure. Let $\mathbf{X}$ and $\mathbf{Y}$ be two random fields, let $\mathcal{G}$ be an undirected graph over $\mathbf{Y}$. Let $\mathcal{C}$ be the set of all cliques of $\mathcal{G}$. The conditional probability distribution is of the form:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{y}_c, \mathbf{x})$$

where ψ_c is the potential function for the clique c and $Z(\mathbf{x})$ is a normalization factor. Each potential function has the form:

$$\psi_c(\mathbf{y}_c, \mathbf{x}) = \exp\Big(\sum_k \lambda_k f_k(\mathbf{y}_c, \mathbf{x}, c)\Big)$$

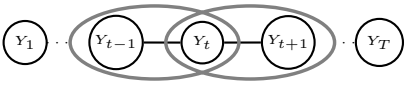
for some real-valued parameter vector $\Lambda = \{\lambda_k\}$, and for some set of real-valued feature functions $\{f_k\}$. This form ensures that the family of distributions parameterized by Λ is an exponential family. The feature function values only depend on $\mathbf{y}_c$, *i.e.* the assignments of the random variables in the clique c, and the whole observable $\mathbf{x}$.

The two main problems that arise for CRFs are:

Inference: given a new observable $\mathbf{x}$, find the most likely labeling $\hat{\mathbf{y}}$ for $\mathbf{x}$, *i.e.* compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$.

Training: given a sample set S of pairs $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$, learn the best real-valued parameter vector Λ according to some criteria. In this paper, the criterion for training is the maximum conditional penalized log-likelihood.

Linear chain crfs



In first-order linear chain CRFs, the maximal cliques of the graph are pairs of consecutive nodes as depicted on the right. Thus, there are feature functions over node labels called node features, and feature functions over pairs of labels called edge features. For ease of notation, node features and edge features are merged and f_k denotes the k^{th} feature. The conditional probability can be written as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\Big(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(y_{t-1}, y_t, \mathbf{x}, t)\Big) \tag{2.1}$$

In first-order linear chain CRFs, the inference task can be performed efficiently and exactly by the standard dynamic-programming Viterbi algorithm. For training linear chain CRFs, the problem is, given an input sample S, to learn the parameter vector Λ which maximizes the log-likelihood. The parameter vector can be learnt using traditional log-likelihood maximization methods. Since the optimal parameters cannot be found analytically, gradient ascent techniques are used. [23] has experimentally shown that the most effective technique in the case of linear-chain CRFs is the limited memory BFGS algorithm (L-BFGS) [24]. Both the function $Z(\mathbf{x})$ in the likelihood and the marginal distributions in the gradient can be computed by forward-backward techniques.

2.2 XCRFs: Conditional Random Fields for xml Trees

XML documents are represented by their DOM tree. We only consider element nodes, attribute nodes and text nodes of the DOM representation. Other types of nodes[1] are not concerned by labeling tasks. Attribute nodes are unordered, while element nodes and text nodes are ordered. We identify a node by a position which is an integer sequence n and we denote by x_n the symbol in a tree $\mathbf{x}$ in position n. The k ordered children of a node in position n are identified by positions $n.1$ to $n.k$. As a running example, consider the two XML trees $\mathbf{x}$ (on the top) and $\mathbf{y}$ (on the bottom) in Figure 1. The set of nodes for both trees is $\{\epsilon, 1, 1.1, 1.2, 2, 2.1, 2.2, 2.3, 2.4\}$, ϵ being the root. The symbol in position 2.1 in $\mathbf{x}$ is `td`; in its labeling $\mathbf{y}$, the label in position 2.1 is `name`. $\{2.1, 2.2, 2.3, 2.4\}$ is the set of children of 2. Ordered children (2.1, 2.2 and 2.3) are listed before unordered children (2.4).

With every set of nodes, we associate a random field $\mathbf{X}$ of observable variables X_n and a random field $\mathbf{Y}$ of output variables Y_n where n is a position. The realizations of X_n will be the symbols of the input trees, and the realizations of Y_n will be the labels of their labelings. In the following, we freely identify realizations of these random fields with ordered unranked trees.

[1] comments, processing instructions...

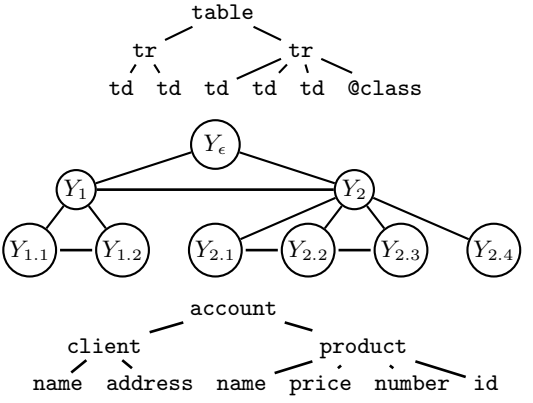


Fig. 1. An ordered unranked tree, its graph and its labeling

For their ordered parts, the structure of XML trees is governed by the next-sibling and the child orderings. We translate this structural property into XCRFs defining maximal cliques of the undirected graph over an ordered unranked tree $\mathbf{y}$ to be triangles $(Y_n, Y_{n.i}, Y_{n.(i+1)})$, for $i < o$, where o is the number of ordered children of n. For unordered parts of XML trees, the graph only includes pairs $(Y_n, Y_{n.i})$ because the next-sibling ordering is meaningless. In Fig. 1, we show the graph for our running example. Feature functions are thus defined over nodes (node features), pairs of nodes (edge features) and triples of nodes (triangle features). Triangle feature functions have the form: $f_k\big(y_n, y_{n.i}, y_{n.(i+1)}, \mathbf{x}, (n, n.i, n.(i+1))\big)$. Their arguments are the labels assigned to the node n and to two of its consecutive children $n.i$ and $n.(i+1)$, the whole observable $\mathbf{x}$, and the identifier of the clique in the tree $(n, n.i, n.(i+1))$. In fact, a triangular clique $(n, n.i, n.(i+1))$ can be shortly identified by $n.i$. We denote by $\mathcal{C}$ the set of cliques in the dependency graph. Every feature function f_k is associated with a real-valued parameter λ_k, defining the vector $\Lambda = \{\lambda_k\}$. It is worth pointing out that our model uses the same set of feature functions with the same parameters for every clique in the graph.

The introduction of triangle features is the main difference between linear chain CRFs and XCRFs. Training and inference algorithms need to be adapted because these triangle features bring both a horizontal and a vertical recursion in the graph structure. Therefore for ease of notation, we only consider such triangle features in the formal definition of the algorithms. The conditional probability distribution for an XCRF can be written as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{n.i \in \mathcal{C}} \psi_{n.i}(y_n, y_{n.i}, y_{n.(i+1)}, \mathbf{x}) \tag{2.2}$$

where

$$\psi_{n.i}(y_n, y_{n.i}, y_{n.(i+1)}, \mathbf{x}) = \exp\Big(\sum_k \lambda_k f_k(y_n, y_{n.i}, y_{n.(i+1)}, \mathbf{x}, n.i)\Big) \tag{2.3}$$

and

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \Big(\prod_{n.i \in \mathcal{C}} \psi_{n.i}(y_n, y_{n.i}, y_{n.(i+1)}, \mathbf{x}) \Big) \tag{2.4}$$

2.3 Algorithms for XCRFs

Inference Algorithm for $Z(\mathbf{x})$. The normalization factor $Z(x)$ must be computed to compute the normalized conditional probability $p(\mathbf{y}|\mathbf{x})$ and to compute the likelihood in training CRFs. In XCRFs, $Z(\mathbf{x})$ is defined in Eq. (2.4) as a sum over all possible labelings. It can be efficiently computed using dynamic programming. To do so, for every node n and every label $y \in \mathcal{Y}$, we define the *inside variable* $\beta_n(y)$ as the sum of the unnormalized probabilities over all the possible labelings of the subtree rooted in node n in which n is labeled with y. The recursive definition of $\beta_n(y)$ is 1 if n is a leaf, and $\beta_n(y) = \sum_{(y_1,\ldots,y_m)\in\mathcal{Y}^m} \Big(\prod_{i=1}^m \beta_{n.i}(y_i)\psi_{n.i}(y, y_i, y_{i+1}, \mathbf{x}) \Big)$ if n has m children. Clearly, the sum over all possible label assignments for the children of n again leads to a combinatorial explosion. Therefore we also use a dynamic programming technique for the horizontal recursion. For every node n with m children, for every $k \leq m$, and for every pair of labels $(y, y') \in \mathcal{Y}^2$, we define the *backward variable* $\beta'_{n,k}(y, y')$ as the sum of the the unnormalized probabilities over all labelings of the subtree rooted in the node n whose $k-1$ first subtrees are deleted and where n is labeled with y and $n.k$ with y'. If $k = m$ we have $\beta'_{n,k}(y, y') = \beta_{n.m}(y')$, and otherwise

$$\beta'_{n,k}(y, y') = \beta_{n.k}(y') \sum_{y''\in\mathcal{Y}} \Big(\psi_{n.k}(y, y', y'', \mathbf{x})\beta'_{n,k+1}(y, y'') \Big)$$

Therefore, $\beta_n(y) = \sum_{y,y_1\in\mathcal{Y}\times\mathcal{Y}} \beta'_{n,1}(y, y_1)$

Thus, $Z(\mathbf{x}) = \sum_{y\in\mathcal{Y}} \beta_\epsilon(y)$ can be computed in $O(N \times M^3)$ where N is the number of nodes of $\mathbf{x}$ and M is the number of labels in $\mathcal{Y}$.

Inference for XCRFs. When computing $Z(\mathbf{x})$, the aim was to compute the sum of the unnormalized conditional probabilities for all labelings. Here, we want to compute the most likely labeling. The Viterbi recursion is obtained by replacing the `sum` function by the `max` function in the inside and backward recursions of the computation of $Z(\mathbf{x})$. Finding the most likely labeling $\hat{\mathbf{y}}$ then consists in the memorization of the Viterbi path associated with the maximum unnormalized conditional probability.

Training XCRFs. Training an XCRF means learning its parameter vector Λ. We are given iid training data S of pairs of the form (observable tree, labeled tree). Parameter estimation is typically performed by penalized maximum likelihood. The conditional log-likelihood, defined as $\mathcal{L}_\Lambda = \sum_{(\mathbf{x},\mathbf{y})\in S} \log p(\mathbf{y}|\mathbf{x}; \Lambda)$, is used. This function is concave and the global optimum is the vector of parameters with which the first derivative is null. However, finding analytically this

derivative with respect to all the model parameters is impossible. A gradient ascent (L-BFGS), which requires the calculation of the partial derivatives of $\mathcal{L}_\Lambda$ for each parameter, is therefore used. Replacing $p(\mathbf{y}|\mathbf{x};\Lambda)$ by its definition (*c.f.* equation (2.2)), $\mathcal{L}_\Lambda$ becomes:

$$\mathcal{L}_\Lambda = \sum_{(\mathbf{x},\mathbf{y})\in S} \sum_{n.i\in\mathcal{C}} \sum_k \lambda_k f_k(y_n, y_{n.i}, y_{n.(i+1)}, \mathbf{x}, n.i) - \sum_{(\mathbf{x},\mathbf{y})\in S} \log Z(\mathbf{x}) \ . \quad (2.5)$$

Thus partial derivatives can be written as:

$$\frac{\partial \mathcal{L}_\Lambda}{\partial \lambda_k} = \sum_{(\mathbf{x},\mathbf{y})\in S} \sum_{n.i\in\mathcal{C}} f_k(y_n, y_{n.i}, y_{n.(i+1)}, \mathbf{x}, n.i) - \sum_{(\mathbf{x},\mathbf{y})\in S} \sum_{n.i\in\mathcal{C}} \sum_{y_1,y_2,y_3} P(y_1,y_2,y_3) f_k(y_1,y_2,y_3,\mathbf{x},n.i)$$

where $P(y_1,y_2,y_3) = p(Y_n = y_1, Y_{n.i} = y_2, Y_{n.i+1} = y_3|\mathbf{x};\Lambda)$. The computation of the first term is relatively straightforward. On the opposite, calculating the second one, *i.e.* the marginal probability for a given clique, is more difficult since it requires to sum over all possible labelings outside the clique. To make these computations tractable, we introduce a dynamic programming algorithm using both forward-backward variables and inside-outside variables.

For every node n and every label y, the *outside variable* $\alpha_n(y)$ is the sum of all the unnormalized probabilities over all the possible labelings of the context of the subtree rooted at the node n, in which n is labeled with $y \in \mathcal{Y}$. We have $\alpha_n(y) = 1$ if n is the root and $\alpha_n(y) = \sum_{y'\in\mathcal{Y}} \alpha_{n'}(y') \frac{\beta_{n'}(y')}{\beta_n(y)}$ if $n = n'.i$.

The *forward variable* $\alpha'_{n,k}(y,y')$ is introduced for horizontal recursion. It is defined as the sum of the unnormalized probabilities over all labelings of the subtree rooted in the node n whose $(m-k+1)$ last subtrees are deleted and where n is labeled with y and $n.k$ with y'. We have $\alpha'_{n,k}(y,y') = 1$ when $k = 1$, otherwise

$$\alpha'_{n,k}(y,y') = \sum_{y''\in\mathcal{Y}} \Big(\beta_{n.(k-1)}(y'')\psi_{n.k-1}(y,y'',y',\mathbf{x})\alpha'_{n,k-1}(y,y'')\Big)$$

Then, the marginals can be computed by:

$$P(y_1,y_2,y_3) = \frac{1}{Z(\mathbf{x})}\alpha_n(y_1)\psi_{n.i}(y_1,y_2,y_3,\mathbf{x}) \alpha'_{n,n.i}(y_1,y_2)\beta_{ni}(y_2)\beta'_{n,n.(i+1)}(y_1,y_3) \quad (2.6)$$

Complexity Issues and Discussion. We have shown that $Z(\mathbf{x})$ can be computed in $O(N \times M^3)$ where N is the number of nodes of $\mathbf{x}$ and M is the number of labels in $\mathcal{Y}$. This result can be extended to the computation of the marginal

probabilities in the gradient. This leads to an overall complexity for training in $O(N \times M^3 \times G)$ where N is the total number of nodes of the trees in the input sample S, M is the number of labels in $\mathcal{Y}$, and G is the number of gradient steps. For linear chain CRFs only a factor M^2 occurs.

We have presented the inference algorithms for XCRFs as extensions of the algorithms for linear chain CRFs. An alternative approach would be to consider XCRFs as a particular case of general CRFs. Indeed, the treewidth of undirected graphs for XCRFs is 2. For every graph associated with an XCRF, a junction tree can be computed in linear time. Then the belief propagation algorithm can be applied [25,7]. The inference algorithms for XCRFs, given in the previous section, can be considered as inference algorithms for general CRFs using the knowledge of the tree-shaped graphical structures associated with XCRFs.

3 Experiments with the XCRF System

3.1 The XCRF System

The XCRF model is implemented by a freely available JAVA library [2]. For training, parameters are estimated by maximizing the penalized log-likelihood. The L-BFGS gradient ascent package from the "RISO Project"[3] is used. The system allows to label element, attribute and text nodes of XML trees. An XCRF is specified by an XML file. Feature functions are 0-1 valued functions defined by XPATH expressions. There are node features, edge features, attribute features (edge features for unordered children), and triangle features. An example of triangle feature is given in Fig. 2.

```
<Feature name="f_title" weight="7.46" xsi:type="TriangleFeature">
 <Y value="0" />
 <Yi value="title" />
 <Yj value="year" />
 <TestX value="parent::*/name() = 'tr'"/>
 <TestX value="name() = 'td'"/>
 <TestX value="name(following-sibling::*[1])='td'"/>
</Feature>
```

Fig. 2. XML definition of a feature function of weight 7.46 defined by $f_{\text{title}}(y_n, y_{n.i}, y_{n.(i+1)}, \mathbf{x}, n.i) = 1$ if $y_n = \perp$, $y_{n.i}$ ='title', $y_{n.(i+1)}$ ='year', x_n ='tr', $x_{n.i}$ ='td', $x_{n.(i+1)}$ ='td'

3.2 Feature Generation

In the following experiments, the set of feature functions we used were automatically generated from a set of labeled documents, typically the learning set.

[2] http://treecrf.gforge.inria.fr/
[3] http://riso.sourceforge.net/

Table 1. Structure Attributes computed during the preprocessing

Attribute	Description
nbChildren	number of children of the node
depth	depth of the node
childPos	node is the i^{th} child of its father

Table 2. Text Attributes computed during the preprocessing

Attribute	Description
containsComma	text contains a comma
containsColon	text contains a colon
containsSemiColon	text contains a semi-colon
containsAmpersand	text contains an ampersand
containsArobas	text contains an arobas
isUpperCase	text is in upper case
firstUpperCase	first letter is in upper case
onlyDigits	text is made of digits
oneDigit	text is a single digit
containsDigits	text contains digits
rowHeader	text value of the row header in a table (HTML only)
columnHeader	text value of the column header in a table (HTML only)

There are essentially two kinds of generic feature functions: structure features and attribute features.

Before the automatic generation of the feature functions, a first step consists in preprocessing additional attributes. These attributes are purely informative. Therefore, during the experiments, the XCRFs do not consider them as regular nodes and do not attempt to label them. These preprocessed attributes give additional information on the structure of the trees and basic information on the content of the text nodes. Tables 1 and 2 show the different kind of information given by these preprocessed attributes.

The first kind of automatically generated feature functions are structure features. These feature functions are node features, edge features and triangle features testing solely the node symbols (nodes can be element, attribute or text nodes) and labels. Let 1_p be 1 if and only if predicate p is true. If the tree in Figure 3 is part of the learning set, the following features are generated:

- Node features testing the label of node n its the node symbol, *e.g.* $1_{(y_n=\texttt{title})(x_n=\texttt{td})}$
- Node features similar to the previous one, but testing if the node symbol is different from the one in the learning set, *e.g.* $1_{(y_n=\texttt{title})(x_n\neq\texttt{td})}$
- Edge features testing the labels of a node n and one of its children i and the node symbols, *e.g.* $1_{(y_n=\perp)(y_i=\texttt{title})(x_n=\texttt{tr})(x_i=\texttt{td})}$

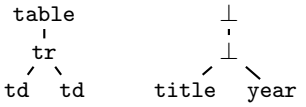


Fig. 3. Feature generation example

– Triangle features on the same principle as the edge features above, but testing on two consecutive children i and j of node n:
  $1_{(y_n=\bot)(y_i=\texttt{title})(y_j=\texttt{year})(x_n=\texttt{tr})(x_i=\texttt{td})(x_j=\texttt{td})}$

The second kind of feature functions that are generated are attribute features. These feature functions are based on attribute values. The attributes used to generate these features are both the ones originally in the corpus, for instance the `class` attribute of a `div` element in HTML, or attributes resulting from the preprocessing performed earlier. With all these attributes, for each node in the tree we generate feature functions testing the label assigned to this node and one or two attributes of the node itself, its father, grandfather, great-grandfather, previous sibling and next sibling. For instance, on the example in Figure 3, the following feature functions are generated:

$$1_{(y_n=\texttt{title})(father(x_n)@nbChildren=2)(x_n@childPos=1)}$$

$$1_{(y_n=\texttt{title})(x_@ndepth=2)}$$

With such a generation procedure, we get feature functions that are mostly domain independent. However, they prove to be a very relevant set of feature functions when dealing with well-structured XML documents.

3.3 Experiments on the Structure Mapping Task

This second experiment is on a bigger scale. It was conducted as part of the Structure Mapping task of the XML Document Mining Challenge [26]. The Structure Mapping task consists in transforming layout-oriented HTML documents into content-oriented XML documents. The dataset is made of HTML descriptions of movies taken from the website allmovie[4] and their XML counterpart taken from the IMDB repository[5]. Each document gives thorough information about a single movie such as the title, the director, the cast and crew, related movies, *etc.* Since the DTD of the output XML documents was not given as part of the challenge, we used the algorithm given in [27] to build it. The DTD contains 63 elements, among which 39 contain textual information.

There are two tasks with two different input HTML datasets, called html1 and html2. Both HTML datasets describe the same movies, but the documents in html1 contain only the HTML table describing the movie, whereas the documents in html2 also contain useless information. Documents from the html1 dataset are

[4] http://www.allmovie.com/
[5] http://www.imdb.com

therefore subtrees of the documents in the html2 dataset. This makes the task a bit harder for html2. The average number of nodes in a document is 470 in the html1 dataset, and 530 in the html2 dataset.

Method. We chose to model a transformation from an HTML document into an XML document by a labeling of the input HTML document.

In the HTML documents, a text node can contain several distinct information, which will appear in different text nodes in the output XML document. For instance, the release date, the movie format and the movie length are all located in the same text node of the HTML document. Therefore, as a preprocessing step, text nodes are tokenized in several nodes.

The labels used in the labeling task are: the names of the 39 elements of the output DTD which contain textual information, *e.g.* `director`, `synopsis`; the labels indicating that a node is part of a set of consecutive nodes corresponding to a single element of the output DTD,*e.g.* `director_continued`, `synopsis_continued`; the label $\perp$ for useless nodes. This leads to a total number of 79 labels. However, only 66 of them are useful, since some information are never split over several text nodes in the HTML documents (*e.g.* the year of release, the duration, *etc.*).

A very simplified example of a labeled HTML tree is given in Figure 4. The XML output documents are then computed using a simple post-processing function which takes a labeled HTML document and the output DTD as parameters.

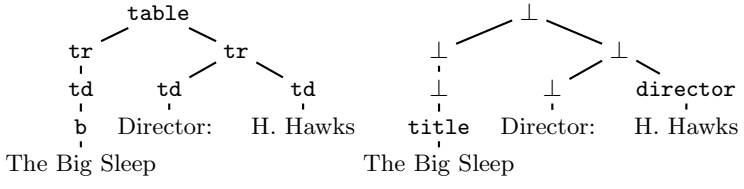


Fig. 4. a (simplified) HTML input tree (left) and its labeling (right) for the Structure Mapping task of the XML Mining challenge

We assume that the text nodes appear in the same order in the input HTML document and in its output XML counterpart, which is the case in this Structure Mapping task. With this assumption, one can easily build the learning set of labeled HTML documents from the original learning set of (HTML, XML) pairs of documents. Indeed, automatically finding the labeling of the HTML document corresponding to the transformation is straightforward. One only has to parse both the HTML and the XML documents at the same time and label the text leaves of the HTML document which occur in the XML document.

For the testing phase of our experiments, the assumption we made also gives us the ability to easily transform an HTML document into an XML document. Indeed, we first have to label the HTML document using XCRFs. Then, from this labeled document, and knowing the DTD of the output XML document and the

fact that the text leaves will appear in the same order, we can build this output using a very simple XSLT stylesheet or an easily writable script.

An XCRF is defined with more than 7000 feature functions over the set of 66 labels. It should label HTML trees of average size 950 nodes (after tokenization). A naive application should fail for complexity reasons: $7000 \times 950 \times 66^3 \approx 2.10^{12}$. Therefore, we had to use a technique called "sequential partition composition" of XCRFs to perform this task.

This technique consists in breaking the complexity factor M^3, where M is the number of possible labels in the XCRF, by combining several XCRFs where M is considerably smaller than the original one. Let $\mathcal{Y}$ be the original set of labels. First, we need to build a partition $\mathcal{Y}_1, \ldots, \mathcal{Y}_k$ of $\mathcal{Y}$. The choice of this partition is guided by the DTD of the output XML documents. For instance, all the information about an award (name, location and year) are in a single part, whereas the title of the movie is alone, since it is not directly related to any other information. With this policy, we obtained a partition of $k = 30$ subsets of labels containing from 2 to 6 labels.

Then, we define k XCRFs over the k parts of the partition. To label a document, these k XCRFs are applied sequentially, following the order on the subsets of labels. At step i, the XCRF labels the HTML document with the labels in $\mathcal{Y}_i$. Since the XCRFs are applied sequentially, labeling at step i can use the labelings performed at the previous steps j, where $j < i$. Previous labeling are encoded in the HTML document. This allows for long distance dependencies between these labels. Once the HTML documents have been labeled by the k XCRFs, for some nodes, two or more labels might have been predicted by different XCRFs. In this case, a choice needs to be made. We decided to choose the label with the higher marginal probability.

The training of the k XCRFs is also performed sequentially. When training the i^{th} XCRF, the learning set is composed of the HTML documents enriched with the labels of the previous subsets of labels $\mathcal{Y}_j$, where $j < i$.

Using this method, the XCRFs were trained on a learning set of 692 labeled documents. We evaluated them on a testing set of 693 documents. We first evaluate the quality of the labeling performed by the XCRF. Then, we evaluate the performance of our method on the overall HTML to XML transformation task.

Evaluation of the xcrf for the labeling task. To evaluate the quality of the labeling performed by the XCRFs, we measure precision, recall and F1-measure for all the 66 labels in the task. In Table 3, we only show the micro-average of these results. Since these results might be biased by the great proportion of

Table 3. Evaluation of the labeling on the Structure Mapping task

Dataset	Average method	Rec.	Prec.	F1
html1	Micro	93.00	94.11	93.55
	Micro (without $\perp$)	94.96	77.09	85.10
html2	Micro	92.41	93.10	92.76
	Micro (without $\perp$)	94.10	69.95	80.24

nodes labeled with $\perp$, *i.e.* nodes which are not used in the final transformation, we also give the micro-average without this insignificant label.

First, it is worth noticing that the micro-averaged results over all the labels are very good on both versions of the dataset. Both the recall and the precision are above 92%, which proves the quality of the labeling. When averaging over all the labels except $\perp$ (*i.e.* only on the significant labels), two behaviours occur. On the one hand, the recall increases, meaning that most of the significant information in the HTML documents are correctly identified. On the other hand, there is a drop in precision. The explanation is that some useless information sometimes occur in the HTML documents in a structural context very similar to that of significant information and the XCRFs can not make the difference between them. This drop is slightly more important with the html2 dataset. This is not a surprise since this dataset contains more useless information than html1. This drop could be avoided by using feature functions which uses longer distance dependencies on the observation.

Evaluation of the html to xml transformation. Now, we evaluate the overall quality of the complete transformation from the HTML documents to their XML counterpart. This transformation is performed in two steps: first, the HTML documents are labeled with XCRFs; then a simple transformation using the output DTD and the labeled HTML documents produces the predicted XML documents. The evaluation is therefore a comparison between the predicted XML documents and the correct ones. To compare them, we compute the F1-measure according to two different criteria:

- **Paths:** for each text node in the predicted XML document, we build a couple made of the text content and the path to the text node. These couples are compared to those in the correct XML document.
- **Subtrees:** for each node in the XML document, we consider its subtrees, and the subtrees are compared to those in the correct XML document.

The main difference between these two criteria is that the first one does not take into account the order in which the text leaves appear. Overall, the second criteria is more precise and gives a better idea of the accuracy of the predicted XML documents.

Table 4 shows the average F1-measure over all the documents in the test dataset for the four criteria. First, we notice that, as when we evaluated the labeling, results show that our system performs very well on the html1 dataset. The results are very similar with both criteria. This is partly due to the text nodes appearing in the same order in both the HTML and the XML documents.

Table 4. Evaluation of the transformation on the Structure Mapping Task

Dataset	F1 on paths	F1 on subtrees
html1	91.81	89.76
html2	79.60	71.79

Moreover, since the construction of the XML documents is guided by the output DTD, our system always predicts XML documents conform to the DTD. Therefore, it is very unlikely that an undesired node is inserted. This explains the stability between the two measures.

With the html2 dataset, results are a bit lower, which is consistent with the results we observed when evaluating the labeling. Still, the F1-measure on the paths of the XML documents is good and close to 80. The drop of the F1-measure on the subtrees can be explained by the nature of the html2 dataset. Indeed, with this dataset, the XCRF sometimes failed to identify useless information. Therefore, these information are in the predicted XML documents. This results in several subtrees being incorrect, and a drop of the F1-measure, although the correct information are present in the predicted XML documents.

4 Conclusion and Future Work

We have introduced XCRF that are conditional models for labeling XML trees. We tackle the problem of structure mapping presented in this INEX challenge as a labeling problem for XML data. Experimental results show that XCRFs perform very well and confirm that XCRFs are well suited for such applications.

Our main line of future research involves extending our system to handle more sophisticated applications. Datasets we used in the challenge assume that document order is preserved during the transformation. To overcome this limitation, it can be necessary to extend the set of allowed editions operations. Another point is to integrate the transformation step in the XCRF training phase for a better parameter estimation. Other improvements include the combination of linear chain CRFs and XCRFs, and the introduction of ontologies, NLP outcomes, and domain knowledge to take advantage of both the structure and content of XML documents.

References

1. Doan, A., Halevy, A.Y.: Semantic integration research in the database community: A brief survey. AI magazine 26(1), 83–94 (2005)
2. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of ICML 2001, pp. 282–289 (2001)
3. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: Proceedings of HLT-NAACL, pp. 213–220 (2003)
4. McCallum, A., Li, W.: Early results for named entity recognition with conditional random fields. In: Proceedings of CoNLL'2003 (2003)
5. Pinto, D., McCallum, A., Wei, X., Croft, W.B.: Table extraction using conditional random fields. In: Proceedings of the 26th Annual International ACM SIGIR Conference, pp. 235–242 (2003)
6. Sarawagi, S., Cohen, W.W.: Semi-markov conditional random fields for information extraction. In: Proceedings of NIPS, pp. 1185–1192 (2004)
7. Sutton, C., Rohanimanesh, K., McCallum, A.: Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In: Proceedings of ICML 2004, pp. 783–790 (2004)

8. Sutton, C., McCallum, A.: An Introduction to Conditional Random Fields for Relational Learning. In: Introduction to Statistical Relational Learning. Lise getoor and ben taskar edn, MIT Press, Cambridge, MA (2006)
9. Riezler, S., King, T., Kaplan, R., Crouch, R., Maxwell, J., Johnson, M.: Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In: Proceedings of ACL 2002, pp. 271–278 (2002)
10. Clark, S., Curran, J.R.: Parsing the WSJ using CCG and log-linear models. In: Proceedings of ACL 2004, pp. 103–110 (2004)
11. Sutton, C.: Conditional probabilistic context-free grammars. Master's thesis, University of Massachusetts (2004)
12. Viola, P., Narasimhan, M.: Learning to extract information from semistructured text using a discriminative context free grammar. In: Proceedings of the ACM SIGIR, pp. 330–337 (2005)
13. Jousse, F., Gilleron, R., Tellier, I., Tommasi, M.: Conditional random fields for XML trees. In: ECML Workshop on Mining and Learning in Graphs (2006)
14. Doan, A., Domingos, P., Halevy, A.: Reconciling schemas of disparate data sources: A machine-learning approach. In: Proceedings of the ACM SIGMOD Conference, pp. 509–520 (2001)
15. Carme, J., Gilleron, R., Lemay, A., Niehren, J.: Interactive learning of node selecting tree transducers. Machine Learning 66(1), 33–67 (2007)
16. Cohen, W., Hurst, M., Jensen, L.: A Flexible Learning System for Wrapping Tables and Lists in HTML Documents. In: Web Document Analysis: Challenges and Opportunities, World Scientific, Singapore (2003)
17. Hsu, C.N., Dung, M.T.: Generating finite-state transducers for semi-structured data extraction from the web. Information Systems 23(8), 521–538 (1998)
18. Kushmerick, N.: Wrapper Induction for Information Extraction. PhD thesis, University of Washington (1997)
19. Muslea, I., Minton, S., Knoblock, C.: Active learning with strong and weak views: a case study on wrapper induction. In: IJCAI 2003, pp. 415–420 (2003)
20. Raeymaekers, S., Bruynooghe, M., Van den Bussche, J.: Learning (k,l)-contextual tree languages for information extraction. In: Proceedings of ECML 2005. Lecture Notes in Artificial Intelligence, vol. 3720, Springer, Heidelberg (2005)
21. Chidlovskii, B., Fuselier, J.: A probabilistic learning method for xml annotation of documents. In: Proceedings of IJCAI 2005 (2005)
22. Gallinari, P., Wisniewski, G., Denoyer, L., Maes, F.: Stochastic models for document restructuration. In: Proceedings of ECML Workshop On Relational Machine Learning (2005)
23. Wallach, H.: Efficient training of conditional random fields. Master's thesis, University of Edinburgh (2002)
24. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM Journal on Computing 16(5), 1190–1208 (1995)
25. Taskar, B., Abbeel, P., Koller, D.: Discriminative probabilistic models for relational data. In: Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02), pp. 485–492 (2002)
26. Denoyer, L., Gallinari, P.: Report on the XML mining track at INEX 2005 and INEX 2006. In: Proceedings of INEX 2006. LNCS, Springer, Heidelberg (2006)
27. Bex, G.J., Neven, F., Schwentick, T., Tuyls, K.: Inference of concise DTDs from XML data. In: Proceedings of 32nd Conference on Very Large databases - VLDB, pp. 115–126 (2006)

XML Structure Mapping

Application to the PASCAL/INEX 2006 XML Document Mining Track*

Francis Maes, Ludovic Denoyer, and Patrick Gallinari

LIP6 - University of Paris 6
firstname.lastname@lip6.fr

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 model 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.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 540–551, 2007.

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 Challenge[1]. 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 webpages to XML.
- *Legacy Document conversion:* conversion from flat text, loosely structured text, or any other layout oriented format (*e.g.* PDF) to XML.

```
<table>
  <tr>
    <td>Korben Dallas</td>
    <td>...</td>
    <td><a href="">Bruce Willis</a></td>
  </tr>
  <tr>
    <td>Leelo</td>
    <td>...</td>
    <td><a href="">Milla Jovovich</a></td>
  </tr>
</table>
```

```
<p>Casting: </p>
<ul>
  <li>Bruce Willis (Korben Dallas)</li>
  <li>Milla Jovovich (Leelo)</li>
</ul>
```

```
<cast>
  <character>
    <actor>Bruce Willis</actor>
    <characterName>Korben Dallas</characterName>
  </character>
  <character>
    <actor>Milla Jovovich</actor>
    <characterName></characterName>
  </character>
</cast>
```

Fig. 1. Example of XML heterogeneity. The same movie description extracted from three sources: two HTML styles and one XML general movie schema.

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 $d_{in} \in D_{in}$ into an XML document $d^*_{out} \in D_{out}$ 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 a specific XML schema. As training data, an user provides a set of pair $\left\{(d_{in}^i, d_{out}^{i*})\right\}_{i\in[1,N]}$ where N is the number of examples, d_{in} is an input document in D_{in}^i and d_{out}^{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. Such a model is parameterized by θ which is vector of real parameters. 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:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \Delta(f_\theta(d_{in}^i), d_{out}^{i*}) \tag{1}$$

Structure mapping models have to deal with two major difficulties. First, they have to support a large variety of transformations. 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

Because of the exponential number of valid output documents that can correspond to a given input document, the simple strategy, which consists in generating all possible output to select the best one, is unrealistic. One way to break the complexity of the task, is to decompose it into simpler sub-problems. This can be achieved by the incremental structure mapping (ISM) algorithm we propose.

First, we describe the ISM process in the case of a simple HTML to XML structure mapping example. We then detail our general structure mapping inference algorithm. Finally, we present the learning algorithm that allows to find the parameters θ which minimize the empirical risk on the training set.

3.1 Incremental Structure Mapping

ISM rely on the idea that the structure mapping can be realized by considering successively each leaf of the input document and working out its position in the output document. The process is decomposed into successive elementary steps. Each of these steps do two things: reading/analysing a part of the input document and adding corresponding nodes in the current output document. Figure 2

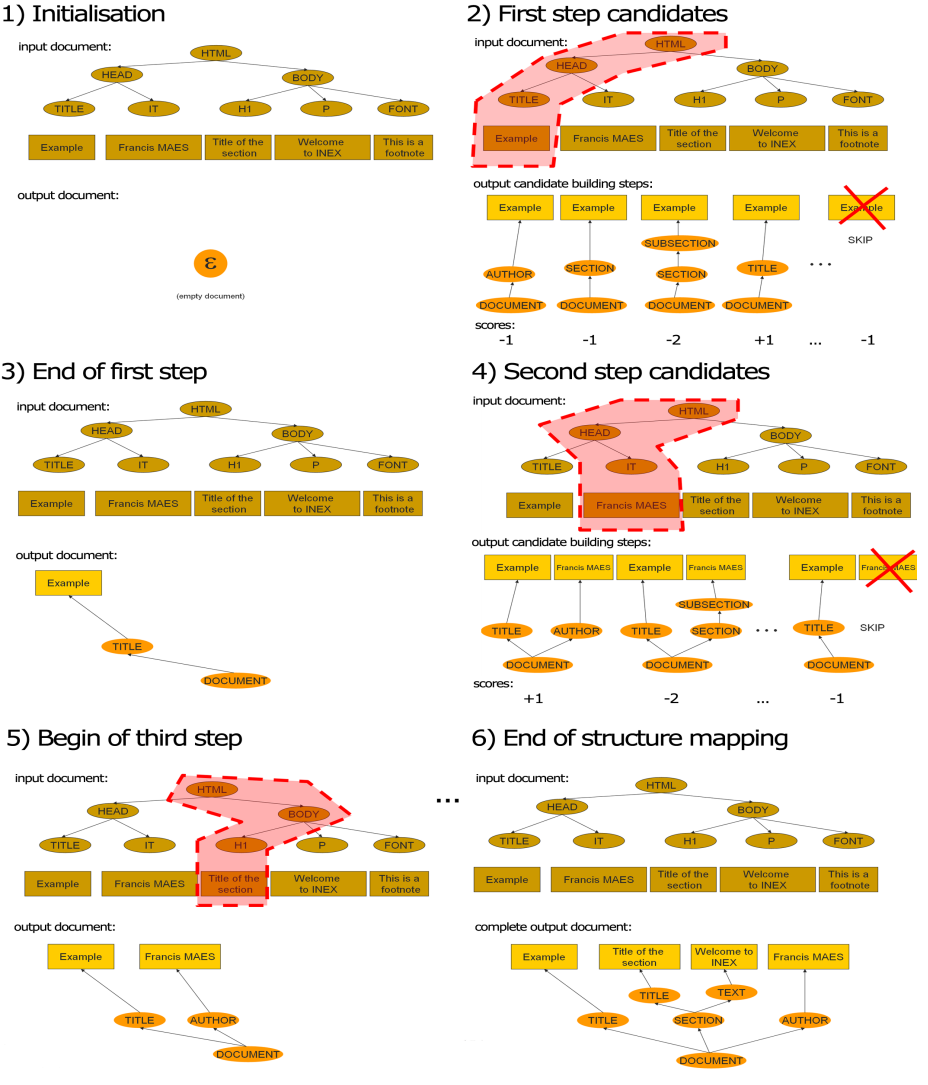


Fig. 2. 1)We initialize the process with the empty output tree (denoted ϵ). 2) First step: we focus on the first input leaf (with content "Example"). We enumerate building-step candidates. For each of this candidates there is an associated score. In this example the best building-step has score 1 and consists in adding "Example" as the output *Document Title*. 3) End of first step: the content "Example" has been added in the current output document. 4) Second step: we focus on the second input leaf, consider building-step candidates, where the best one is to add "Francis Maes" as the *Document Author*. 5) At the beginning of the third step, the *Author* has been added in the partial output tree. 6) Final state of the ISM process (after 5 steps) the current output tree is completed and can be returned to the user. We see in the final output tree that the *Section* nodes have been inserted between the two already existing nodes *Title* and *Author*.

Algorithm 1. Structure Mapping Inference

$d_{in} \in D_{in}$: the input document
$\phi : L_{in} \times B_{out} \rightarrow \Re^p$: the representation function
$\theta : \Re^p$: the learned parameters

1: $d_{out} \leftarrow \epsilon$
2: **for all** input leaf $n_i \in d_{in}$ **do**
3: candidates $\leftarrow$ computeOutputCandidatePaths(d_{in},d_{out}) $\bigcup$ `SKIP`
4: outputpath $\leftarrow$ argmax$_{c \in candidates} < \phi(n_i, c), \theta >$
5: **if** outputpath = `SKIP` **then**
6: **continue** (with next input leaf)
7: **else**
8: $d_{out} \leftarrow$ addNodeInOutputTree(d_{in}, outputpath, d_{out})
9: **end if**
10: **end for**
return d_{out}

gives an illustrative example where an HTML document is being converted into XML.

We initialize the process with the empty output tree denoted ϵ. After initialization we iterate ISM steps, one for each input leaf. Thus an ISM step starts with selecting a new input leaf. Given this input leaf and the current output tree, we compute a set of building-step candidates. Each of these candidates is a particular way to insert the current input leaf content into the current partial output tree. It defines both new node labels and new node positions in the existing tree. Each building-step candidate has also an associated score. This score quantifies the immediate interest of executing a particular building-step. The model then chooses the best scoring building-step candidate, *execute* it and starts with the next ISM step.

The key idea with ISM is that instead of considering all valid output documents we consider only the valid building-steps at each step of the process. This way, the complexity of structure mapping only depends on the number of ISM steps and on the average number of candidates per step.

3.2 Inference Algorithm

The behavior of ISM is directly related to the scoring function of building-step candidates. This scores are only available for training document pairs. In the general case, these scores will be estimated using a linear predictor. Learning the ISM model from document pair examples is reduced to learning the ISM score predictor. Learning is described below, see 3.3. For the moment, let us consider that the model has been learned. Algorithm 1. shows the general ISM inference procedure.

The ISM inference algorithm has three parameters: the input document $d_{in} \in D_{in}$ and two parameters which defines the score predictor. In order to predict such a score, we first describe a (input leaf $\in L_{in}$, building-step candidate $\in$

B_{out}) pair using the ϕ function. This function produces a feature vector in $\Re^p$. Examples of such features are given in table 1. We produce features by combining state features with action features. To describe a (state,action) pair, we make the cross product between all state events and all action events. This way, ϕ produces sparse joint (state,action) representations where the number of distinct features p ranges usually from 10^3 to 10^6.

Table 1. Some examples of features which jointly describe a (input node, building-step) pair using the ϕ function. These features are usually binary (valued in $\{0, 1\}$) but general real valued features are also possible (*e.g.* the percentage of upper case words). The features are generated in a data-driven way: a feature is considered only once it is observed in the learning data. Depending on the corpora, the features vectors have from 10^3 to 10^6 distinct components.

Description	Value
We are processing the first input node and the building-step has labels *DOCUMENT TITLE*.	1
The input node has label *IT* and the building-step has labels *DOCUMENT AUTHOR*.	1
The current input leaf has 3 parents and the building-step inserts the node between a *TITLE* and a *SECTION*.	0
The last word of the current input node is "footnote" and the building-step is `SKIP`.	1
The last word of the input node is punctuation symbol and the building-step has labels *DOCUMENT SECTION TEXT*.	0
...	...

Given a description in $\Re^p$, the score is estimated by a dot product between the description vector and the parameters vector ($\theta \in \Re^p$). The dot product between two vectors is denoted $< ., . >$.

ISM inference is performed by iterating over input leaves. For each of this nodes, we first enumerate the set of building-step candidates (line 3). In order to include the possibility to skip some input leaves we also consider the `SKIP` building-step. `SKIP`ing a node means that this node will not be included in the output tree. In line 4, we estimate the scores of all candidates using our linear predictor. The best estimated building-step candidate is chosen. If the best building-step is `SKIP` we can continue with the next input leaf (line 6). In any other case, the building-step is *executed* with the *addNodeInOutputTree* function (line 8). This produces a new partial output tree d_{out}. Once all input leaves have been processed, the ISM is fulfilled and we return the current output tree d_{out} to the user.

3.3 Learning Algorithm

The ISM learning procedure is described in algorithm 2. It aims at finding the parameters θ that leads to a good structure mapping inference procedure.

Algorithm 2. Structure Mapping Learning

$S = \left\{(d_{in}^{i}, d_{out}^{i*})\right\}_{i\in[1,N]}$: Training Set
$\phi : L_{in} \times B_{out} \rightarrow \Re^{p}$: the representation function

1: $\theta \leftarrow \mathbf{0}$
2: **repeat**
3: $d_{in}, d_{out}^{*} \leftarrow$ pickTrainingPair(S)
4: $d_{out} \leftarrow$ structureMappingInference(d_{in}, ϕ, θ)
5: loss $\leftarrow \Delta(d_{out}, d_{out}^{*})$
6: **for all** $n_i \in d_{in}$ **do**
7: $\theta \leftarrow$ applyGradientCorrection(θ, n_i, loss)
8: **end for**
9: **until** convergence of θ
return θ

The algorithm has two parameters: the training set S of document pairs and the representation function ϕ described in previous section. The algorithm returns the learned parameters θ that can be used in inference.

ISM learning is done by iteratively evaluating and improving the parameter vector θ. In each iteration, we pick randomly a new training document pair (line 3). We then evaluate the current parameters θ by calling the inference procedure (line 4) and computing the resulting loss (line 5). We can now improve the parameters θ by applying a little correction for each ISM step that was performed during inference (line 6-8).

The details of the learning procedure are omitted here for the sake of clarity. However algorithms 1. and 2. relies on established machine learning techniques. Briefly, the ISM procedure can be modelled as a Markov Decision Process (MDP) [2]. MDPs provides 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. The fields of Reinforcement Learning [3] and Approximate Dynamic Programming [4] provides several learning algorithms for solving MDPs. Our learning procedure can be seen as a particular case of the Sarsa(0) algorithm. We differ the interested reader to [5] for more details.

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 is the INEX IEEE corpus which is composed of 12017 scientific articles in XML format. Each document comes from a journal (18 different 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 more than 13000 movie descriptions available in three versions: two different XHTML 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 displacements.

In order to compare our model, we also made experiments on the Shakespeare corpora[2]. As in [6], we have randomly selected 60 Shakespearean scenes from the collection. These scenes have an average length of 85 leaf nodes and 20 internal nodes over 7 distinct tags. As a baseline, we implemented the model of [6] which is based on probabilistic context free grammars and maximum entropy classifiers. Due to its complexity (roughly cubic in the number of input leafs wheras ours is linear) this model cannot be applied to the others corpora.

Each corpus is split in two parts: 50% for training and 50% for testing. The table 2 summarizes the properties of our corpora.

Table 2. Description of the corpora used in our experiments. From left to right: the name of the corpus, the task, the number of documents, the mean number of internal nodes per document, the mean number of leaves per document, the number of disctinct tags.

Corpus	Tasks	Corpus size	Internal Nodes	Leaves	Labels
INEX IEEE	Flat → XML	12,017 docs	≈ 200	≈ 500	139
Movie 1	XHTML → XML	13,045 docs	≈ 78	≈ 31	16
Movie 2	XHTML → XML	13,038 docs	≈ 49	≈ 40	19
Shakespeare	Flat → XML	60 docs	≈ 20	≈ 85	7

4.2 Loss Function and Evaluation Measures

In order to evaluate the quality of structure mapping we have used two measures: $F_{content}$ and $F_{structure}$. The first measure reflects the quality of document leaves labelling. The second measure reflects the quality of the internal tree structure. Both measure are the mean of a F_1 score computed for all (predicted document, correct document) pairs. For $F_{content}$ we compute the F_1 score between leaves labels. This first measure is similar to the *Word Error Ratio* used in natural language. $F_{structure}$ is based on the F_1 score between all subtrees. This F_1 score between two trees is computed in the following way:

1. Build the set of all subtrees of the two trees. There is one sub-tree 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 leaves), and the same children trees (for internal nodes).
3. Compute the F1 score: $F1 = \frac{2*Recall*Precision}{Recall+Precision}$.

[2] http://metalab.unc.edu/bosak/xml/eg/shaks200.zip

This corresponds to a common measure in the natural language parsing field (under the name of *F1 parsing score*). Note that this measure decreases quickly with only a few errors. For example if there is only one labelling error in a leaf, the $F_{structure}$ measure typically equals to $\approx 80\%$.

4.3 Results

The loss-function $\Delta(d_{out}, d^*_{out})$ used for training the model is based on the $F_{structure}$ measure:

$$\Delta(d_{out}, d^*_{out}) = 1 - F_{structure}(d_{out}, d^*_{out})$$

Figure 4.3 shows the results obtained for the different experiments. All $F_{content}$ scores are greater than 75 % while the more difficult $F_{structure}$ is still greater than ≈ 60 %. These scores have to be contrasted with the intrinsic difficulty of the structure mapping tasks. For example for INEX, the only hints for predicted between more than hundred labels come from the textual content of the input document.

Corpus	Method	$F_{content}$	$F_{structure}$	Learning time	Testing time
INEX IEEE	ISM	75.8 %	67.5 %	$\approx$ 2 days	$\approx$ 2 s / doc
Movie 1	ISM	80.3 %	65.8 %	$\approx$ 17 min	$\approx$ 0.08 s / doc
Movie 2	ISM	75.4 %	57.0 %	$\approx$ 19 min	$\approx$ 0.07 s / doc
Shakespeare	ISM	89.4 %	84.4 %	$\approx$ 20 min	$\approx$ 0.02 s / doc
Shakespeare	PCFG+ME	98.7 %	97.9 %	$\approx$ 2 min	$\approx$ 1 min / doc

Fig. 3. Structure mapping results on the tree corpora. Two measure are used: $F_{content}$ and $F_{structure}$. Approximate learning and testing time are indicated - the experiments were performed on a standard 3.2Ghz Computer.

These results also show how fast ISM is at testing time. Most documents are processed in less than one second. This mean that ISM could be used with large scale corpora containing thousands or millions documents.

The Shakespeare database shows a comparison between ISM and our baseline called PCFG+ME (see part 4.1). ISM scores are less good than the baseline scores. A first explanation for this phenomenon is that PCFG+ME does a global optimization using Dynamic Programming. This has to be contrasted with ISM which performs a greedy search of the output tree. We also suspect ISM to suffer from over-fitting since there are few documents and many distinct features (see 1).

On the other side, ISM is approximatively thousand times faster in inference than the baseline. Moreover, due to its complexity the baseline cannot be applied to our real world corpora. We believe that in the context of heterogeneous information retrieval, fast inference time is much more important than perfect structure mapping.

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 [7]. [8] 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 [9] 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 ([10], [11]) 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 [6]. 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. Howard, R.A.: Dynamic Programming and Markov Processes. Technology Press-Wiley, Cambridge, Massachusetts (1960)
3. Sutton, R., Barto, A.: Reinforcement learning: an introduction. MIT Press, Cambridge (1998)
4. Si, J., Barto, A.G., P.W.B., II, D.W. : Handbook of Learning and Approximate Dynamic Programming. Wiley&Sons, INC., Publications, New York (2004)
5. Sutton, R.S.: Generalization in reinforcement learning: Successful examples using sparse coarse coding. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems, vol. 8, pp. 1038–1044. The MIT Press, Cambridge, MA (1996)
6. Chidlovskii, B., Fuselier, J.: A probabilistic learning method for xml annotation of documents. In: IJCAI, pp. 1016–1021 (2005)
7. Doan, A., Halevy, A.Y.: Semantic integration research in the database community: A brief survey. AI Magazine, Special Issue on Semantic Integration (2005)
8. Doan, A., Domingos, P., Halevy, A.: Learning to match the schemas of data sources: A multistrategy approach. Maching Learning 50(3), 279–301 (2003)
9. Califf, M.E., Mooney, R.J.: Bottom-up relational learning of pattern matching rules for information extraction. J. Mach. Learn. Res. 4, 177–210 (2003)
10. Young-Lai, M., Tompa, F.W.: Stochastic grammatical inference of text database structure. Mach. Learn. 40(2), 111–137 (2000)
11. Chidlovskii, B., Fuselier, J.: Supervised learning for the legacy document conversion. In: DocEng '04, pp. 220–228. ACM Press, New York (2004)

Author Index