



**INEX 2009
Workshop
Pre-proceedings**

Shlomo Geva, Jaap Kamps, Andrew Trotman
(editors)

December 6–10, 2009
Woodlands of Marburg, Ipswich, Queensland,
Australia
<http://www.inex.otago.ac.nz/>

Copyright ©2009 remains with the author/owner(s).

The unreviewed pre-proceedings are collections of work submitted before the December workshops. They are not peer reviewed, are not quality controlled, and contain known errors in content and editing. The proceedings, published after the Workshop, is the authoritative reference for the work done at INEX. Published by: IR Publications, Amsterdam. ISBN 978-90-814485-2-9.

Preface

Welcome to the 8th workshop of the Initiative for the Evaluation of XML Retrieval (INEX)!

Now, in its eighth year, INEX is an established evaluation forum for XML information retrieval (IR), with over 100 organizations worldwide registered and over 50 groups participating actively in at least one of the tracks. INEX aims to provide an infrastructure, in the form of a large structured test collection and appropriate scoring methods, for the evaluation of focused retrieval 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.

INEX 2009 was an exciting year for INEX in which a new collection was introduced that is again based on a the Wikipedia but is over 4 times larger, with longer articles and additional semantic annotations. In total eight research tracks were included, which studied different aspects of focused information access:

Ad hoc Track The main track of INEX 2009 is investigating the effectiveness of XML-IR and Passage Retrieval for four ad hoc retrieval tasks (Thorough, Focused, Relevant in Context, Best in Context).

Book Track Investigating information access to, and IR techniques for searching full texts of digitized books.

Efficiency Track Investigating both the effectiveness and efficiency of XML ranked retrieval approaches on real data and real queries.

Entity Ranking Track Investigating entity retrieval rather than text retrieval: 1) Entity Ranking, 2) Entity List Completion.

Interactive Track Investigating the behavior of users when interacting with XML documents, as well as develop retrieval approaches which are effective in user-based environments.

Question Answering Track Investigating technology for accessing structured documents that can be used to address real-world focused information needs formulated as natural language questions.

Link the Wiki Track Investigating link discovery between Wikipedia documents, both at the file level and at the element level.

XML Mining Track Investigating structured document mining, especially the classification and clustering of structured documents.

The aim of the INEX 2009 workshop is to bring together researchers in the field of XML IR who participated in the INEX 2009 campaign. During the past year participating organizations contributed to the building of a large-scale XML test collection by creating topics, performing retrieval runs and providing relevance assessments. The workshop concludes the results of this large-scale effort, summarizes and addresses encountered issues and devises a work plan for the future evaluation of XML retrieval systems.

All INEX tracks start from having available suitable text collections. We gratefully acknowledge the data made available by: Amazon (Interactive Track), New Zealand Ministry for Culture and Heritage (*Te Ara*, Link-the-Wiki Track), Microsoft (Book Track), Wikipedia, and to Ralf Schenkel of the Max-Planck Institute for the conversion of the Wikipedia.

INEX has outgrown its previous home at *Schloss Dagstuhl* and is held in Brisbane, Australia. Thanks to Richi Nayak and the QUT team for preserving the unique atmosphere of INEX—a setting where informal interaction and discussion occurs naturally and frequently—in the unique location of the Woodlands of Marburg. Thanks to HCSNet, the Australian Research Council’s Research Network in Human Communication Science, for sponsoring the invited talks. Finally, INEX is run for, but especially by, the participants. It is a result of tracks and tasks suggested by participants, topics created by participants, systems built by participants, and relevance judgments provided by participants. So the main thank you goes each of these individuals!

December 2009

Shlomo Geva
Jaap Kamps
Andrew Trotman

Organization

Steering Committee

Charlie Clarke (University of Waterloo)
Norbert Fuhr (University of Duisburg-Essen)
Shlomo Geva (Queensland University of Technology)
Jaap Kamps (University of Amsterdam)
Mounia Lalmas (Queen Mary, University of London)
Stephen Robertson (Microsoft Research Cambridge)
Andrew Trotman (University of Otago)
Ellen Voorhees (NIST)

Chairs

Shlomo Geva (Queensland University of Technology)
Jaap Kamps (University of Amsterdam)
Andrew Trotman (University of Otago)

Track Organizers

Ad Hoc

Shlomo Geva (Queensland University of Technology)
Jaap Kamps (University of Amsterdam)
Miro Lethonen (University of Helsinki)
James A. Thom (RMIT)
Andrew Trotman (University of Otago)
Ralf Schenkel (Max-Planck-Institut für Informatik)

Book

Antoine Doucet (University of Caen)
Gabiella Kazai (Microsoft Research Limited)
Marijn Koolen (University of Amsterdam)
Monica Landoni (University of Strathclyde)

Efficiency

Ralf Schenkel (Max-Planck-Institut für Informatik)
Martin Theobald (Max-Planck-Institut für Informatik)

Entity Ranking

Gianluca Demartini (L3S)
Tereza Iofciu (L3S)
Arjen de Vries (CWI)

Interactive (iTrack)

Nisa Fachry (University of Amsterdam)
Norbert Fuhr (University of Duisburg-Essen)
Ragnar Nordlie (Oslo University College)
Nils Pharo (Oslo University College)

Question Answering (QA)

Veronique Moriceau (LIMSI-CNRS, University Paris-Sud 11)
Eric SanJuan (University of Avignon)
Xavier Tannier (LIMSI-CNRS, University Paris-Sud 11)

Link-the-Wiki

Shlomo Geva (Queensland University of Technology)
Andrew Trotman (University of Otago)

XML-Mining

Ludovic Denoyer (University Paris 6)
Chris De Vries (Queensland University of Technology)
Patrick Gallinari (University Paris 6)
Sangeetha Kutty (Queensland University of Technology)
Richi Nayak (Queensland University of Technology)

Table of Contents

Front matter.

Preface	iii
Organization	v
Table of Contents	vii

Invited Talks.

Is There Something Quantum-like about the Human Mental Lexicon? <i>(invited talk)</i>	13
<i>Peter Bruza</i>	
Supporting for Real-World Tasks: Producing Summaries of Scientific Articles Tailored to the Citation Context <i>(invited talk)</i>	14
<i>Cecile Paris</i>	
Semantic Document Processing using Wikipedia as a Knowledge Base <i>(invited talk)</i>	15
<i>Ian Witten</i>	

Ad hoc Track.

Overview of the INEX 2009 Ad Hoc Track	16
<i>Shlomo Geva, Jaap Kamps, Miro Lehtonen, Ralf Schenkel, James A. Thom, Andrew Trotman</i>	
LIP6 at Inex'09 : OWPC for adhoc track	51
<i>David Buffoni, Nicolas Usunier, Patrick Gallinari</i>	
Peking University at INEX 2009: Ad Hoc Track.....	53
<i>Ning Gao, Zhi-Hong Deng, Yong-Qing Xiang, Hang Yu</i>	
UJM at INEX 2009 Ad Hoc track	60
<i>Mathias Géry, Christine Largeton</i>	
A Method of Generating Answer XML Fragment from Ranked Results... ..	67
<i>Atsushi Keyaki, Jun Miyazaki, Kenji Hatano</i>	
ENSM-SE at INEX 2009 : scoring with proximity and semantic tag information	75
<i>Annabelle Mercier, Amelie Imafouo, Michel Beigbeder</i>	

Use of Language Model, Phrases and Wikipedia Forward Links for INEX 2009	85
<i>Philippe Mulhem, Jean-Pierre Chevallet</i>	
Indian Statistical Institute at INEX 2009 Adhoc Focused Task.....	94
<i>Sukomal Pal, Mandar Mitra, Debasis Ganguly</i>	
Universities of Avignon and Lyon3 at INEX 2009	99
<i>Eric SanJuan, Fidelia Ibekwe-SanJuan</i>	
How well does Best in Context reflect ad hoc XML retrieval? Reprise	106
<i>James A. Thom</i>	
Exploiting Semantic Tags in XML Retrieval	108
<i>Qiuyue Wang, Qiushi Li, Shan Wang, Xiaoyong Du, Yina Geng, Zuoyan Qin</i>	
Book Track.	
Overview of the INEX 2009 Book Track	120
<i>Gabriella Kazai, Antoine Doucet, Marijn Koolen, Monica Landoni</i>	
A Methodology for Producing Improved Focused Elements	133
<i>Carolyn Crouch, Donald Crouch, Dinesh Bhirud, Pavan Poluri, Chaitanya Polumetla, Varun Sudhaker</i>	
Resurgence for the Book Structure Extraction Competition	136
<i>Emmanuel Giguet, Alexandre Baudrillart, Nadine Lucas</i>	
XRCE Participation to the Book Structure Task	143
<i>Hervé Déjean, Jean-Luc Meunier</i>	
Ranking and Fusion Approaches for XML Book Retrieval	153
<i>Ray Larson</i>	
Twente University at INEX 2009: Ad-hoc Track	163
<i>Rongmei Li</i>	
OUC's participation in the 2009 INEX Book Track	170
<i>Michael Preminger, Ragnar Nordlie, Nils Pharo</i>	
Efficiency Track.	
Overview of the INEX 2009 Efficiency Track	175
<i>Ralf Schenkel, Martin Theobald</i>	
Index Tuning for Efficient Proximity-Enhanced Query Processing	188
<i>Andreas Broschart, Ralf Schenkel</i>	

INEX Efficiency Track meets XQuery Full Text in BaseX	192
<i>Sebastian Gath, Christian Grün, Alexander Holupirek, Marc H. Scholl</i>	
TopX 2.0 at the INEX 2009 Ad-Hoc and Efficiency Tracks	198
<i>Martin Theobald, Ablimit Aji, Ralf Schenkel</i>	
Fast and Effective Focused Retrieval	209
<i>Andrew Trotman, Xiang-Fei Jia, Shlomo Geva</i>	
Experiments regarding the Efficiency Track and the Ad Hoc Track at INEX'09: Searching in Large-Scale Collections	222
<i>Judith Winter, Gerold Kühne</i>	

Entity Ranking Track.

Overview of the INEX 2009 Entity Ranking Track	233
<i>Gianluca Demartini, Tereza Iofciu, Arjen de Vries</i>	
The University of Amsterdam (ISLA) at INEX 2009	238
<i>Jiyin He, Krisztian Balog, Marc Bron, Wouter Weerkamp, Maarten de Rijke</i>	
University of Waterloo at INEX 2009: Ad Hoc, Book, Entity Ranking, and Link-the-Wiki Tracks	249
<i>Kelly Y. Itakura, Charles L. A. Clarke</i>	
University of Amsterdam at INEX 2009: Ad hoc, Book and Entity Ranking Tracks	260
<i>Marijn Koolen, Rianne Kaptein, Jaap Kamps</i>	
A Recursive approach to Entity Ranking and List Completion using Entity Determining Terms, Qualifiers and Prominent n-grams	273
<i>Madhu Ramanathan, Srikanth Rajagopal, Venkatesh Karthik, Meenakshi Sundaram Murugesan, Saswati Mukherjee</i>	

Interactive Track.

Overview of the INEX 2009 Interactive Track	282
<i>Nils Pharo, Ragnar Nordlie</i>	

Link the Wiki Track.

Overview of INEX 2009 Link the Wiki Track	290
<i>Darren Huang, Andrew Trotman, Shlomo Geva</i>	
Link Prediction for Interlinked Documents by using Probability Measure Self Organizing Maps for Structured Domains	302
<i>Markus Hagenbuchner, Milly Kc, Rowena Chau, Ah Chung Tsoi, Vin- cent Lee</i>	

Discovering Links Using Semantic Relatedness	314
<i>Johannes Hoffart, Daniel Bär, Torsten Zesch, Iryna Gurevych</i>	

Link Discovery in the Wikipedia	326
<i>Eric Tang, Shlomo Geva, Andrew Trotman</i>	

QA Track.

QA@INEX 2009: A common task for QA, focused IR and automatic summarization systems	334
<i>Véronique Moriceau, Eric SanJuan, Xavier Tannier</i>	

XML Mining Track.

Report on the XML Mining Classification Track at INEX 2009	339
<i>Ludovic Denoyer, Patrick Gallinari</i>	

Report on the XML Mining Track's Clustering Task at INEX 2009	343
<i>Richi Nayak, Chris De Vries, Sangeetha Kutty, Shlomo Geva</i>	

Exploiting Index Pruning Methods for Clustering XML Collections	349
<i>Ismail Sengor Altıngövdü, Duygu Atilgan, Özgür Ulusoy</i>	

Multi-label Wikipedia classification with textual and graph features	355
<i>Boris Chidlovskii</i>	

Clustering with Random Indexing K-tree and XML Structure	365
<i>Chris De Vries, Shlomo Geva, Lance De Vine</i>	

Supervised Encoding of Graph-of-Graphs for Classification and Regression Problems	367
<i>Markus Hagenbuchner, ShuJia Zhang, Franco Scarselli, Ah Chung Tsoi</i>	

Clustering XML documents using Multi-feature Model	379
<i>Sangeetha Kutty, Richi Nayak, Yuefeng Li</i>	

UJM at INEX 2009 XML Mining Track	381
<i>Christine Largeton, Christophe Moulin, Mathias Géry</i>	

BUAP: Performance of K-Star at the INEX'09 Clustering Task	391
<i>David Pinto, Mireya Tovar, Darnes Vilarino, Beatriz Beltran, Héctor Jiménez Salazar</i>	

Link-based text classification using Bayesian networks	398
<i>Alfonso E. Romero, Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete, Andrés R. Masegosa</i>	

Extended VSM for XML Document Classification using Frequent Subtrees	408
<i>Jianwu Yang, Songlin Wang</i>	

Back matter.

Author Index 416

Is There Something Quantum-like about the Human Mental Lexicon?

Peter Bruza

Faculty of Science and Technology
Queensland University of Technology, Australia
p.bruza@qut.edu.au

Abstract. This talk proceeds from the premise that IR should engage in a more substantial dialogue with cognitive science. After all, how users decide relevance, or how they chose terms to modify a query are processes rooted in human cognition. Recently, there has been a growing literature applying quantum theory (QT) to model cognitive phenomena ranging from human memory to decision making. Two aspects will be highlighted. The first will show how concept combinations can be modelled in a way analogous to quantum entangled twin-state photons. Details will be presented of cognitive experiments to test for the presence of “entanglement” in cognition via an analysis of bi-ambiguous concept combinations. The second aspect of the talk will show how quantum inference effects currently being used to fit models of human decision making may be applied to model interference between different dimensions of relevance.

The underlying theme behind this talk is QT can potentially provide the theoretical basis of new genre of information processing models more aligned with human cognition.

Acknowledgments This research is supported in part by the Australian Research Council Discovery grant DP0773341.

Supporting for Real-World Tasks: Producing Summaries of Scientific Articles Tailored to the Citation Context

Cécile Paris

Information and Communication Technology (ICT) Centre
CSIRO, Australia
`Cecile.Paris@csiro.au`

Abstract. The amount of scientific material available electronically is forever increasing. This makes reading the published literature, whether to stay up-to-date on a topic or to get up to speed on a new topic, a difficult task. Yet, this is an activity in which all researchers must be engaged on a regular basis. Based on a user requirements analysis, we developed a new research tool, called the Citation-Sensitive In-Browser Summariser (CSIBS), which supports researchers in this browsing task. CSIBS enables readers to obtain information about a citation at the point at which they encounter it. This information is aimed at enabling the reader to determine whether or not to invest the time in exploring the cited article further, thus alleviating information overload. CSIBS builds a summary of the cited document, bringing together meta-data about the document and a citation-sensitive preview that exploits the citation context to retrieve the sentences from the cited document that are relevant at this point. In this talk, I will briefly present our user requirements analysis, then describe the system and, finally, discuss the observations from an initial pilot study. We found that CSIBS facilitates the relevancy judgment task, by increasing the users' self-reported confidence in making such judgements.

Semantic Document Processing using Wikipedia as a Knowledge Base

Ian H. Witten

Department of Computer Science
University of Waikato, New Zealand
`ihw@cs.waikato.ac.nz`

Abstract. Wikipedia is a goldmine of information; not just for its many readers, but also for the growing community of researchers who recognize it as a resource of exceptional scale and utility. It represents a vast investment of manual effort and judgment: a huge, constantly evolving tapestry of concepts and relations that is being applied to a host of tasks. This talk will introduce the process of "wikification"; that is, automatically and judiciously augmenting a plain-text document with pertinent hyperlinks to Wikipedia articles – as though the document were itself a Wikipedia article. This amounts to a new semantic representation of text in terms of the salient concepts it mentions, where "concept" is equated to "Wikipedia article." Wikification is a useful process in itself, adding value to plain text documents. More importantly, it supports new methods of document processing.

I first describe how Wikipedia can be used to determine semantic relatedness, and then introduce a new, high-performance method of wikification that exploits Wikipedia's 60 M internal hyperlinks for relational information and their anchor texts as lexical information, using simple machine learning. I go on to discuss applications to knowledge-based information retrieval, topic indexing, document tagging, and document clustering. Some of these perform at human levels. For example, on CiteULike data, automatically extracted tags are competitive with tag sets assigned by the best human taggers, according to a measure of consistency with other human taggers.

Although this work is based on English it involves no syntactic parsing, and the techniques are largely language independent. The talk will include live demos.

Overview of the INEX 2009 Ad Hoc Track

Shlomo Geva¹, Jaap Kamps², Miro Lethonen³,
Ralf Schenkel⁴, James A. Thom⁵, and Andrew Trotman⁶

¹ Queensland University of Technology, Brisbane, Australia
`s.geva@qut.edu.au`

² University of Amsterdam, Amsterdam, The Netherlands
`kamps@uva.nl`

³ University of Helsinki, Helsinki, Finland
`miro.lehtonen@helsinki.fi`

⁴ Max-Planck-Institut für Informatik, Saarbrücken, Germany
`schenkel@mpi-sb.mpg.de`

⁵ RMIT University, Melbourne, Australia
`james.thom@rmit.edu.au`

⁶ University of Otago, Dunedin, New Zealand
`andrew@cs.otago.ac.nz`

Abstract. This paper gives an overview of the INEX 2009 Ad Hoc Track. The main goals of the Ad Hoc Track were three-fold. The first goal was to investigate the impact of the collection scale and markup, by using a new collection that is again based on a the Wikipedia but is over 4 times larger, with longer articles and additional semantic annotations. For this reason the Ad Hoc track tasks stayed unchanged, and the Thorough Task of INEX 2002–2006 returns. The second goal was to study the impact of more verbose queries on retrieval effectiveness, by using the available markup as structural constraints—now using both the Wikipedia’s layout-based markup, as well as the enriched semantic markup—and by the use of phrases. The third goal was to compare different result granularities by allowing systems to retrieve XML elements, ranges of XML elements, or arbitrary passages of text. This investigates the value of the internal document structure (as provided by the XML mark-up) for retrieving relevant information. The INEX 2009 Ad Hoc Track featured four tasks: For the *Thorough Task* a ranked-list of results (elements or passages) by estimated relevance was needed. For the *Focused Task* a ranked-list of non-overlapping results (elements or passages) was needed. For the *Relevant in Context Task* non-overlapping results (elements or passages) were returned grouped by the article from which they came. For the *Best in Context Task* a single starting point (element start tag or passage start) for each article was needed. We discuss the setup of the track, the results for the four tasks, and examine the relative effectiveness of element and passage retrieval. This is examined in the context of content only (CO, or Keyword) search as well as content and structure (CAS, or structured) search. In addition, we look at the effectiveness of systems using a reference run with a solid article ranking, and of systems using the phrase query. Finally, we look at the ability of focused retrieval techniques to rank articles.

1 Introduction

This paper gives an overview of the INEX 2009 Ad Hoc Track. There are three main research questions underlying the Ad Hoc Track. The first main research question is the impact of the new collection—four times the size, with longer articles, and additional semantic markup—on focused retrieval. That is, what is the impact of collection size? What is the impact of document length, and hence the complexity of the XML structure in the DOM tree? The second main research question is the impact of more verbose queries—using either the XML structure, or using multi-word phrases. That is, what is the impact of semantic annotation on both the submitted queries, and their retrieval effectiveness? What is the impact of explicitly annotated multi-word phrases? The third main research question is that of the value of the internal document structure (mark-up) for retrieving relevant information. That is, does the document structure help to identify where the relevant information is within a document?

To study the value of the document structure through direct comparison of element and passage retrieval approaches, the retrieval results were liberalized to arbitrary passages. Every XML element is, of course, also a passage of text. At INEX 2008, a simple passage retrieval format was introduced using file-offset-length (FOL) triplets, that allow for standard passage retrieval systems to work on content-only versions of the collection. That is, the offset and length are calculated over the text of the article, ignoring all mark-up. The evaluation measures are based directly on the highlighted passages, or arbitrary best-entry points, as identified by the assessors. As a result it is possible to fairly compare systems retrieving elements, ranges of elements, or arbitrary passages. These changes address earlier requests to liberalize the retrieval format to ranges of elements [1] and to arbitrary passages of text [11].

The INEX 2009 Ad Hoc Track featured four tasks:

1. For the *Thorough Task* a ranked-list of results (elements or passages) by estimated relevance must be returned. It is evaluated by mean average interpolated precision relative to the highlighted (or believed relevant) text retrieved.
2. For the *Focused Task* a ranked-list of non-overlapping results (elements or passages) must be returned. It is evaluated at early precision relative to the highlighted (or believed relevant) text retrieved.
3. For the *Relevant in Context Task* non-overlapping results (elements or passages) must be returned, these are grouped by document. It is evaluated by mean average generalized precision where the generalized score per article is based on the retrieved highlighted text.
4. For the *Best in Context Task* a single starting point (element’s starting tag or passage offset) per article must be returned. It is also evaluated by mean average generalized precision but with the generalized score (per article) based on the distance to the assessor’s best-entry point.

We discuss the results for the four tasks, giving results for the top 10 participating groups and discussing their best scoring approaches in detail. We also examine

the relative effectiveness of element and passage runs, and with content only (CO) queries and content and structure (CAS) queries.

The rest of the paper is organized as follows. First, Section 2 describes the INEX 2009 ad hoc retrieval tasks and measures. Section 3 details the collection, topics, and assessments of the INEX 2009 Ad Hoc Track. In Section 4, we report the results for the Thorough Task (Section 4.2); the Focused Task (Section 4.3); the Relevant in Context Task (Section 4.4); and the Best in Context Task (Section 4.5). Section 5 details particular types of runs (such as element versus passage, using phrases or using the reference run), and on particular subsets of the topics (such as topics with a non-trivial CAS query). Section 6 looks at the article retrieval aspects of the submissions, treating any article with highlighted text as relevant. Finally, in Section 7, we discuss our findings and draw some conclusions.

2 Ad Hoc Retrieval Track

In this section, we briefly summarize the ad hoc retrieval tasks and the submission format (especially how elements and passages are identified). We also summarize the measures used for evaluation.

2.1 Tasks

Thorough Task The core system’s task underlying most XML retrieval strategies is the ability to estimate the relevance of potentially retrievable elements or passages in the collection. Hence, the Thorough Task simply asks systems to return elements or passages ranked by their relevance to the topic of request. Since the retrieved results 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.

Focused Task The scenario underlying the Focused Task is the return, to the user, of a ranked list of elements or passages for their topic of request. The Focused Task requires systems to find the most focused results that satisfy an information need, without returning “overlapping” elements (shorter is preferred in the case of equally relevant elements). Since ancestors elements and longer passages are always relevant (to a greater or lesser extent) it is a challenge to chose the correct granularity.

The task has a number of assumptions:

Display the results are presented to the user as a ranked-list of results.

Users view the results top-down, one-by-one.

Relevant in Context Task The scenario underlying the Relevant in Context Task is the return of a ranked list of articles and within those articles the relevant information (captured by a set of non-overlapping elements or passages). A relevant article will likely contain relevant information that could be spread across different elements. The task requires systems to find a set of results that corresponds well to all relevant information in each relevant article. The task has a number of assumptions:

Display results will be grouped per article, in their original document order, access will be provided through further navigational means, such as a document heat-map or table of contents.

Users consider the article to be the most natural retrieval unit, and prefer an overview of relevance within this context.

Best in Context Task The scenario underlying the Best in Context Task is the return of a ranked list of articles and the identification of a best-entry-point from which a user should start reading each article in order to satisfy the information need. Even an article completely devoted to the topic of request will only have one best starting point from which to read (even if that is the beginning of the article). The task has a number of assumptions:

Display a single result per article.

Users consider articles to be natural unit of retrieval, but prefer to be guided to the best point from which to start reading the most relevant content.

2.2 Submission Format

Since XML retrieval approaches may return arbitrary results from within documents, a way to identify these nodes is needed. At INEX 2009, we allowed the submission of three types of results: XML elements, file-offset-length (FOL) text passages, and ranges of XML elements. The submission format for all tasks is a variant of the familiar TREC format extended with two additional fields.

```
topic Q0 file rank rsv run_id column.7 column.8
```

Here:

- The first column is the topic number.
- The second column (the query number within that topic) is currently unused and should always be Q0.
- The third column is the file name (without .xml) from which a result is retrieved, which is identical to the jid_i of the Wikipedia
- The fourth column is the rank the document is retrieved.
- The fifth column shows the retrieval status value (RSV) or score that generated the ranking.
- The sixth column is called the "run tag" identifying the group and for the method used.

Element Results XML element results are identified by means of a file name and an element (node) path specification. File names in the Wikipedia collection are unique, and (with the .xml extension removed) identical to the `<id>` of the Wikipedia document. That is, file `9996.xml` contains the article as the target document from the Wikipedia collection with `<id>` `9996`.

Element paths are given in XPath, but only fully specified paths are allowed. The next example identifies the first “article” element, then within that, the first “body” element, then the first “section” element, and finally within that the first “p” element.

```
/article[1]/body[1]/section[1]/p[1]
```

Importantly, XPath counts elements from 1 and counts element types. For example if a section had a title and two paragraphs then their paths would be: `title[1]`, `p[1]` and `p[2]`.

A result element may then be identified unambiguously using the combination of its file name (or `<id>`) in column 3 and the element path in column 7. Column 8 will not be used. Example:

```
1 Q0 9996 1 0.9999 I09UniXRun1 /article[1]/bdy[1]/sec[1]
1 Q0 9996 2 0.9998 I09UniXRun1 /article[1]/bdy[1]/sec[2]
1 Q0 9996 3 0.9997 I09UniXRun1 /article[1]/bdy[1]/sec[3]/p[1]
```

Here the results are from 9996 and select the first section, the second section, and the first paragraph of the third section.

FOL passages Passage results can be given in File-Offset-Length (FOL) format, where offset and length are calculated in characters with respect to the textual content (ignoring all tags) of the XML file. A special text-only version of the collection is provided to facilitate the use of passage retrieval systems. File offsets start counting a 0 (zero).

A result element may then be identified unambiguously using the combination of its file name (or `<id>`) in column 3 and an offset in column 7 and a length in column 8. The following example is effectively equivalent to the example element result above:

```
1 Q0 9996 1 0.9999 I09UniXRun1 465 3426
1 Q0 9996 2 0.9998 I09UniXRun1 3892 960
1 Q0 9996 3 0.9997 I09UniXRun1 4865 496
```

The results are from article 9996, and the first section starts at the 466th character (so 465 characters beyond the first character which has offset 0), and has a length of 3,426 characters.

Ranges of Elements To support ranges of elements, elemental passages can be specified by their containing elements. We only allow elemental paths (ending in an element, not a text-node in the DOM tree) plus an optional offset.

A result element may then be identified unambiguously using the combination of its file name (or `<id>`) in column 3, its start at the element path in column 7, and its end at the element path in column 8. Example:

```
1 Q0 9996 1 0.9999 I09UniRun1 /article[1]/bdy[1]/sec[1] /article[1]/bdy[1]/sec[1]
```

Here the result is again the first section from 9996. Note that the seventh column will refer to the beginning of an element (or its first content), and the eighth column will refer to the ending of an element (or its last content). Note that this format is very convenient for specifying ranges of elements, e.g., the first three sections:

```
1 Q0 9996 1 0.9999 I09UniXRun1 /article[1]/bdy[1]/sec[1] /article[1]/bdy[1]/sec[3]
```

2.3 Evaluation Measures

We briefly summarize the main measures used for the Ad Hoc Track. Since INEX 2007, we allow the retrieval of arbitrary passages of text matching the judges ability to regard any passage of text as relevant. Unfortunately this simple change has necessitated the deprecation of element-based metrics used in prior INEX campaigns because the “natural” retrieval unit is no longer an element, so elements cannot be used as the basis of measure. We note that properly evaluating the effectiveness in XML-IR remains an ongoing research question at INEX.

The INEX 2009 measures are solely based on the retrieval of highlighted text. We simplify all INEX tasks to highlighted text retrieval and assume that systems will try to return all, and only, highlighted text. We then compare the characters of text retrieved by a search engine to the number and location of characters of text identified as relevant by the assessor. For best in context we use the distance between the best entry point in the run to that identified by an assessor.

Thorough Task Precision is measured as the fraction of retrieved text that was highlighted. Recall is measured as the fraction of all highlighted text that has been retrieved. Text seen before is automatically discounted. The notion of rank is relatively fluid for passages so we use an interpolated precision measure which calculates interpolated precision scores at selected recall levels. Since we are most interested in overall performance, the main measure is mean average interpolated precision (MAiP), calculated over over 101 standard recall points (0.00, 0.01, 0.02, ..., 1.00). We also present interpolated precision at early recall points (iP[0.00], iP[0.01], iP[0.05], and iP[0.10]),

Focused Task As above, precision is measured as the fraction of retrieved text that was highlighted and recall is measured as the fraction of all highlighted text that has been retrieved. We use an interpolated precision measure which

calculates interpolated precision scores at selected recall levels. Since we are most interested in what happens in the first retrieved results, the main measure is interpolated precision at 1% recall (iP[0.01]). We also present interpolated precision at other early recall points, and (mean average) interpolated precision over 101 standard recall points (0.00, 0.01, 0.02, ..., 1.00) as an overall measure.

Relevant in Context Task The evaluation of the Relevant in Context Task is based on the measures of generalized precision and recall [7] over articles, where the per document score reflects how well the retrieved text matches the relevant text in the document. Specifically, the per document score is the harmonic mean of precision and recall in terms of the fractions of retrieved and highlighted text in the document. We use an F_β score with $\beta = 1/4$ making precision four times as important as recall. We are most interested in overall performances, so the main measure is mean average generalized precision (MAGP). We also present the generalized precision scores at early ranks (5, 10, 25, 50).

Best in Context Task The evaluation of the Best in Context Task is based on the measures of generalized precision and recall where the per document score reflects how well the retrieved entry point matches the best entry point in the document. Specifically, the per document score is a linear discounting function of the distance d (measured in characters)

$$\frac{n - d(x, b)}{n}$$

for $d < n$ and 0 otherwise. We use $n = 500$ which is roughly the number of characters corresponding to the visible part of the document on a screen. We are most interested in overall performance, and the main measure is mean average generalized precision (MAGP). We also show the generalized precision scores at early ranks (5, 10, 25, 50).

For further details on the INEX measures, we refer to [6]

3 Ad Hoc Test Collection

In this section, we discuss the corpus, topics, and relevance assessments used in the Ad Hoc Track.

3.1 Corpus

Starting in 2009, INEX uses a new document collection based on the Wikipedia. The original Wiki syntax has been converted into XML, using both general tags of the layout structure (like *article*, *section*, *paragraph*, *title*, *list* and *item*), typographical tags (like *bold*, *emphatic*), and frequently occurring link-tags. The annotation is enhanced with semantic markup of articles and outgoing links,

```

<article xmlns:xlink="http://www.w3.org/1999/xlink">
<holder confidence="0.9511911446218017" wordnetid="103525454">
<entity confidence="0.9511911446218017" wordnetid="100001740">
<musical_organization confidence="0.8" wordnetid="108246613">
<artist confidence="0.9511911446218017" wordnetid="109812338">
<group confidence="0.8" wordnetid="100031264">
<header>
<title>Queen (band)</title>
<id>42010</id>
...
</header>
<body>
...
<songwriter wordnetid="110624540" confidence="0.9173553029164789">
<person wordnetid="100007846" confidence="0.9508927676800064">
<manufacturer wordnetid="110292316" confidence="0.9173553029164789">
<musician wordnetid="110340312" confidence="0.9173553029164789">
<singer wordnetid="110599806" confidence="0.9173553029164789">
<artist wordnetid="109812338" confidence="0.9508927676800064">
<link xlink:type="simple" xlink:href="../068/42068.xml">
Freddie Mercury</link></artist>
</singer>
</musician>
</manufacturer>
</person>
</songwriter>
...
</body>
</group>
</artist>
</musical_organization>
</entity>
</holder>
</article>

```

Fig. 1. INEX 2009 Ad Hoc Track document 42010.xml (in part).

based on the semantic knowledge base YAGO, explicitly labeling more than 5,800 classes of entities like persons, movies, cities, and many more. For a more technical description of a preliminary version of this collection, see [10].

The collection was created from the October 8, 2008 dump of the English Wikipedia articles and incorporates semantic annotations from the 2008-w40-2 version of YAGO. It contains 2,666,190 Wikipedia articles and has a total uncompressed size of 50.7 Gb. There are 101,917,424 XML elements of at least 50 characters (excluding white-space).

Figure 1 shows part of a document in the corpus. The whole article has been encapsulated with tags, such as the `<group>` tag added to the Queen page.

This allows us to find particular article types easily, e.g., instead of a query requesting articles about Freddie Mercury:

```

<topic id="2009114" ct_no="310">
  <title>self-portrait</title>
  <castitle>//painter//figure[about(//caption, self-portrait)]</castitle>
  <phrasetitle>"self portrait"</phrasetitle>
  <description>Find self-portraits of painters.</description>
  <narrative>
    I am studying how painters visually depict themselves in their
    work. Relevant document components are images of works of art, in
    combination with sufficient explanation (i.e., a reference to the
    artist and the fact that the artist him/herself is depicted in the
    work of art). Also textual descriptions of these works, if
    sufficiently detailed, can be relevant. Document components
    discussing the portrayal of artists in general are not relevant, as
    are artists that figure in painters of other artists.
  </narrative>
</topic>

```

Fig. 2. INEX 2009 Ad Hoc Track topic 2009114.

```
//article[about(., Freddie Mercury)]
```

we can specifically ask about a group about Freddie Mercury:

```
//group[about(., Freddie Mercury)]
```

which will return pages of (pop) groups mentioning Freddy Mercury. In fact, also all internal Wikipedia links have been annotated with the tags assigned to the page they link to, e.g., in the example about the link to Freddie Mercury gets the `<singer>` tag assigned. We can also use these tags to identify pages where certain types of links occur, and further refine the query as:

```
//group[about(//singer, Freddie Mercury)]
```

The exact NEXI query format used to express the structural hints will be explained below.

3.2 Topics

The ad hoc topics were created by participants following precise instructions. Candidate topics contained a short CO (keyword) query, an optional structured CAS query, a phrase title, a one line description of the search request, and narrative with a details of the topic of request and the task context in which the information need arose. For candidate topics without a `<castitle>` field, a default CAS-query was added based on the CO-query: `/**[about(., "CO-query")]`. Figure 2 presents an example of an ad hoc topic. Based on the submitted candidate topics, 115 topics were selected for use in the INEX 2009 Ad Hoc Track as topic numbers 2009001–2009115.

Each topic contains

title A short explanation of the information need using simple keywords, also known as the content only (CO) query. It serves as a summary of the content of the user’s information need.

castitle A short explanation of the information need, specifying any structural requirements, also known as the content and structure (CAS) query. The castitle is optional but the majority of topics should include one.

phrasetitle A more verbose explanation of the information need given as a series of phrases, just as the `<title>` is given as a series of keywords.

description A brief description of the information need written in natural language, typically one or two sentences.

narrative A detailed explanation of the information need and the description of what makes an element relevant or not. The `<narrative>` should explain not only what information is being sought, but also the context and motivation of the information need, i.e., why the information is being sought and what work-task it might help to solve. Assessments will be made on compliance to the narrative alone; it is therefore important that this description is clear and precise.

The `<castitle>` contains the CAS query, an XPath expressions of the form: `A[B]` or `A[B]C[D]` where `A` and `C` are navigational XPath expressions using only the descendant axis. `B` and `D` are predicates using functions for text; the arithmetic operators `<`, `<=`, `>`, and `>=` for numbers; or the connectives `and` and `or`. For text, the `about` function has (nearly) the same syntax as the XPath function `contains`. Usage is restricted to the form `about(.path, query)` where `path` is empty or contains only tag-names and descendant axis; and `query` is an IR query having the same syntax as the CO titles (i.e. query terms). The `about` function denotes that the content of the element located by the path is about the information need expressed in the query. As with the title, the castitle is only a hint to the search engine and does not have definite semantics.

The purpose of the phrasetitle field is to explicate the order and grouping of the query terms in the title. The absence of a phrasetitle implies the absence of a phrase, e.g. a query with independent words. The title and phrasetitle together make the “phrase query” for phrase-aware search. Some topics come with quotations marks in the title, in which case the phrasetitle is at least partially redundant. However, we have made sure that the phrasetitle does not introduce words other than those in the title and that the identified phrases are encapsulated in quotation marks. This setting helps us study whether systems can improve their performance when given explicit phrases as opposed to individual words as implicit phrases.

3.3 Judgments

Topics were assessed by participants following precise instructions. The assessors used the GPXrai assessment system that assists assessors in highlight relevant text. Topic assessors were asked to mark all, and only, relevant text in a pool of documents. After assessing an article with relevance, a separate best entry point decision was made by the assessor. The Thorough, Focused and Relevant in Context Tasks were evaluated against the text highlighted by the assessors, whereas the Best in Context Task was evaluated against the best-entry-points.

Table 1. Statistics over judged and relevant articles per topic.

	total		# per topic				
	topics	number	min	max	median	mean	st.dev
judged articles	68	50,725	380	766	754	746.0	49.0
articles with relevance	68	4,858	5	351	52	71.4	72.5
highlighted passages	68	7,957	5	594	75.5	117.0	121.5
highlighted characters	68	18,838,137	4,453	2,776,635	97,550.5	277,031.4	442,113.9

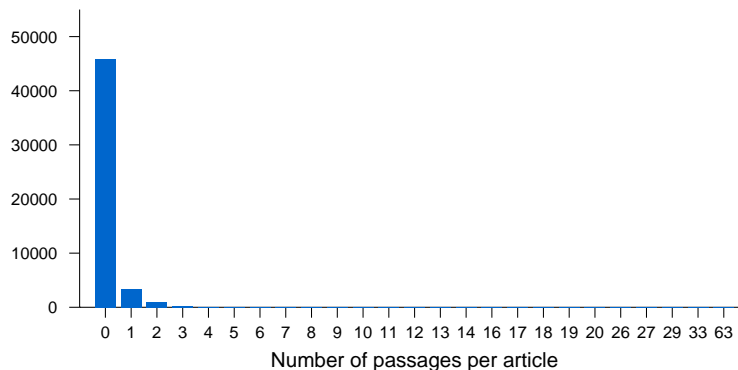


Fig. 3. Distribution of passages over articles.

The relevance judgments were frozen on November 10, 2009. At this time 68 topics had been fully assessed. Moreover, some topics were judged by two separate assessors, each without the knowledge of the other. All results in this paper refer to the 68 topics with the judgments of the first assigned assessor, which is typically the topic author.

- The 68 assessed topics were numbered 2009*n* with *n*: 001–006, 010–015, 020, 022, 023, 026, 028, 029, 033, 035, 036, 039–043, 046, 047, 051, 053–055, 061–071, 073, 074, 076–079, 082, 085, 087–089, 091–093, 095, 096, 104, 105, 108–113, and 115

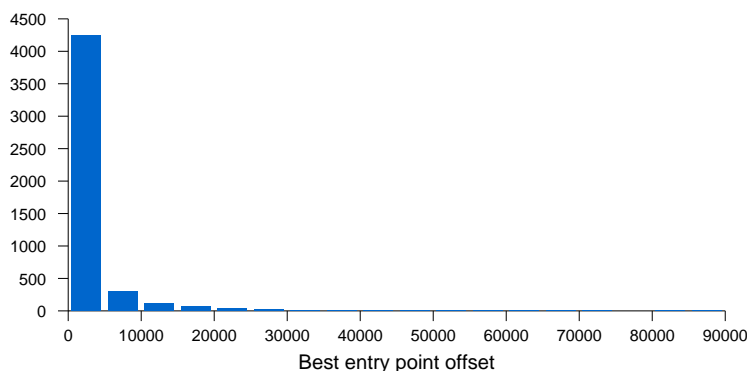
Table 1 presents statistics of the number of judged and relevant articles, and passages. In total 50,725 articles were judged. Relevant passages were found in 4,858 articles. The mean number of relevant articles per topic is 71, but the distribution is skewed with a median of 52. There were 7,957 highlighted passages. The mean was 117 passages and the median was 76 passages per topic.¹

Figure 3 presents the number of articles with the given number of passages. The vast majority of relevant articles (3,339 out of 4,858) had only a single highlighted passage, and the number of passages quickly tapers off.

¹ Recall from above that for the Focused Task the main effectiveness measures is precision at 1% recall. Given that the average topic has 117 relevant passages in 52 articles, the 1% recall roughly corresponds to a relevant passage retrieved—for many systems this will be accomplished by the first or first few results.

Table 2. Statistics over relevant articles.

	total		# per relevant article				
	topics	number	min	max	median	mean	st.dev
best entry point offset	68	4,858	2	86,545	311.5	2,493.2	6,481.8
first relevant character offset	68	4,858	2	86,545	295	2,463.0	6,375.6
length relevant documents	68	4,858	204	159,892	5,774.5	11,691.5	15,745.1
relevant characters	68	4,858	8	110,191	1,137	3,877.8	7,818.5
fraction highlighted text	68	4,858	0.00022	1.000	0.330	0.442	0.381

**Fig. 4.** Distribution of best entry point offsets.

Assessors were requested to provide a separate best entry point (BEP) judgment, for every article where they highlighted relevant text. Table 2 presents statistics on the best entry point offset, on the first highlighted or relevant character, and on the fraction of highlighted text in relevant articles. We first look at the BEPs. The mean BEP is well within the article with 2,493 but the distribution is very skewed with a median BEP offset of only 311. Figure 4 shows the distribution of the character offsets of the 4,858 best entry points. It is clear that the overwhelming majority of BEPs is at the beginning of the article.

The statistics of the first highlighted or relevant character (FRC) in Table 2 give very similar numbers as the BEP offsets: the mean offset of the first relevant character is 2,463 but the median offset is only 295. This suggests a relation between the BEP offset and the FRC offset. Figure 5 shows a scatter plot the BEP and FRC offsets. Two observations present themselves. First, there is a clear diagonal where the BEP is positioned exactly at the first highlighted character in the article. Second, there is also a vertical line at BEP offset zero, indicating a tendency to put the BEP at the start of the article even when the relevant text appears later on.

Table 2 also shows statistics on the length of relevant articles. Many articles are relatively short with a median length of 5,775 characters, the mean length is 11,691 characters. This is considerably longer than the INEX 2008 collection, where the relevant articles had a median length of 3,030 and a mean length of 6,793. The length of highlighted text in characters is on average 3,876 (mean

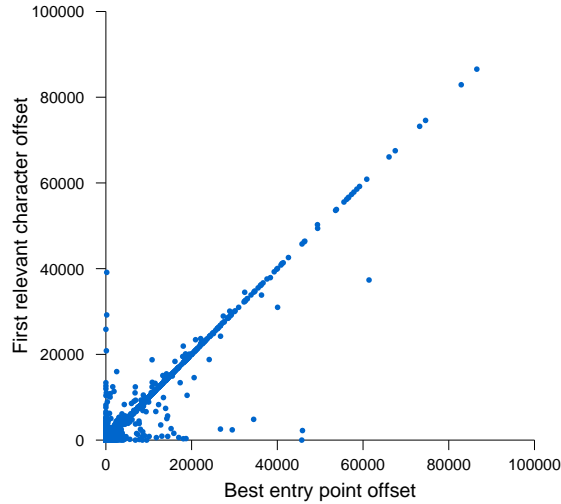


Fig. 5. Scatter plot of best entry point offsets versus the first relevant character.

1,137), in comparison to an average length of 2,338 (mean 838) in 2008. Table 2 also show that amount of relevant text varies from almost nothing to almost everything. The mean fraction is 0.44, and the median is 0.33, indicating that typically over one-third of the article is relevant. This is considerably less than the INEX 2008 collection, where over half of the text of articles was considered relevant. Given that the majority of relevant articles contain such a large fraction of relevant text plausibly explains that BEPs being frequently positioned on or near the start of the article.

3.4 Questionnaires

At INEX 2009, all candidate topic authors and assessors were asked to complete a questionnaire designed to capture the context of the topic author and the topic of request. The candidate topic questionnaire (shown in Table 3) featured 20 questions capturing contextual data on the search request. The post-assessment questionnaire (shown in Table 4) featured 14 questions capturing further contextual data on the search request, and the way the topic has been judged (a few questions on GPXrai were added to the end).

The responses to the questionnaires show a considerable variation over topics and topic authors in terms of topic familiarity; the type of information requested; the expected results; the interpretation of structural information in the search request; the meaning of a highlighted passage; and the meaning of best entry points. There is a need for further analysis of the contextual data of the topics in relation to the results of the INEX 2009 Ad Hoc Track.

Table 3. Candidate Topic Questionnaire.

-
- B1 How familiar are you with the subject matter of the topic?
 - B2 Would you search for this topic in real-life?
 - B3 Does your query differ from what you would type in a web search engine?
 - B4 Are you looking for very specific information?
 - B5 Are you interested in reading a lot of relevant information on the topic?
 - B6 Could the topic be satisfied by combining the information in different (parts of) documents?
 - B7 Is the topic based on a seen relevant (part of a) document?
 - B8 Can information of equal relevance to the topic be found in several documents?
 - B9 Approximately how many articles in the whole collection do you expect to contain relevant information?
 - B10 Approximately how many relevant document parts do you expect in the whole collection?
 - B11 Could a relevant result be (check all that apply): a single sentence; a single paragraph; a single (sub)section; a whole article
 - B12 Can the topic be completely satisfied by a single relevant result?
 - B13 Is there additional value in reading several relevant results?
 - B14 Is there additional value in knowing all relevant results?
 - B15 Would you prefer seeing: only the best results; all relevant results; don't know
 - B16 Would you prefer seeing: isolated document parts; the article's context; don't know
 - B17 Do you assume perfect knowledge of the DTD?
 - B18 Do you assume that the structure of at least one relevant result is known?
 - B19 Do you assume that references to the document structure are vague and imprecise?
 - B20 Comments or suggestions on any of the above (optional)

Table 4. Post Assessment Questionnaire.

-
- C1 Did you submit this topic to INEX?
 - C2 How familiar were you with the subject matter of the topic?
 - C3 How hard was it to decide whether information was relevant?
 - C4 Is Wikipedia an obvious source to look for information on the topic?
 - C5 Can a highlighted passage be (check all that apply): a single sentence; a single paragraph; a single (sub)section; a whole article
 - C6 Is a single highlighted passage enough to answer the topic?
 - C7 Are highlighted passages still informative when presented out of context?
 - C8 How often does relevant information occur in an article about something else?
 - C9 How well does the total length of highlighted text correspond to the usefulness of an article?
 - C10 Which of the following two strategies is closer to your actual highlighting:
(I) I located useful articles and highlighted the best passages and nothing more,
(II) I highlighted all text relevant according to narrative, even if this meant highlighting an entire article.
 - C11 Can a best entry point be (check all that apply): the start of a highlighted passage; the sectioning structure containing the highlighted text; the start of the article
 - C12 Does the best entry point correspond to the best passage?
 - C13 Does the best entry point correspond to the first passage?
 - C14 Comments or suggestions on any of the above (optional)

Table 5. Participants in the Ad Hoc Track.

Id Participant	Thorough	Focused	Relevant in Context	Best in Context	CO query	CAS query	Phrase query	Reference run	Element results	Range of elements results	FOL results	# valid runs	# submitted runs
4 University of Otago	0	0	1	0	1	0	0	1	1	0	0	1	1
5 Queensland University of Technology	4	12	12	12	20	20	0	0	32	8	0	40	48
6 University of Amsterdam	4	2	2	2	7	3	0	0	10	0	0	10	10
10 Max-Planck-Institut Informatik	3	8	0	2	11	2	1	0	13	0	0	13	13
16 University of Frankfurt	0	2	0	0	0	2	0	0	2	0	0	2	2
22 ENSM-SE	0	4	0	0	4	0	4	0	4	0	0	4	4
25 Renmin University of China	1	3	3	2	7	2	0	0	9	0	0	9	9
29 INDIAN STATISTICAL INSTITUTE	0	2	0	0	2	0	0	0	2	0	0	2	2
36 University of Tampere	0	0	3	3	6	0	0	2	4	2	0	6	6
48 LIG	3	3	3	3	12	0	0	4	12	0	0	12	12
55 Doshisha University	0	1	0	0	0	1	0	0	1	0	0	1	1
60 Saint Etienne University	3	4	3	3	13	0	0	4	13	0	0	13	13
62 RMIT University	0	0	0	2	2	0	0	0	1	0	1	2	2
68 University Pierre et Marie Curie - LIP6	2	2	0	0	4	0	0	0	4	0	0	4	4
72 University of Minnesota Duluth	2	3	3	1	9	0	0	0	9	0	0	9	9
78 University of Waterloo	0	4	0	0	4	0	0	0	2	0	2	4	4
92 University of Lyon3	2	2	0	2	5	1	6	0	6	0	0	6	8
167 School of Electronic Engineering and Computer Science	3	3	1	3	10	0	0	4	10	0	0	10	12
346 University of Twente	3	2	2	2	0	9	0	4	9	0	0	9	12
Total runs	30	57	33	37	117	40	11	19	144	10	3	157	172

4 Ad Hoc Retrieval Results

In this section, we discuss, for the four ad hoc tasks, the participants and their results.

4.1 Participation

A total of 172 runs were submitted by 19 participating groups. Table 5 lists the participants and the number of runs they submitted, also broken down over the tasks (Thorough, Focused, Relevant in Context, or Best in Context); the used query (Content-Only or Content-And-Structure); whether it used the Phrase query or Reference run; and the used result type (Element, Range of elements, or FOL passage). Unfortunately, no less than 15 runs turned out to be invalid and will only be evaluated with respect to their “article retrieval” value in Section 6.

Table 6. Top 10 Participants in the Ad Hoc Track Thorough Task.

Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p48-LIG-2009-thorough-3T	0.5967	0.5841	0.5444	0.5019	0.2855
p6-UAmSIN09article	0.5938	0.5880	0.5385	0.4981	0.2818
p5-BM25thorough	0.6168	0.5983	0.5360	0.4917	0.2585
p92-Lyon3LIAmanlmnt*	0.5196	0.4956	0.4761	0.4226	0.2496
p60-UJM_15494	0.5986	0.5789	0.5293	0.4813	0.2435
p346-utCASartT09	0.5461	0.5343	0.4929	0.4415	0.2350
p10-MPII-CASThBM	0.5860	0.5537	0.4821	0.4225	0.2133
p167-09RefT	0.3205	0.3199	0.2779	0.2437	0.1390
p68-I09LIP6OWATH	0.3975	0.3569	0.2468	0.1945	0.0630
p25-ruc-base-coT	0.5440	0.4583	0.3020	0.1898	0.0577

Participants were allowed to submit up to two element result-type runs per task and up to two passage result-type runs per task (for all four tasks). In addition, we allowed for an extra submission per task based on a reference run containing an article-level ranking using the BM25 model. This totaled to 20 runs per participant.² The submissions are spread well over the ad hoc retrieval tasks with 30 submissions for Thorough, 57 submissions for Focused, 33 submissions for Relevant in Context, and 37 submissions for Best in Context.

4.2 Thorough Task

We now discuss the results of the Thorough Task in which a ranked-list of non-overlapping results (elements or passages) was required. The official measure for the task was mean average interpolated precision (MAiP). Table 6 shows the best run of the top 10 participating groups. The first column gives the participant, see Table 5 for the full name of group. The second to fifth column give the interpolated precision at 0%, 1%, 5%, and 10% recall. The sixth column gives mean average interpolated precision over 101 standard recall levels (0%, 1%, . . . , 100%).

Here we briefly summarize what is currently known about the experiments conducted by the top five groups (based on official measure for the task, MAiP).

LIG Element retrieval run using the CO query. Description: Starting from 2K elements for each of the section types (sec, ss1, ss2, ss3, ss4) according to a multinomial language model with Dirichlet smoothing, we then interleave these five lists according to the score. We then group these results by the ranking of the reference run on articles, keeping within a document the element ranking. The run is based on the reference run.

University of Amsterdam Element retrieval run using the CO query. Description: A standard run on an article index, using a language model with a standard linear length prior. The run is retrieving only articles.

² As it turns out, one group submitted more runs than allowed: the *Queensland University of Technology* submitted 24 extra element runs. Some other groups submitted too many runs of a certain type or task. At this moment, we have not decided on any repercussions other than mentioning them in this footnote.

Queensland University of Technology Element retrieval run using the CO query. Description: Starting from a BM25 article retrieval run on an index of terms and tags-as-terms (produced by Otago), the top 50 retrieved articles are further processed by extracting the list of all (overlapping) elements which contained at least one of the search terms. The list is padded with the remaining articles, if needed.

University of Lyon3 A *manual* element retrieval run using the CO query. Description: Using Indri with Dirichlet smoothing and combining two language models: one of the full articles and one on the following tags: b, bdy, category, causal_agent, country, entry, group, image, it, list, location, p, person, physical_entity, sec, software, table, title. Special queries are created used NLP tools such as a summarizer and terminology extraction: the initial query based on the topic’s phrase and CO title is expanded with related phrases extracted from the other topic fields and from an automatic summary of the top ranked documents by this initial query. In addition, standard query expansion are used, skip phrases are allowed, and occurrences in the title are extra weighted.

Saint Etienne University Element retrieval run using the CO query. Description: Using BM25 on an element index with element frequency statistics. The b and k parameters were tuned on the INEX 2008 collection, leading to value different from standard document retrieval. The resulting run is filtered for elements from articles in the reference run, while retaining the original element ranking. The run is based on the reference run.

Based on the information from these and other participants:

- All ten runs use retrieve element type results. Three out of ten runs retrieve only article elements: the second ranked *p6-UAmSIN09article*, sixth ranked *p346-utCASartT09*, and the eighth ranked *p167-09RefT*.
- Eight of the ten runs use the CO query, the runs ranked sixth, *p346-utCASartT09*, and seventh, *p10-MPII-CASThBM* use the structured CAS query.
- Three runs are based on the *reference run*: the first ranked *p48-LIG-2009-thorough-3T*, the fifth ranked *p60-UJM.15494*, and the eighth ranked *p167-09RefT*

4.3 Focused Task

We now discuss the results of the Focused Task in which a ranked-list of non-overlapping results (elements or passages) was required. The official measure for the task was (mean) interpolated precision at 1% recall (iP[0.01]). Table 7 shows the best run of the top 10 participating groups. The first column gives the participant, see Table 5 for the full name of group. The second to fifth column give the interpolated precision at 0%, 1%, 5%, and 10% recall. The sixth column gives mean average interpolated precision over 101 standard recall levels (0%, 1%, ..., 100%).

Here we briefly summarize what is currently known about the experiments conducted by the top five groups (based on official measure for the task, iP[0.01]).

Table 7. Top 10 Participants in the Ad Hoc Track Focused Task.

Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p78-UWatFERBM25F	0.6797	0.6333	0.5006	0.4095	0.1854
p68-I09LIP6Okapi	0.6244	0.6141	0.5823	0.5290	0.3001
p10-MPII-COFoBM	0.6740	0.6134	0.5222	0.4474	0.1973
p60-UJM_15525	0.6241	0.6060	0.5742	0.4920	0.2890
p6-UamsFSsec2docbi100	0.6328	0.5997	0.5140	0.4647	0.1928
p5-BM25BOTrangeFOC	0.6049	0.5992	0.5619	0.5057	0.2912
p16-Spirix09R001	0.6081	0.5903	0.5342	0.4979	0.2865
p48-LIG-2009-focused-1F	0.5861	0.5853	0.5431	0.5055	0.2702
p22-emse2009-150*	0.6671	0.5844	0.4396	0.3699	0.1470
p25-ruc-term-coF	0.6128	0.4973	0.3307	0.2414	0.0741

University of Waterloo FOL passage retrieval run using the CO query. Description: the run uses the Okapi BM25 model in Wumpus to score all content-bearing elements such as sections and paragraphs. It uses a fielded Okapi BM25F over two fields: a title composed of the concatenation of article and all ancestor’s and current section titles, and a body field is the rest of the section. Training was done at element level and an average field length was used.

LIP6 Element retrieval run using the CO query. Description: A BM25 run with $b=0.2$ and $k=2.0$ and retrieving 1,500 articles for the CO queries, where negated words are removed from the query. For each document, the /article[1] element is retrieved. The run is retrieving only articles.

Max-Planck-Institut für Informatik Element retrieval run using the CO query. Description: Using EBM25, an XML-specific extension of BM25 using element frequencies of individual tag-term pairs, i.e., for each distinct tag and term, we precompute an individual element frequency, capturing the amount of tags under which the term appears in the entire collection. A static decay factor for the TF component is used to make the scoring function favor smaller elements rather than entire articles.

Saint Etienne University An element retrieval run using the CO query. Description: Using BM25 on an standard article index. The b and k parameters were tuned on the INEX 2008 collection. The run is retrieving only articles.

University of Amsterdam Element retrieval run using the CAS query. Description: Language model run on a non-overlapping section index with top 100 reranked using a link degree prior. The link degree prior is the indegree+outdegree using local links from the retrieved sections. The link degree prior is applied to the article level, thus all sections from the same article have the same link prior.

Based on the information from these and other participants:

- Seven runs use the CO query. Three runs, the fifth ranked *p6-UamsFSsec2docbi100*, the sixth ranked *p5-BM25BOTrangeFOC*, and the seventh ranked *p16-Spirix09R001* use the structured CAS query. The ninth run, *p22-emse2009-150*, uses a manually expanded query using words from the description and narrative fields.

Table 8. Top 10 Participants in the Ad Hoc Track Relevant in Context Task.

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p5-BM25RangeRIC	0.3345	0.2980	0.2356	0.1786	0.1885
p4-Reference	0.3311	0.2936	0.2298	0.1716	0.1847
p6-UamsRSCMartCMdocbi100	0.3192	0.2794	0.2074	0.1660	0.1773
p48-LIG-2009-RIC-1R	0.3027	0.2604	0.2055	0.1548	0.1760
p36-utampere_given30_nolinks	0.3128	0.2802	0.2101	0.1592	0.1720
p346-utCASrefR09	0.2216	0.1904	0.1457	0.1095	0.1188
p60-UJM_15502	0.2003	0.1696	0.1311	0.0998	0.1075
p167-09RefR	0.1595	0.1454	0.1358	0.1205	0.1045
p25-ruc-base-casF	0.2113	0.1946	0.1566	0.1380	0.1028
p72-umd_ric_1	0.0943	0.0801	0.0574	0.0439	0.0424

- Eight runs retrieve elements as results. The top ranked *p78-UWatFERBM25F* retrieves FOL passages, and the sixth ranked *p5-BM25BOTrangeFOC* retrieves ranges of elements.
- The systems at rank second, (*p68-I09LIP6Okapi*), fourth (*p60-UJM_15525*), and seventh (*p16-Spirix09R001*) are retrieving only full articles.

4.4 Relevant in Context Task

We now discuss the results of the Relevant in Context Task in which non-overlapping results (elements or passages) need to be returned grouped by the article they came from. The task was evaluated using generalized precision where the generalized score per article was based on the retrieved highlighted text. The official measure for the task was mean average generalized precision (MAgP).

Table 8 shows the top 10 participating groups (only the best run per group is shown) in the Relevant in Context Task. The first column lists the participant, see Table 5 for the full name of group. The second to fifth column list generalized precision at 5, 10, 25, 50 retrieved articles. The sixth column lists mean average generalized precision.

Here we briefly summarize the information available about the experiments conducted by the top five groups (based on MAgP).

Queensland University of Technology Run retrieving ranges of elements using the CO query. Description: Starting from a BM25 article retrieval run on an index of terms and tags-as-terms (produced by Otago), the top 50 retrieved articles are further processed by identifying the first and last element in the article (in reading order) which contained any of the search terms. The focused result was then specified as a range of two elements (which could be one and the same). The list is padded with the remaining articles.

University of Otago Element retrieval run using the CO query. Description: the run uses the Okapi BM25 model on an article index, with parameters trained on the INEX 2008 collection. The run is retrieving only articles and is based on the reference run—in fact, it is the original reference run.

University of Amsterdam Element retrieval run using the CO query. Description: The results from section index are grouped and ranked based on the the article ranking from the article index. The section run is reranked using the Wikipedia categories as background models before we cut-off the section run at 1,500 results per topic. The article run is similarly reranked using the Wikipedia categories as background models and link degree priors using the local incoming and outgoing links at article level.

LIG Element retrieval run using the CO query. Description: First, separate lists of 2K elements are generated for the element types sec, ss1, ss2, ss3, and ss4, the five lists are merged according to score. Second, an article ranking is obtained using a multinomial language model with Dirichlet smoothing. Third, the element results are group using the article ranking, by retaining with each article the reading order. Then we remove overlaps according to the reading order.

University of Tampere Element retrieval run using the CO query. Description: For each document the only retrieved passage was between the first and the last link to the top 30 documents. If there were no such links, the whole article was returned. The run is based on the reference run.

Based on the information from these and other participants:

- The runs ranked sixth (*p346-utCASrefR09*) and ninth (*p25-ruc-base-casF*) are using the CAS query. All other runs use only the CO query in the topic’s title field.
- The top scoring run retrieves ranges of elements, all other runs retrieve elements as results.
- Solid article ranking seems a prerequisite for good overall performance, with second best run, *p4-Reference* and the eighth best run, *p167-09RefR*, retrieving only full articles.

4.5 Best in Context Task

We now discuss the results of the Best in Context Task in which documents were ranked on topical relevance and a single best entry point into the document was identified. The Best in Context Task was evaluated using generalized precision but here the generalized score per article was based on the distance to the assessor’s best-entry point. The official measure for the task was mean average generalized precision (MAgP).

Table 9 shows the top 10 participating groups (only the best run per group is shown) in the Best in Context Task. The first column lists the participant, see Table 5 for the full name of group. The second to fifth column list generalized precision at 5, 10, 25, 50 retrieved articles. The sixth column lists mean average generalized precision.

Here we briefly summarize the information available about the experiments conducted by the top five groups (based on MAgP).

Table 9. Top 10 Participants in the Ad Hoc Track Best in Context Task.

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p5-BM25bepBIC	0.2941	0.2690	0.2119	0.1657	0.1711
p62-RMIT09titleO	0.3112	0.2757	0.2156	0.1673	0.1710
p10-MPII-COBIBM	0.2903	0.2567	0.2053	0.1598	0.1662
p48-LIG-2009-BIC-3B	0.2778	0.2564	0.1969	0.1469	0.1571
p6-UamsBAfbCMdocbi100	0.2604	0.2298	0.1676	0.1478	0.1544
p92-Lyon3LIAmanBEP*	0.2887	0.2366	0.1815	0.1482	0.1483
p36-utampere_given30_nolinks	0.2141	0.1798	0.1462	0.1234	0.1207
p346-utCASrefB09	0.1993	0.1737	0.1248	0.0941	0.1056
p25-ruc-term-coB	0.1603	0.1610	0.1274	0.0976	0.1013
p167-09LrnRefB	0.1369	0.1250	0.1181	0.1049	0.0953

Queensland University of Technology Element retrieval run using the CO query. Description: Starting from a BM25 article retrieval run on an index of terms and tags-as-terms (produced by Otago), the top 50 retrieved articles are further processed by identifying the first element (in reading order) containing any of the search terms. The list is padded with the remaining articles.

RMIT University Element retrieval run using the CO query. Description: Using Zettair with Okapi BM25 on an article-level index. The BEP is assumed to be at the start of the article. The run is retrieving only articles.

Max-Planck-Institut für Informatik Element retrieval run using the CO query. Description: Using EBM25, an XML-specific extension of BM25 using element frequencies of individual tag-term pairs, i.e., for each distinct tag and term, we precompute an individual element frequency, capturing the amount of tags under which the term appears in the entire collection. A static decay factor for the TF component is used to make the scoring function favor smaller elements rather than entire articles, but the final run returns the start of the article as BEP. The run is retrieving only articles.

LIG Element retrieval run using the CO query. Description: First, separate lists of 2K elements are generated for the element types sec, ss1, ss2, ss3, and ss4, the five lists are merged according to score. Second, an article ranking is obtained from the reference run. Third, for each article the best scoring element is used as the entry point. The run is based on the reference run.

University of Amsterdam Element retrieval run using the CO query. Description: Article index run with standard pseudo-relevance feedback (using Indri), reranked with Wikipedia categories as background models and link degree priors using the local incoming and outgoing links at article level. The run is retrieving only articles.

Based on the information from these and other participants:

- The second best run (*p62-RMIT09titleO*) retrieves FOL passages, all other runs return elements as results. The FOL passage run is a degenerate case that always puts the BEP at the start of the article.
- As for the Relevant in Context Task, we see again that solid article ranking is very important. In fact, we see runs putting the BEP at the start

Table 10. Statistical significance (t-test, one-tailed, 95%).

(a) Thorough Task											(b) Focused Task											
	1	2	3	4	5	6	7	8	9	10		1	2	3	4	5	6	7	8	9	10	
p48	-	-	*	-	*	-	*	*	*	*	p78	-	-	-	-	-	-	-	-	-	-	*
p6	-	*	-	*	-	*	*	*	*	*	p68	-	-	-	-	-	*	-	*	-	*	-
p5	*	-	*	-	*	*	*	*	*	*	p10	-	-	-	-	-	-	-	-	-	-	*
p92	-	-	-	*	*	*	-	-	-	-	p60	-	-	-	-	-	-	-	-	-	-	*
p60	-	-	*	*	*	*	-	-	-	-	p6	-	-	-	-	-	-	-	-	-	-	*
p346	-	*	*	*	*	-	-	-	-	-	p5	-	-	-	-	-	-	-	-	-	-	*
p10	*	*	*	*	-	-	-	-	-	-	p16	-	-	-	-	-	-	-	-	-	-	*
p167	-	-	-	-	-	-	-	-	-	-	p48	-	-	-	-	-	-	-	-	-	-	*
p68	-	-	-	-	-	-	-	-	-	-	p22	-	-	-	-	-	-	-	-	-	-	*
p25	-	-	-	-	-	-	-	-	-	-	p25	-	-	-	-	-	-	-	-	-	-	*

(c) Relevant in Context Task											(d) Best in Context Task											
	1	2	3	4	5	6	7	8	9	10		1	2	3	4	5	6	7	8	9	10	
p5	*	-	*	*	*	*	*	*	*	*	p5	-	-	*	*	-	*	*	*	*	*	*
p4	-	-	*	*	*	*	*	*	*	*	p62	-	*	-	-	*	*	*	*	*	*	*
p6	-	-	*	*	*	*	*	*	*	*	p10	-	-	-	*	*	*	*	*	*	*	*
p48	-	*	*	*	*	*	*	*	*	*	p48	-	-	*	*	*	*	*	*	*	*	*
p36	*	*	*	*	*	*	-	-	-	-	p6	-	*	*	*	*	*	*	*	*	*	*
p346	-	-	-	*	-	-	-	-	-	-	p92	-	*	*	*	*	*	*	*	*	*	*
p60	-	-	*	-	-	-	-	-	-	-	p36	-	-	*	-	-	-	-	-	-	-	*
p167	-	-	-	-	-	-	-	-	-	-	p346	-	-	-	-	-	-	-	-	-	-	-
p25	-	-	-	-	-	-	-	-	-	-	p25	-	-	-	-	-	-	-	-	-	-	-
p72	-	-	-	-	-	-	-	-	-	-	p167	-	-	-	-	-	-	-	-	-	-	-

of all the retrieved articles at rank two (*p62-RMIT09titleO*), rank three (*p10-MPII-COBIBM*), rank five (*p6-UamsBAfbCMdocbi100*), and rank ten (*p167-09LrnRefB*).

- With the exception of the run ranked eight (*p346-utCASrefB09*), which used the CAS query, all the other best runs per group use the CO query.

4.6 Significance Tests

We tested whether higher ranked systems were significantly better than lower ranked system, using a t-test (one-tailed) at 95%. Table 10 shows, for each task, whether it is significantly better (indicated by “*”) than lower ranked runs. For the Thorough Task, we see that the performance (measured by MAiP) of the top scoring run is significantly better than the runs at rank 4, 6, 8, 9, and 10. The same holds for the second and third best run. The fourth best run is significantly better than the runs at rank 8 and 9. The fifth, sixth, and seventh ranked runs are all significantly better than the runs at rank 8, 9, and 10. Of the 45 possible pairs of runs, there are 26 (or 58%) significant differences. For the Focused Task, we see that the early precision (at 1% recall) is a rather unstable measure. All runs are significantly better than the run at rank 10, the second best run also is significantly better than the run at rank 8. Of the 45 possible pairs of runs,

Table 11. Ad Hoc Track: Runs with ranges of elements or FOL passages.

(a) Focused Task					
Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p78-UWatFERBM25F	0.6797	0.6333	0.5006	0.4095	0.1854
p5-BM25BOTrangeFOC	0.6049	0.5992	0.5619	0.5057	0.2912

(b) Relevant in Context Task					
Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p5-BM25RangeRIC	0.3345	0.2980	0.2356	0.1786	0.1885
p36-utampere_auth_40_top30	0.2717	0.2509	0.2006	0.1583	0.1185

(c) Best in Context Task					
Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p62-RMIT09titleO	0.3112	0.2757	0.2156	0.1673	0.1710

there are only 10 (or 22%) significant differences. Hence we should be careful when drawing conclusions based on the Focused Task results. For the Relevant in Context Task, we see that the top run is significantly better than ranks 2 and 4 through 10. The second best run is significantly better than ranks 5 through 10. The third, fourth, and fifth ranked systems are significantly better than ranks 6 through 10. The sixth to ninth systems are significantly better than rank 10. Of the 45 possible pairs of runs, there are 33 (or 73%) significant differences, making MAgP a very discriminative measure. For the Best in Context Task, we see that the top run is significantly better than ranks 4 and 5, and 7 through 10. The second best run is significantly better than than ranks 4 and 7 to 10. The third, fourth, and fifth ranked runs are significantly better than than ranks 7 to 10. The seventh ranked system is better than the systems ranked 8 to 10, and the eighth ranked system better than rank 9 10. Of the 45 possible pairs of runs, there are 27 (or 60%) significant differences.

5 Analysis of Run and Topic Types

In this section, we will discuss relative effectiveness of element and passage retrieval approaches, and on the relative effectiveness of systems using the keyword and structured queries.

5.1 Elements versus passages

We received 13 submissions using ranges of elements of FOL-passage results, from in total 4 participating groups. We will look at the relative effectiveness of element and passage runs.

As we saw above, in Section 4, for three tasks there were high ranking runs using FOL passages or ranges of elements in the top 10. Table 11 shows the best runs using ranges of elements or FOL passages for three ad hoc tasks, there were no such submissions for the Thorough Task. As it turns out, the best focused run retrieving FOL passages was the top ranked run in Table 7; the best relevant

Table 12. Ad Hoc Track: Runs using the phrase query.

(a) Thorough Task					
Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p92-Lyon3LIAmanlmnt*	0.5196	0.4956	0.4761	0.4226	0.2496

(b) Focused Task					
Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p22-emse2009-150*	0.6671	0.5844	0.4396	0.3699	0.1470
p10-MPII-COArBPP	0.5563	0.5477	0.5283	0.4681	0.2566
p92-Lyon3LIAmanQE*	0.4955	0.4861	0.4668	0.4271	0.2522

(c) Best in Context Task					
Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p92-Lyon3LIAmanBEP*	0.2887	0.2366	0.1815	0.1482	0.1483

in context retrieving ranges of elements was the top scoring run in Table 8; and the best best in context run retrieving FOL passages was the second best run in Table 9. Given the low number of submissions using passages or ranges of elements, this is an impressive result. However, looking at the runs in more detail, their character is often unlike what one would expect from a “passage” retrieval run. For Focused, *p5-BM25BOTrangeFOC* is an article retrieving run using ranges of elements, based on the CAS query. For Relevant in Context, *p5-BM25RangeRIC* is an article retrieving run using ranges of elements. For Best in Context, *p62-RMIT09titleO* is an article run using FOL passages. Hence, this is not sufficient evidence to warrant any conclusion on the effectiveness of passage level results. We hope and expect that the test collection and the passage runs will be used for further research into the relative effectiveness of element and passage retrieval approaches.

5.2 Phrase queries

We received 10 submissions based on the phrase query. Table 12 shows the best runs using the phrase query for three of the ad hoc tasks, there were no valid submissions using the phrase title for Relevant in Context. The best phrase submission for the Thorough Task did rank 5th in the overall results. The best phrase submission for the Focused Task did rank 9th in the overall results. The best phrase submission for the Best in Context Task did rank 6th in the overall results.

Although few runs were submitted, the phrase title seems competitive, but not superior to the use of the CO query. The only participant submitting both types of runs, the *Max-Planck-Institute für Informatik* for the Focused Task, had marginally better performance for the CO query run over all 68 topics, and marginally better performance for the combined CO and Phrase title run over the 60 topics having a proper phrase in the Phrase title field. The differences between the query types are very small. A possible explanation for this is that all CO query have been expanded to contain the same terms as the more verbose phrase query. Hence the only difference is the explicit phrase markup, which

Table 13. Ad Hoc Track: Runs using the reference run.

(a) Thorough Task					
Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p48-LIG-2009-thorough-3T	0.5967	0.5841	0.5444	0.5019	0.2855
p60-UJM_15494	0.5986	0.5789	0.5293	0.4813	0.2435
p346-utCASrefF09	0.4834	0.4525	0.4150	0.3550	0.1982
p167-09RefT	0.3205	0.3199	0.2779	0.2437	0.1390

(b) Focused Task					
Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p48-LIG-2009-focused-3F	0.5946	0.5822	0.5344	0.5018	0.2732
p60-UJM_15518	0.5559	0.5136	0.4003	0.3104	0.1019
p346-utCASrefF09	0.4801	0.4508	0.4139	0.3547	0.1981
p167-09LrnRefF	0.3162	0.3072	0.2512	0.2223	0.1292

(c) Relevant in Context Task					
Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p4-Reference	0.3311	0.2936	0.2298	0.1716	0.1847
p48-LIG-2009-RIC-3R	0.3119	0.2790	0.2193	0.1629	0.1757
p36-utampere_given30_nolinks	0.3128	0.2802	0.2101	0.1592	0.1720
p346-utCASrefR09	0.2216	0.1904	0.1457	0.1095	0.1188
p167-09RefR	0.1595	0.1454	0.1358	0.1205	0.1045
p60-UJM_15503	0.1825	0.1548	0.1196	0.0953	0.1020

(d) Best in Context Task					
Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p48-LIG-2009-BIC-3B	0.2778	0.2564	0.1969	0.1469	0.1571
p36-utampere_given30_nolinks	0.2141	0.1798	0.1462	0.1234	0.1207
p346-utCASrefB09	0.1993	0.1737	0.1248	0.0941	0.1056
p167-09LrnRefB	0.1369	0.1250	0.1181	0.1049	0.0953
p60-UJM_15508	0.1274	0.1123	0.0878	0.0735	0.0795

requires special handling by the search engines. The available test collection with explicit phrases marked up in 60 topics is a valuable result of INEX 2009, and it can be studied in-depth in future experiments.

5.3 Reference run

There were 19 submissions using the reference run. Table 13 shows the best runs using the reference runs for the four ad hoc tasks. For the Thorough Task, the best submission based on the reference run ranked first. For the Focused Task, the best submission based on the reference run would have ranked tenth. For the Relevant in Context Task, the best submission based on the reference run—in fact, the actual reference run itself—ranked second. For the Best in Context Task, the best submission based on the reference run ranked fourth. The results show that the reference run indeed provides competitive article ranking that forms a good basis for retrieval.

There are also considerable differences in performance of the runs based on the same reference run. This suggests that the runs do not retrieve the exact

Table 14. Top 10 Participants in the Ad Hoc Track: Article retrieval based on the reference run.

Participant	P5	P10	1/rank	map	bpref
p4-Reference	0.6147	0.5294	0.8240	0.3477	0.3333
p36-utampere_given30_nolinks	0.6147	0.5294	0.8240	0.3477	0.3333
p48-LIG-2009-BIC-3B	0.6147	0.5294	0.8240	0.3463	0.3336
p60-UJM_15508	0.5324	0.4544	0.7020	0.2910	0.2925
p346-utCASrefB09	0.5441	0.4750	0.7494	0.2833	0.2768
p167-09RefT	0.3765	0.3603	0.5761	0.2443	0.2540

same set of articles. As explained later, in Section 6, we can look at the article rankings induced by the runs. Table 14 shows the best run of the top 10 participating groups, using the reference run. With the exception of *p36-utampere_given30_nolinks* the article rankings of the runs vary considerably.

5.4 CO versus CAS

We now look at the relative effectiveness of the keyword (CO) and structured (CAS) queries. As we saw above, in Section 4, one of the best runs per group for the Relevant in Context Task, and two of the top 10 runs for the Best in Context Task used the CAS query.

All topics have a CAS query since artificial CAS queries of the form

```
/**[about(., keyword title)]
```

were added to topics without CAS title. Table 15 show the distribution of target elements, with YAGO tags in emphatic. In total 81 topics had a non-trivial CAS query.³ These CAS topics are numbered 2009*n* with *n*: 001–009, 011–013, 015–017, 020–025, 028–032, 036, 037, 039–045, 048–053, 057, 058, 060, 061, 064–072, 074, 080, 085–096, 098, 099, 102, 105, 106, and 108–115. As it turned out, 50 of these CAS topics were assessed. The results presented here are restricted to only these 50 CAS topics.

Table 16 lists the top 10 participants measured using just the 50 CAS topics and for the Thorough Task (a and b) and the Focused Task (c and d). For the Thorough Task the best CAS run, *p5-BM25BOTthorough*, would have ranked sixth amongst the CO runs on MAiP. The two participants submitting both CO and CAS runs had better MAiP scores for the CO runs. However, the best CAS run has higher scores on early precision, iP[0.00] through iP[0.05] than any of the CO submissions. For the Focused Task the best CAS run, *p6-UamsFSsec2docbi100*, would have ranked fifth amongst the CO runs. Two participants submitting both CO and CAS runs had better iP[0.01] scores for the CO runs, one participant had a better CAS run. For Relevant in Context Task (not shown), the best CAS run, *p5-BM25BOTrangeRIC*, would have ranked third among the CO runs. One participants submitting both CO and CAS runs had

³ Note that some of the wild-card topics (using the “*” target) in Table 15 had non-trivial about-predicates and hence have not been regarded as trivial CAS queries.

Table 15. CAS query target elements over all 115 topics (YAGO tags slanted).

Target Element	Frequency
*	41
article	32
sec	9
group	5
p	4
<i>music_genre</i>	2
<i>vehicles</i>	1
<i>theory</i>	1
<i>song</i>	1
<i>revolution</i>	1
(p sec person)	1
(p sec)	1
<i>protest</i>	1
(<i>person chemist alchemist scientist physicist</i>)	1
<i>personality</i>	1
<i>museum</i>	1
link	1
image	1
<i>home</i>	1
<i>food</i>	1
figure	1
<i>facility</i>	1
<i>driver</i>	1
<i>dog</i>	1
<i>director</i>	1
(<i>classical_music opera orchestra performer singer</i>)	1
<i>bicycle</i>	1
(article sec p)	1

better MAgP scores for a CO run, another participant had a better CAS run. For the Best in Context Task (not shown), the best CAS run, *p5-BM25BOTbepBIC*, would rank seventh among the CO runs. All three participants submitting both CO and CAS runs had better MAgP scores for their CO runs. Overall, we see that teams submitting runs with both types of queries have higher scoring CO runs, with participant 5 as a notable exception for Focused.

6 Analysis of Article Retrieval

In this section, we will look in detail at the effectiveness of Ad Hoc Track submissions as article retrieval systems.

6.1 Article retrieval: Relevance Judgments

We will first look at the topics judged during INEX 2009, but now using the judgments to derive standard document-level relevance by regarding an article

Table 16. Ad Hoc Track CAS Topics: CO runs versus CAS runs.

(a) Thorough Task: CO runs					
Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p48-LIG-2009-thorough-1T	0.5781	0.5706	0.5315	0.4834	0.2729
p6-UamsIN09article	0.5900	0.5821	0.5149	0.4613	0.2629
p92-Lyon3LIAmanlmnt*	0.5365	0.5039	0.4794	0.4330	0.2450
p5-BM25thorough	0.6273	0.6023	0.5191	0.4620	0.2389
p60-UJM_15494	0.6034	0.5766	0.5131	0.4612	0.2280
p10-MPII-COThBM	0.6436	0.5916	0.5135	0.3783	0.1909
p167-09RefT	0.3245	0.3237	0.2682	0.2392	0.1291
p68-I09LIP6OWATh	0.4146	0.3651	0.2512	0.1963	0.0608
p25-ruc-base-coT	0.5328	0.4333	0.2538	0.1653	0.0505
p72-umd.thorough.3	0.4073	0.2893	0.1697	0.0999	0.0494

(b) Thorough Task: CAS runs					
Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p5-BM25BOTthorough	0.6460	0.6169	0.5359	0.4472	0.2279
p346-utCASartT09	0.5541	0.5381	0.4819	0.4136	0.2227
p10-MPII-CASThBM	0.5747	0.5308	0.4406	0.3627	0.1651

(c) Focused Task: CO runs					
Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p78-UWatFERBM25F	0.6742	0.6222	0.4905	0.3758	0.1737
p60-UJM_15525	0.6373	0.6127	0.5696	0.4585	0.2811
p10-MPII-COArBM	0.6201	0.6060	0.5387	0.4648	0.2684
p68-I09LIP6Okapi	0.6130	0.6005	0.5660	0.5064	0.2798
p5-ANTbigramsRangeFOC	0.6089	0.5936	0.5331	0.4531	0.2597
p48-LIG-2009-focused-3F	0.5971	0.5802	0.5205	0.4775	0.2583
p22-emse2009-150*	0.6453	0.5598	0.4211	0.3471	0.1371
p92-Lyon3LIAmanQE*	0.5185	0.5058	0.4815	0.4339	0.2472
p25-ruc-term-coF	0.6277	0.4955	0.2900	0.2065	0.0668
p167-09LrnRefF	0.3357	0.3234	0.2536	0.2211	0.1216

(c) Focused Task: CAS runs					
Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p6-UamsFSsec2docbi100	0.6151	0.5974	0.4851	0.4230	0.1718
p16-Spirix09R001	0.6201	0.5958	0.5386	0.4920	0.2794
p5-BM25BOTrangeFOC	0.6031	0.5954	0.5470	0.4789	0.2713
p10-MPII-CASFoBM	0.5643	0.5161	0.4454	0.3634	0.1644
p25-ruc-base-casF	0.5114	0.4775	0.4077	0.3214	0.1666
p346-utCASrefF09	0.4353	0.3955	0.3477	0.2781	0.1471
p55-doshisha09f	0.1273	0.0651	0.0307	0.0227	0.0060

as relevant if some part of it is highlighted by the assessor. We derive an article retrieval run from every submission using a first-come, first served mapping. That is, we simply keep every first occurrence of an article (retrieved indirectly through some element contained in it) and ignore further results from the same article.

Table 17. Top 10 Participants in the Ad Hoc Track: Article retrieval.

Participant	P5	P10	1/rank	map	bpref
p6-UamsTAbi100	0.6500	0.5397	0.8555	0.3578	0.3481
p48-LIG-2009-BIC-1B	0.6059	0.5338	0.8206	0.3573	0.3510
p62-RMIT09title	0.6029	0.5279	0.8237	0.3540	0.3488
p5-BM25ArticleRIC	0.6147	0.5294	0.8240	0.3477	0.3333
p4-Reference	0.6147	0.5294	0.8240	0.3477	0.3333
p36-utampere_given30_nolinks	0.6147	0.5294	0.8240	0.3477	0.3333
p68-I09LIP6OWA	0.6118	0.5147	0.8602	0.3420	0.3258
p10-MPII-COArBP	0.6353	0.5471	0.8272	0.3371	0.3458
p92-Lyon3LIAmanQE*	0.6265	0.5265	0.7413	0.3335	0.3416
p78-UWatFERBase	0.5765	0.5088	0.8093	0.3267	0.3205

We use `trec_eval` to evaluate the mapped runs and `qrels`, and use mean average precision (map) as the main measure. Since all runs are now article retrieval runs, the differences between the tasks disappear. Moreover, runs violating the task requirements are now also considered, and we work with all 172 runs submitted to the Ad Hoc Track.

Table 17 shows the best run of the top 10 participating groups. The first column gives the participant, see Table 5 for the full name of group. The second and third column give the precision at ranks 5 and 10, respectively. The fourth column gives the mean reciprocal rank. The fifth column gives mean average precision. The sixth column gives binary preference measures (using the top R judged non-relevant documents). No less than seven of the top 10 runs retrieve exclusively full articles: only rank two (*p48-LIG-2009-BIC-1B*), rank six (*p36-utampere_given30_nolinks*) and rank ten (*p78-UWatFERBase*) retrieve elements proper. The relative effectiveness of these article retrieval runs in terms of their article ranking is no surprise. Furthermore, we see submissions from all four ad hoc tasks. A run from the Thorough task at rank 1; runs from the Best in Context task at ranks 2 and 3; runs from the Relevant in Context task at ranks 4, 5 and 6; and runs from the Focused task at ranks 7, 8, 9 and 10.

If we break-down all runs over the original tasks, shown in Table 18), we can compare the ranking to Section 4 above. We see some runs that are familiar from the earlier tables: five Thorough runs correspond to Table 6, four Focused runs correspond to Table 7, six Relevant in Context runs correspond to Table 8, and five Best in Context runs correspond to Table 9. More formally, we looked at how the two system rankings correlate using kendall’s tau.

- Over all 30 Thorough Task submissions the system rank correlation is 0.646 between MAiP and map.
- Over all 57 Focused task submissions the system rank correlation is 0.420 between $iP[0.01]$ and map, and 0.638 between MAiP and map.
- Over all 33 Relevant in Context submissions the system rank correlation between MAgP and map is 0.598.
- Over all 37 Best in Context submissions the system rank correlation between MAgP and map is 0.517.

Table 18. Top 10 Participants in the Ad Hoc Track: Article retrieval per task.

(a) Thorough Task					
Participant	P5	P10	1/rank	map	bpref
p6-UamsTAbi100	0.6500	0.5397	0.8555	0.3578	0.3481
p48-LIG-2009-thorough-1T	0.6118	0.5191	0.8042	0.3493	0.3392
p92-Lyon3LIAmanlmnt*	0.6382	0.5279	0.7706	0.3305	0.3374
p5-BM25thorough	0.6147	0.5294	0.8240	0.3188	0.3142
p10-MPII-COThBM	0.5853	0.5206	0.8084	0.3087	0.3138
p346-utCASartT09	0.5176	0.4588	0.7138	0.2913	0.2986
p60-UJM_15486	0.5647	0.4765	0.7149	0.2797	0.2884
p68-I09LIP6OWATh	0.4735	0.4353	0.7100	0.2665	0.2745
p72-umd.thorough_3	0.5382	0.4515	0.7406	0.2486	0.2674
p167-09RefT	0.3765	0.3603	0.5761	0.2443	0.2540

(b) Focused Task					
Participant	P5	P10	1/rank	map	bpref
p48-LIG-2009-focused-1F	0.6059	0.5338	0.8206	0.3569	0.3506
p5-BM25ArticleFOC	0.6147	0.5294	0.8240	0.3477	0.3333
p68-I09LIP6OWA	0.6118	0.5147	0.8602	0.3420	0.3258
p10-MPII-COArBP	0.6353	0.5471	0.8272	0.3371	0.3458
p92-Lyon3LIAmanQE*	0.6265	0.5265	0.7413	0.3335	0.3416
p78-UWatFERBase	0.5765	0.5088	0.8093	0.3267	0.3205
p60-UJM_15525	0.5824	0.4926	0.8326	0.3256	0.3169
p16-Spirix09R002	0.5206	0.4588	0.7250	0.3133	0.3149
p6-UamsFSsec2docbi100	0.5941	0.4779	0.8958	0.2985	0.2994
p346-utCASartF09	0.5176	0.4588	0.7138	0.2913	0.2986

(c) Relevant in Context Task					
Participant	P5	P10	1/rank	map	bpref
p48-LIG-2009-RIC-1R	0.6059	0.5338	0.8206	0.3569	0.3506
p6-UamsRSCMartCMdocbi100	0.6324	0.5309	0.9145	0.3523	0.3374
p5-BM25ArticleRIC	0.6147	0.5294	0.8240	0.3477	0.3333
p4-Reference	0.6147	0.5294	0.8240	0.3477	0.3333
p36-utampere_given30_nolinks	0.6147	0.5294	0.8240	0.3477	0.3333
p346-utCOartR09	0.5324	0.4882	0.7448	0.3120	0.3137
p72-umd.ric_2	0.5441	0.4544	0.7807	0.2708	0.2867
p167-09RefR	0.3765	0.3603	0.5761	0.2443	0.2540
p25-ruc-base-casF	0.4441	0.4176	0.6270	0.2243	0.2523
p60-UJM_15488	0.4382	0.3853	0.6043	0.2146	0.2343

(d) Best in Context Task					
Participant	P5	P10	1/rank	map	bpref
p48-LIG-2009-BIC-1B	0.6059	0.5338	0.8206	0.3573	0.3510
p62-RMIT09title	0.6029	0.5279	0.8237	0.3540	0.3488
p5-BM25AncestorBIC	0.6147	0.5294	0.8240	0.3477	0.3333
p36-utampere_given30_nolinks	0.6147	0.5294	0.8240	0.3477	0.3333
p6-UamsBAfbCMdocbi100	0.6147	0.5118	0.8531	0.3361	0.3251
p10-MPII-COBIBM	0.5824	0.5191	0.8451	0.3325	0.3315
p92-Lyon3LIAmanBEP*	0.6382	0.5279	0.7706	0.3305	0.3374
p25-ruc-term-coB	0.5206	0.4779	0.7158	0.3197	0.3251
p346-utCOartB09	0.5324	0.4882	0.7448	0.3120	0.3137
p60-UJM_15508	0.5324	0.4544	0.7020	0.2910	0.2925

Overall, we see a reasonable correspondence between the rankings for the ad hoc tasks in Section 4 and the rankings for the derived article retrieval measures. The correlation between article retrieval and the “in context” tasks was much higher (0.79) for the INEX 2008 collection. A likely effect of the increasing length of (relevant) Wikipedia articles.

7 Discussion and Conclusions

In this paper we provided an overview of the INEX 2009 Ad Hoc Track that contained four tasks: For the *Thorough Task* a ranked-list of results (elements or passages) by estimated relevance was required. For the *Focused Task* a ranked-list of non-overlapping results (elements or passages) was required. For the *Relevant in Context Task* non-overlapping results (elements or passages) grouped by the article that they belong to were required. For the *Best in Context Task* a single starting point (element’s starting tag or passage offset) per article was required. We discussed the results for the four tasks, and analysed the relative effectiveness of element and passage runs, of runs using phrases, of runs using the reference run, and of keyword (CO) queries and structured queries (CAS). We also look at effectiveness in term of article retrieval.

Given the efforts put into the fair comparison of element and passage retrieval approaches, the number submissions using FOL passages and range of elements was disappointing. Thirteen submissions used ranges of elements or FOL passage results, whereas 144 submissions used element results. In addition, several of the passage or FOL submissions used exclusively full articles as results. Still the non-element submissions were competitive with the top ranking runs for both the Focused and Relevant in Context Tasks, and the second ranking run for the Best in Context Task. There were too few submissions to draw any definite conclusions, but the outcome broadly confirms earlier results using passage-based element retrieval [3, 4].

There were also few submissions using the explicitly annotated phrases of the phrase query: ten in total. Phrase query runs were competitive with several of them in the overall top 10 results, but the impact of the phrases seemed marginal. Recall, that the exact same terms were present in the CO query, and the only difference was the phrase annotation. This is in line with earlier work. The use of phrases in queries has been studied extensively. In early publications, the usage of phrases and proximity operators showed improved retrieval results but rarely anything substantial [e.g., 2]. As retrieval models became more advanced, the usage of query operators was questioned. E.g., Mitra et al. [8] conclude that when using a good ranking algorithm, phrases have no effect on high precision retrieval (and sometimes a negative effect due to topic drift). Rasolofo and Savoy [9] combine term-proximity heuristics with an Okapi model, obtaining marginal improvements for early precision but with hardly observable impact on the MAP scores.

There were 19 submissions using the reference run providing a solid article ranking for further processing. These runs turned out to be competitive, with

runs in the top 10 for all tasks. Hence the reference run was successful in helping participants to create high quality runs. However, run based on the reference run were not directly comparable, since participants used these runs in different ways leading to substantially different underlying article rankings.

When examining the relative effectiveness of CO and CAS we found that for all tasks the best scoring runs used the CO query but some CAS runs were in the top 10 for all four tasks. Part of the explanation may be in the low number of CAS submissions (40) in comparison with the number of CO submissions (117). Only 50 of the 68 judged topics had a non-trivial CAS query, and the majority of those CAS queries made only reference to particular tags and not on their structural relations. The YAGO tags potentially expressing an information need naturally in terms of structural constraints, were popular: 36 CAS queries used them (21 of them judged). Over the 50 non-trivial CAS queries, most groups had a better performing run using the CO query. A notable exception was *QUT* who had better performance for CAS on the Focused Task. This is in accordance with earlier results showing that structural hints can help promote initial precision [5].

As in earlier years, we saw that article retrieval is a reasonably effective at XML-IR: for each of the ad hoc tasks there were three article-only runs among the best runs of the top 10 groups. When looking at the article rankings inherent in all Ad Hoc Track submissions, we saw that again three of the best runs of the top 10 groups in terms of article ranking (across all three tasks) were in fact article-only runs. This also suggests that element-level or passage-level evidence is valuable for article retrieval. When comparing the system rankings in terms of article retrieval with the system rankings in terms of the ad hoc retrieval tasks, over the exact same topic set, we see a reasonable correlation. The systems with the best performance for the ad hoc tasks, also tend to have the best article rankings.

Finally, the Ad Hoc Track had three main research questions. The first main research question was to study the effect of the new collection. We saw that the collection's size had little impact, but that the relevant articles were much longer (a mean length 3,030 in 2008 and 5,775 in 2009, a 52% increase), leading to a lower fraction of highlighted text per article (a mean of 58% in 2008 and 33% in 2009). This also reduced the correlation with article retrieval, e.g., from 79% for the "in context" tasks in 2008 to 51–58% in 2009. The second main research question was the impact of verbose queries using phrases or structural hints. The relatively few phase query submissions showed only marginal differences. The CAS query runs were in general less effective than the CO query runs, with one notable exception for the early precision measures of the Focused Task. The second main research question was the comparative analysis of element and passage retrieval approaches, hoping to shed light on the value of the document structure as provided by the XML mark-up. Despite the low number of non-element runs, we saw that some of the best performing system used FOL passages or ranges of elements. For all main research questions, we hope and expect that the resulting test collection will prove its value in future use. After all, the

main aim of the INEX initiative is to create bench-mark test-collections for the evaluation of structured retrieval approaches.

Acknowledgments Jaap Kamps was supported by the Netherlands Organization for Scientific Research (NWO, grants 612.066.513, 639.072.601, and 640.001.501).

Bibliography

- [1] C. L. A. Clarke. Range results in XML retrieval. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 4–5, Glasgow, UK, 2005.
- [2] W. B. Croft, H. R. Turtle, and D. D. Lewis. The use of phrases and structured queries in information retrieval. In *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 32–45, 1991.
- [3] W. Huang, A. Trotman, and R. A. O’Keefe. Element retrieval using a passage retrieval approach. In *Proceedings of the 11th Australasian Document Computing Symposium (ADCS 2006)*, pages 80–83, 2006.
- [4] K. Y. Itakura and C. L. A. Clarke. From passages into elements in XML retrieval. In *Proceedings of the SIGIR 2007 Workshop on Focused Retrieval*, pages 17–22. University of Otago, Dunedin New Zealand, 2007.
- [5] J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. Articulating information needs in XML query languages. *Transactions on Information Systems*, 24:407–436, 2006.
- [6] J. Kamps, J. Pehcevski, G. Kazai, M. Lalmas, and S. Robertson. INEX 2007 evaluation measures. In *Focused access to XML documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2007)*, volume 4862 of *Lecture Notes in Computer Science*, pages 24–33. Springer Verlag, Heidelberg, 2008.
- [7] J. Kekäläinen and K. Järvelin. Using graded relevance assessments in IR evaluation. *Journal of the American Society for Information Science and Technology*, 53:1120–1129, 2002.
- [8] M. Mitra, C. Buckley, A. Singhal, and C. Cardie. An analysis of statistical and syntactic phrases. In *Proceedings of RIAO-97*, 1997.
- [9] Y. Rasolofo and J. Savoy. Term proximity scoring for keyword-based retrieval systems. In *Proceedings of the 25th European Conference on IR Research (ECIR 2003)*, pages 207–218, 2003.
- [10] R. Schenkel, F. M. Suchanek, and G. Kasneci. YAWN: A semantically annotated Wikipedia XML corpus. In *12. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web (BTW 2007)*, pages 277–291, 2007.
- [11] A. Trotman and S. Geva. Passage retrieval and other XML-retrieval tasks. In *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology*, pages 43–50. University of Otago, Dunedin New Zealand, 2006.

A Appendix: Full run names

Group	Run	Label	Task	Query	Results	Notes
4	617	Reference	RiC	CO	Ele	Reference run Article-only
5	744	BM25AncestorBIC	BiC	CO	Ele	Article-only
5	757	BM25thorough	Tho	CO	Ele	
5	775	BM25ArticleFOC	Foc	CO	Ele	Article-only
5	781	BM25BOTrangeFOC	Foc	CAS	Ran	Article-only
5	792	ANTbigramsRangeFOC	Foc	CO	Ran	Article-only
5	796	BM25ArticleRIC	RiC	CO	Ele	Article-only
5	797	BM25RangeRIC	RiC	CO	Ran	Article-only
5	804	BM25BOTrangeRIC	RiC	CAS	Ran	Article-only
5	808	BM25BOTthorough	Tho	CAS	Ele	
5	824	BM25bepBIC	BiC	CO	Ele	Article-only
5	825	BM25BOTbepBIC	BiC	CAS	Ele	Article-only
6	634	UamsIN09article	Tho	CO	Ele	Article-only
6	810	UamsTAbi100	Tho	CO	Ele	Article-only
6	813	UamsFSsec2docbi100	Foc	CAS	Ele	
6	814	UamsRSCMartCMdocbi100	RiC	CO	Ele	
6	816	UamsBAfbCMdocbi100	BiC	CO	Ele	Article-only
6	817	UamsBSfbCMsec2docbi100art1	BiC	CAS	Ele	Article-only
10	618	MPII-CASFoBM	Foc	CAS	Ele	
10	619	MPII-COFoBM	Foc	CO	Ele	
10	620	MPII-CASThBM	Tho	CAS	Ele	
10	621	MPII-COThBM	Tho	CO	Ele	
10	628	MPII-COArBM	Foc	CO	Ele	Article-only
10	632	MPII-COBIBM	BiC	CO	Ele	Article-only
10	700	MPII-COArBP	Foc	CO	Ele	Article-only
10	709	MPII-COArBPP	Foc	CO	Ele	Phrases Article-only
16	872	Spirix09R001	Foc	CAS	Ele	Article-only
16	873	Spirix09R002	Foc	CAS	Ele	Article-only
22	672	emse2009-150	Foc	CO	Ele	Phrases Manual
25	727	ruc-base-coT	Tho	CO	Ele	
25	737	ruc-term-coB	BiC	CO	Ele	
25	738	ruc-term-coF	RiC	CO	Ele	
25	739	ruc-term-coF	Foc	CO	Ele	
25	898	ruc-base-casF	Foc	CAS	Ele	
25	899	ruc-base-casF	RiC	CAS	Ele	
36	688	utampere_given30_nolinks	RiC	CO	Ele	Reference run
36	701	utampere_given30_nolinks	BiC	CO	Ele	Reference run
36	708	utampere_auth_40_top30	RiC	CO	Ran	
48	682	LIG-2009-thorough-1T	Tho	CO	Ele	
48	684	LIG-2009-thorough-3T	Tho	CO	Ele	Reference run
48	685	LIG-2009-focused-1F	Foc	CO	Ele	
48	686	LIG-2009-focused-3F	Foc	CO	Ele	Reference run
48	714	LIG-2009-RIC-1R	RiC	CO	Ele	
48	716	LIG-2009-RIC-3R	RiC	CO	Ele	Reference run
48	717	LIG-2009-BIC-1B	BiC	CO	Ele	

Continued on Next Page. . .

Group	Run	Label	Task	Query	Results	Notes
48	719	LIG-2009-BIC-3B	BiC	CO	Ele	Reference run
55	836	doshisha09f	Foc	CAS	Ele	
60	819	UJM_15518	Foc	CO	Ele	Reference run
60	820	UJM_15486	Tho	CO	Ele	
60	822	UJM_15494	Tho	CO	Ele	Reference run
60	827	UJM_15488	RiC	CO	Ele	
60	828	UJM_15502	RiC	CO	Ele	
60	829	UJM_15503	RiC	CO	Ele	Reference run
60	830	UJM_15490	BiC	CO	Ele	
60	832	UJM_15508	BiC	CO	Ele	Reference run
60	868	UJM_15525	Foc	CO	Ele	Article-only
62	895	RMIT09title	BiC	CO	Ele	Article-only
62	896	RMIT09titleO	BiC	CO	FOL	Article-only
68	679	I09LIP6Okapi	Foc	CO	Ele	Article-only
68	681	I09LIP6OWA	Foc	CO	Ele	Article-only
68	704	I09LIP6OWATh	Tho	CO	Ele	
72	666	umd_ric_1	RiC	CO	Ele	
72	667	umd_ric_2	RiC	CO	Ele	
72	870	umd_thorough_3	Tho	CO	Ele	
78	706	UWatFERBase	Foc	CO	FOL	
78	707	UWatFERBM25F	Foc	CO	FOL	
92	694	Lyon3LIAautoBEP	BiC	CAS	Ele	Phrases
92	695	Lyon3LIAmanBEP	BiC	CO	Ele	Phrases Manual Article-only
92	697	Lyon3LIAmanQE	Foc	CO	Ele	Phrases Manual Article-only
92	699	Lyon3LIAmanlmnt	Tho	CO	Ele	Phrases Manual
167	651	09RefT	Tho	CO	Ele	Reference run Article-only
167	654	09LrnRefF	Foc	CO	Ele	Reference run Article-only
167	657	09RefR	RiC	CO	Ele	Reference run Article-only
167	660	09LrnRefB	BiC	CO	Ele	Reference run Article-only
346	637	utCASartT09	Tho	CAS	Ele	Article-only
346	638	utCASartF09	Foc	CAS	Ele	Article-only Invalid
346	639	utCOartR09	RiC	CO	Ele	Article-only Invalid
346	640	utCOartB09	BiC	CO	Ele	Article-only Invalid
346	645	utCASrefF09	Tho	CAS	Ele	Reference run
346	646	utCASrefF09	Foc	CAS	Ele	Reference run
346	647	utCASrefR09	RiC	CAS	Ele	Reference run
346	648	utCASrefB09	BiC	CAS	Ele	Reference run

LIP6 at Inex'09 : OWPC for adhoc track

David Buffoni and Patrick Gallinari

Laboratoire d'Informatique de Paris 6
104, avenue du PrÃ©sident-Kennedy, F-75016 Paris, France
{buffoni, gallinari}@poleia.lip6.fr

Abstract.

1 Introduction

2 Data preparation

Indexing XML elements without annotation element. We didn't stem the corpus and we didn't use a stop words list. Concerning queries, we use only the title part and removing all negative terms.

2.1 Manually Assessments

The collection is new so we had to build some learning sets for our models.

Training set We assessed manually 20 randomly choosed queries from last INEX competitions (from 2006 to 2008). We keep the same interrogation strategy as told before as take only the title part of queries removing stop words and without stemming step made on words.

The assessment protocol was made by pooling technique where we took top k results of n models which give us a better diversity of results. So for each query we ran three models (BM25[5] with $b = 0.5$ and $k = 1.2$, LogTF[?] and a Language Model with an Absolute Discount smoothing function[?] with $\delta = 0.88$). Then for each results list returned by each model, we selected relevant documents in the top 50 of the list.

At the end, for each relevant document we judged only relevant elements. According to us, we considered only element which could be a member of these XML tags : $\{article, title, bdy, section, p, item\}$. An element is described as relevant when it contains any relevant text according to the query. In total, 214 documents with at least one relevant element have been assessed which gives a total of 1285 relevant elements for 20 queries.

Validation set We need to build a validation set of queries to help us to select the hyperparameter of our models (C in the ??). We randomly took another 20 queries from the INEX 2009 competition. For that we decided to use the reference run given by organisers where we assessed only elements/documents in the top 50 of the list. So we stored 552 relevant elements spread in 164 documents.

2.2 Extracted features for learning

3 Models

4 Experiments

5 Results

5.1 Focused task

5.2 Thorough Task

6 Conclusion

References

1. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. In Jordan, M.I., Kearns, M.J., Solla, S.A., eds.: *Advances in Neural Information Processing Systems*. Volume 10., The MIT Press (1998)
2. 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)
3. Xu, J., Li, H.: AdaRank: A Boosting Algorithm for Information Retrieval. In: *SIGIR '07: Proceedings of the 28th annual international ACM SIGIR conference*. (2007)
4. Tsai, M-F., Liu, T-Y., Qin, T., Chen, H-H., Ma, W-Y. : FRank: A Ranking Method with Fidelity Loss. In: *SIGIR '07: Proceedings of the 28th annual international ACM SIGIR conference*. (2007)
5. Robertson, S.E., Walker, S., Hancock-Beaulieu, M., Gull, A., Lau, M.: Okapi at TREC. In: *Text REtrieval Conference*. (1992) 21–30

Peking University at INEX 2009: Ad Hoc Track

Ning Gao¹, Zhi-Hong Deng¹, Yong-Qing Xiang¹, and Hang Yu¹

¹ Key Laboratory of Machine Perception (Ministry of Education)
School of Electronic Engineering and Computer Science, Peking University
nanacream@gmail.com; zhdeng@cis.pku.edu.cn; xiangyq@cis.pku.edu.cn;
pkucthh@gmail.com

Abstract. This paper describes Peking University' approach to the Ad Hoc Track. In our first participation, results for all four tasks were submitted: the Best In Context, the Focused, the Relevance In Context and the Thorough. Based on retrieval method Okapi BM25, we implement two different ranking methods NormalBM25 and LearningBM25 according to different parameter settings. Specially, the parameters used in LearningBM25 are learnt by a new learning method called ListBM. The evaluation result shows that LearningBM25 is able to beat NormalBM25 in most tasks.

1 Introduction

INEX Ad Hoc Track^[1] aims to evaluate performance in retrieving relevant results (e.g. XML elements or documents) to a certain query. Based on lots of research and comparative experiments, Okapi BM25^[2] is confirmed to be an effective ranking method. It takes both text and structure information into consideration. Plus, evaluation results of Ad Hoc Track show that Okapi BM25 performs better than some other frequently cited ranking models, such as TF*IDF^[3] and so on. Motivated by BM25's excellent performance, many participants prefer BM25 as their basic retrieval model. In INEX 2008 Ad Hoc Track, University of Waterloo^[4] outperforms in all three tasks of Measured as Focused Retrieval, known as Best in Context, Focused and Relevance in Context. The ranking system Waterloo used is "a biased BM25 and language modeling, in addition to Okapi BM25"^[4].

However, in Okapi BM25 formula, there are several parameters used to adjust the proportion of element length and term frequency (tf) in the final score and they are frequently set manually by participants. Here we note that different parameter settings might lead to totally different evaluation results. Thus, in order to get a more rigorous, evidence-based and data-based parameter setting, a listwise machine learning method to learn the parameter settings is proposed. We call it listBM.

In detail, figure1 shows the architecture of our ranking system. Firstly, when user submits a query, a results recognizer will calculate the result elements by matching the Keywords and the Inverted Index. An element is defined as a result element only if it contains all the keywords. The output of results recognizer is a Results Set, in which result elements are disordered. Therefore, a ranking method BM25 is introduced to sort

these result elements according to their relevance to the query. However, according to different parameter settings used in BM25, we implement two different ranking models NormalBM25 and LearningBM25. In NormalBM25, the parameter setting we used is same as what Waterloo used in INEX 2008. We call this parameter setting the Origin Parameter Setting. The result list ranked by using this parameter setting is called NormalBM25 Results. While in LearningBM25 model, the parameters are learned by a machine learning method called ListBM, to be introduced in section 3. The result list ranked by BM25 using this Learnt Parameter Setting is defined as LearningBM25 Results. We submit both the NormalBM25 results and the LearningBM25 results in four tasks of INEX 2009 Ad Hoc Track. The evaluation results show that LearningBM25 results perform better than NormalBM25 results, indicating that our learning method ListBM indeed help to improve the performance.

In section 2, we introduce the concepts of BM25 and background of machine learning method we used. Section 3 describes our learning method ListBM. In section 4, we show the evaluation results. Section 5 is the conclusion and future work.

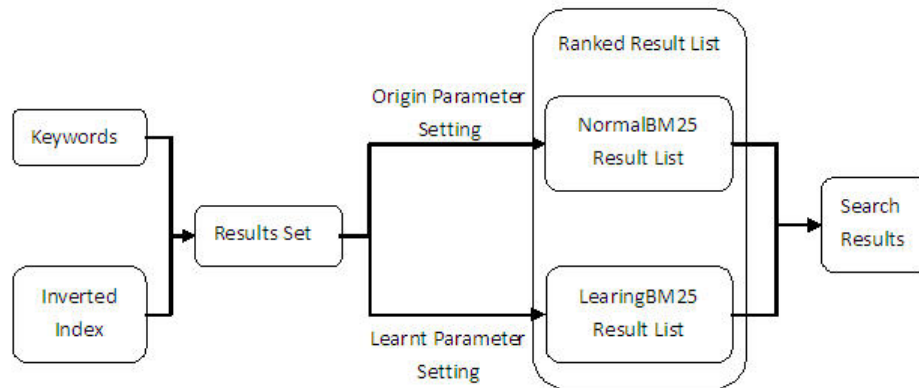


Fig. 1. Architecture of Ranking System

2 Related Work

2.1 Okapi BM25

BM25 is a widely quoted ranking method. It shows excellent performance referring to the evaluation results of INEX in the past years. For our method, to score an element according to its relevance to a certain query, we chose BM25 as our basic ranking model. In detail, the score is defined as follows.

$$score(e, Q) = \sum_{t \in Q} W_t \frac{(k_1 + 1)tf(t, e)}{k_1(1 - b + b \frac{len(e)}{avel}) + tf(t, e)} \quad (1)$$

$score(e, Q)$ measures the relevance of element e to a certain query Q ; W_t is the weight of term t indicating the inverse term frequency (IDF) of term t in collection. $tf(t, e)$ is the frequency of term t appearing in element e ; Nd denotes the number of files in data collection; $n(t)$ denotes the occurrence of term t in collection; $len(e)$ denotes the length of element e and $avel$ denotes the average length of elements in whole collection. Two parameters, k_1 and b , are used to balance the weight of term frequency (tf) and element length(len) in final score.

2.2 Learning-to-Rank Methods

Learning-to-Rank methods focus on using machine learning algorithms for better ranking. Many learning-to-rank algorithms have been proposed. According to different "instance" they use, learning-to-rank methods can be classified into three categories^[5]: pointwise, pairwise and listwise.

In pointwise methods, documents are used as learning instance. The relevant score of a document is calculated by its features such as term frequency (tf), tag name and document links. This kind of algorithm attempts to find classification engine that can mark document as relevant or irrelevant correctly.

Pairwise methods, such as RankBoost^[6] and RankSVM^[7], take document pair as learning instance. Consider two documents d_1 and d_2 , if d_1 is more relevant than d_2 to a certain query Q , then the document pair (d_1, d_2) is set to 1, otherwise it is set to -1. Pairwise methods target at training a learning engine to find the best document pair preferences.

In listwise methods, document list is taken as learning instance to train ranking engines. To find the best ranked list is the final goal. There are several well-known listwise methods such as Listnet^[5], ListMLE^[8], SVM-MAP^[9] and so on.

Comparative tests^[10] have shown that listwise methods perform best in these three categories.

3 Learning Method ListBM

3.1 ListBM

In NormalBM model, the parameter setting used in BM25 is same as waterloo set in INEX 2008. In LearningBM model, the parameters are learnt by a learning method called ListBM. We use INEX 2008 data collection as the training data base.

In training, there is a set of query $Q = \{q^1, q^2, \dots, q^m\}$. Each query q^i is associated with a

Algorithm 1 ListBM: the process of learning k_1

Input: query Q, relevant documents D
Parameter: k_1 , number of iterations T,
Initialize: $b=0.8, \sum_{i=1}^m L(D^i, R^i)=+\infty$

for $t=1$ **to** T **do**
 for $i=1$ **to** m **do**
 search the q^i using BM25, get R^i ;
 compute $L(D^i, R^i)$
 update $k_1 += \frac{1}{L(D^i, R^i)}$
 end for
 if $\sum_{i=1}^m L(D^i, R^i)$ increases, then break;
end for

output the pair of $\{k_{\min}, \min \sum_{i=1}^m L(D^i, R^i); \}$

Fig. 2. Algorithm1

ranked list of relevant documents $D^i = \{d_1^i, d_2^i, \dots, d_n^i\}$, where d_j^i denotes the j -th relevant document to query q^i . These relevant documents lists, downloaded from the website of INEX, are used as standard results in our training. What's more, for each query q^i , we use basic ranking method BM25 mentioned in 2.1 to get a list of search results RL^i . Results returned by BM25 are all in form of elements. The first n result elements of RL^i are recorded in $R^i = \{r_1^i, r_2^i, \dots, r_n^i\}$. Then each documents list D^i and elements list R^i form a "instance".

The loss function is defined as the "distance" between standard results lists D^i and search results lists R^i . Therefore, the objective of learning is formalized as minimization of the total losses with respect to the training data.

$$\sum_{i=1}^m L(D^i, R^i) \quad (2)$$

Suppose there are two search results R, R' and a standard result D, the definition of loss function should meet the following two criterions:

- The loss value should be inversely proportional to the recall. If R contains more relevant results appeared in D than R' does, then the loss value of R should be smaller than the loss value of R'.
- The loss value should be inversely proportional to the precision. If the relevant content contained by R has the higher relevance degree (they appear in the top-ranking documents in D), then the loss value of R should be lower.

According to these two criterions, we define the loss function for a single query q^i as:

$$L(D^i, R^i) = \frac{mn^2}{\sum_{j=1}^n rank_j^i} \quad (3)$$

$$rank_j^i = \begin{cases} 0, & \text{if the } j\text{-th result in } R^i \text{ does not appear in } D^i \\ k, & \text{if the } j\text{-th result in } R^i \text{ is contained by the } k\text{-th result in } D^i \end{cases} \quad (4a)$$

Where m is the number of queries in Q and n is the number of relevant documents in D^i corresponding to a certain query q^i . $rank_j^i$ is divided into two parts: (1) if the j -th result element in R^i isn't contained by any relevant documents in D^i , then $rank_j^i$ is set to 0; (2) if the j -th result element in R^i is contained by the k -th relevant document d_k^i in D^i , then $rank_j^i$ is set to k .

Using the pair of standard results D and searching results R as "instance", L defined above as loss function, we implement a learning method ListBM to learn the two parameters k_1 and b in BM25 separately. Figure 2 describes the procedure of learning parameter k_1 .

When learning k_1 , parameter b will be initialized to 0.8. While randomly entering an original value of k_1 , the loop won't end until the new updated k_1 begin to increase the loss value. The output is a pair of $\{k_{min}, \min \sum_{i=1}^m L(D^i, R^i)\}$, in which $\min \sum_{i=1}^m L(D^i, R^i)$ is the minimum loss value got in the process and k_{min} is the corresponded k_1 . While learning parameter b , k_1 is a content value 4 and b is updated according to the loss function. What's more, the output will be the pair of $\{b_{min}, \min \sum_{i=1}^m L(D^i, R^i)\}$, in which b_{min} is the better b leading to minimum loss value.

3.2 Experiments

We implemented our ranking system in C++. The data collection we use is the English files of wiki provided by INEX 2008 Ad Hoc Track. The total size with these 659,388 files is 4.6G. In the process of reading and analyzing these XML files, we remove all the stop word from a standard stop word list before stemming. We use 8 queries from INEX 2008 topic pool in the training process. Totally 4800 documents are signed as relevant results. Figure 2 and figure 3 show the learning results of k_1 and b .

Figure3 illustrates the learning results of k_1 . In NormalBM model, the k_1 is set to 4. As is shown, in total 4800 search results, only 1572 results are relevant according to the standard results when k_1 is set to 3.277. The best set of k_1 is 35 so that the relevant results can reach up to 1704.

Figure4 shows the learning result of b . The origin set of b is 0.8 according to the parameter setting of Waterloo. Result shows that $b = 0.8$ is indeed the best set leading to a better searching performance. Hence, the parameter setting of $\{k_1, b\}$ is set to $\{35, 0.8\}$ in LearningBM model and $\{4, 0.8\}$ in NormalBM model.

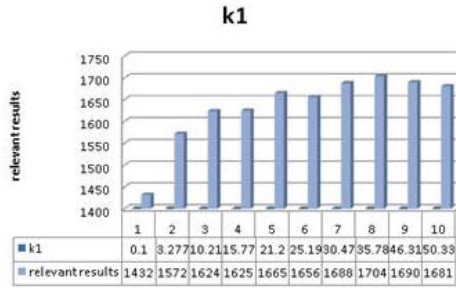


Fig. 3. learning result of k_1

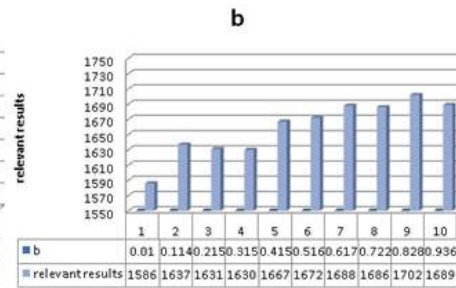


Fig. 4. learning result of b

4 Evaluation Results

We submit both NormalBM results and LearningBM results for all four tasks to compare the performance. We get evaluation results for three of four tasks. Table 1 shows the evaluation results of Measured as Focused Retrieval in which the search results are elements. Table 2 describes the search performance of Measured as Document Retrieval tasks. We can say that, in most conditions, the retrieval effectiveness of LearningBM is better than that of NormalBM.

Table 1. evaluation results of element retrieval

Measured as Focused Retrieval			
	Best in Context	Focused	Through
LerningBM25	0.0953	0.3072	0.0577
NormalBM25	0.0671	0.1779	0.0521

Table 2. evaluation results of element retrieval

Measured as Document Retrieval			
	Best in Context	Focused	Through
LerningBM25	0.2382	0.2382	0.1797
NormalBM25	0.1849	0.1847	0.1689

5 Conclusion and Future Work

We propose a new learning method ListBM to learn the parameters in ranking method BM25. ListBM is a listwise learning method, using the data source of INEX 2008 Ad Hoc Track as training data base. The evaluation results show that parameter setting learnt by ListBM performs better than parameter setting set manually.

For the future work, we will continue to work on the following problems:

- The training data we used is the collection of INEX 2008 Ad Hoc Track. However, data collection has changed a lot for INEX 2009. We will study the learning to rank method on the new collection in the future.
- There are only 8 queries used in training. For the further study, more queries from INEX 2009 topics pool will be searched in the learning process.
- We will propose new definition of "distance" between the search result list and the standard result list. Furthermore, a more reasonable loss function and a new updating method will be introduced.

References

1. <http://www.inex.otago.ac.nz/>
2. M. Theobald, R. Schenkel, G. Wiekum. *An Efficient and Versatile Query Engine for TopX Search*. In *VLDB*, 625–636, 2005.
3. D. Carmel, Y.S. Maarek, M. Mandelbrod, et al. *Searching XML documents via XML fragments*. In *SIGIR*, 151–158, 2003.
4. Kelly Y. Itakura, Charles L. A. Clarke. *University of Waterloo at INEX2008: Adhoc, Book, and Link-the-Wiki Tracks*. In *INEX workshop*, 116–122, 2008.
5. Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, Hang Li. *Learning to Rank: From Pairwise Approach to Listwise Approach*. *Microsoft technique report*.
6. Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. *An efficient boosting algorithm for combining preferences*. *J. Mach. Learn. Res.*, 933–969, 2003.
7. Herbrich, R., Graepel, T., Obermayer, K. *Support vector learning for ordinal regression*. In *Proceedings of ICANN*, 97–102, 1999.
8. Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. *Listwise approach to learning to rank: theory and algorithm*. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, 1192–1199, 2008.
9. Y. Yue, T. Finley, F. Radlinski and T. Joachims. *A Support Vector Method for Optimizing Average Precision*, In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.
10. Min Zhang, Da Kuang, Guichun Hua, Yiqun Liu, Shaoping Ma. *Is learning to rank effective for Web search?* In *SIGIR'09 workshop*, 2009.

UJM at INEX 2009 Ad Hoc track

Mathias Géry, Christine Largeton

Université de Lyon, F-42023, Saint-Étienne, France

CNRS UMR 5516, Laboratoire Hubert Curien

Université de Saint-Étienne Jean Monnet, F-42023, France

{mathias.gery, christine.largeton}@univ-st-etienne.fr

Abstract. This paper¹ presents our participation to the INEX 2009 Ad Hoc track. We have experimented the tuning of various parameters using a "learning" collection (i.e. INEX 2008) quite different than the "testing" collection used for 2009 INEX Ad Hoc. Several parameters have been studied for articles retrieval as well as for element retrieval, especially the two main BM25 weighting function parameters : b and k_1 .

1 Introduction

The focused information retrieval (IR) aims at exploiting the documents structure in order to retrieve the relevant elements (parts of documents) for a user information need. The structure can be used to emphasize some particular words or some parts of the document: the importance of a term depends on its formatting (*e.g.* bold font, italic, etc.), and also on its position in the document (*e.g.*, title terms versus text body). During our previous INEX participations, we have developed a probabilistic model that learn a weight for XML tags, representing the tag capability to emphasize relevant text fragments [2]. One interesting result is that articles retrieval based on BM25 weighting gives good results against elements retrieval, even when considering a precision oriented measure (i.e. $iP[0.01]$): 3 articles retrieval runs appear in the top-10 of the focused task (2nd, 4th and 8th, cf. [5]), and the 3 best *MAiP* runs are 3 articles retrieval runs! Thus a question comes: "Is BM25 suitable for elements retrieval"? Indeed, we can imagine that, BM25 being developed for articles retrieval, its adaptation to element retrieval is a challenging problem ? This problem has been addressed *e.g.* with BM25e weighting [6].

Our objectives during INEX 2009 was to answer to three questions:

- is it possible to apply our probabilistic model on this 2009 new collection ?
- is it possible to reuse the parameters tuned with INEX 2008 collection?
- is it still possible to obtain very good results with articles retrieval against elements retrieval? (i.e. articles retrieval gives good results considering *MAiP*, and elements retrieval is just slightly better considering $iP[0.01]$).

¹ This work has been partly funded by the Web Intelligence project (région Rhône-Alpes, cf. <http://www.web-intelligence-rhone-alpes.org>).

The new INEX collection does not allow to answer the first question, because we do not have a training collection using the same XML tags than in INEX 2009. This paper deals with the two other questions, using the 2008 INEX collection as a training collection. We present the experimental protocol in section 2, then our system overview in section 3, our tuning experiments using INEX 2008 in section 4, and finally our results in the INEX 2009 competition in section 5.

2 Experimental protocol

We have used the INEX Ad-Hoc 2008 collection as a training collection, and the INEX 2009 collection as a test collection. INEX 2008 collection contains 659,388 XML articles extracted from the English Wikipedia in early 2006 [1] and 70 queries. INEX 2009 collection contains 2,666,190 XML articles extracted from the English Wikipedia on 8 october 2008 [8] and 115 queries. All the results presented here on 2008 collection were computed using the INEX 2008 evaluation programs: *eval_inex*, version 1.0.

Our evaluation is based on the main INEX measures (*iP*[x] the precision value at recall x , *AiP* the *interpolated average precision*, *MAiP* the *interpolated mean average precision* and *MAgP* the *generalized mean average precision* [3]). The main ranking of INEX competition is based on *iP*[0.01] instead of the overall measure *MAiP*, allowing to emphasize the precision at low recall levels.

Given that every experiment is submitted to INEX in the form of a ranked list of a maximum of 1,500 XML elements for each query, such measures favor, in terms of recall, the experiments for which whole articles are found (thereby providing a greater quantity of information for 1,500 documents). This is problematic in the case of Focused IR as more focused answers may be penalized even though it is the very purpose of Focused IR to be able to return better granulated answers (in the form of relevant elements, reduced from a whole article). Thus, we also calculated $R[1500]$, the recall rate for 1,500 documents, and $S[1500]$, the size (in Mb) of the 1,500 documents which were found.

3 System overview

Our system is based in the BM25 weighting function [7], that processes articles a_j as well as elements e_j :

$$w_{ji} = \frac{tf_{ji} \times (k_1 + 1)}{k_1 \times ((1 - b) + (b * ndl)) + tf_{ji}} \times \log \frac{N - df_i + 0.5}{df_i + 0.5} \quad (1)$$

with:

- tf_{ji} : the frequency of t_i in article a_j (resp. element e_j).
- N : the number of articles (resp. elements) in the collection.
- df_i : the number of articles (resp. elements) containing the term t_i .
- ndl : the ratio between the length of articles a_j (resp. elements e_j) and the average article (resp. element) length (*i.e.* its number of terms occurrences).

- k_1 and b : the classical BM25 parameters.

Parameter k_1 allows to control the term frequency saturation. Parameter b allows to set the importance of ndl , *i.e.* the importance of document length normalization (cf. equation 1). This is particularly important in focused IR as the length variation for elements is greater than that of articles, as each article is fragmented into elements (we set the minimum element length at 10 words, and the largest article contains 35,000 words).

Our system also considers some other parameters, *e.g.*:

- *logical_tags*: list of XML tags which the system will consider either at indexing time or during the query step (the system will therefore not be able to return an element that does not belong to this list);
- *minimum_size*: minimum size of documents (articles/elements) (# of terms);
- *level_max*: maximum depth of documents (depth of XML tree);
- *df*: df_i value for each term, computed on articles (INEX 2008: $max(df_i) = 659,388$); or on elements (INEX 2008: $max(df_i)$ between 1 and 52 millions);
- stop words: using a stop words list;
- parameters concerning queries handling: andish mode, mandatory or banned query terms (+/- operators).

4 Parameters tuning (INEX 2008)

4.1 System settings

All our runs have been obtained automatically, and using only the query terms (*i.e.* the *title* field of INEX topics). We thus do not use fields *description*, *narrative* nor *castitle*.

Several parameters have been studied for articles retrieval as well as for element retrieval. Some parameters were set after a few initial experiments, *e.g.*:

- *logical_tags* (articles retrieval): article;
- *logical_tags* (elements retrieval): article, li, row, template, cadre, normalist, section, title, indentation1, numberlist, table, item, p, td, tr;
- *minimum_size_terms*: 10 terms. Some analysis on the assessments (not presented here) have shown that it is unnecessary to consider elements smaller than 10 terms, because these small elements are either non-relevant or their father is 100% relevant (and in this case it is better to return the father, which is bigger and thus easier to index). Note that [4] have shown, using former INEX 2002 collection, that an optimal value for this parameter is to be set around 40.;
- *level_max*: 1 for articles retrieval, 23 for elements retrieval;
- *df*: computed on articles (resp. elements) while indexing articles (resp. elements), instead of computing an overall *df* (*e.g.* at article level) used while indexing articles as well as elements. Note that [9] compute an overall *df*.
- stop words: 319 words from Glasgow Information Retrieval Group²,

² List of 319 stop words from Glasgow Information Retrieval Group: http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words

Two important parameters were studied more thoroughly: b et k_1 , using a 2D grid: b varying from 0.1 to 1, with 0.1 steps, and k_1 varying from 0.2 to 3.8 with 0.2 graduations), thus a total of 380 runs (articles and elements retrieval).

4.2 INEX 2008 tuning results

Figure 1 presents the behavior of the classical IR (articles), showing the MAiP and $iP[0.01]$ changes according to the b (resp. k_1) values. For a given b value (resp. k_1 value), the $iP[0.01]$ and MAiP measures drawn are the ones obtained using the optimal k_1 (resp. b) values.

The best (b, k_1) values for classical IR are slightly higher for MAiP $((b, k_1) = (0.6, 2.2))$ than for $iP[0.01]$ $((b, k_1) = (0.4, 1.6))$. These values are not far from the classical values proposed in the literature (e.g. (0.7, 1.2)).

Fig. 1. Classical IR according to b and k_1

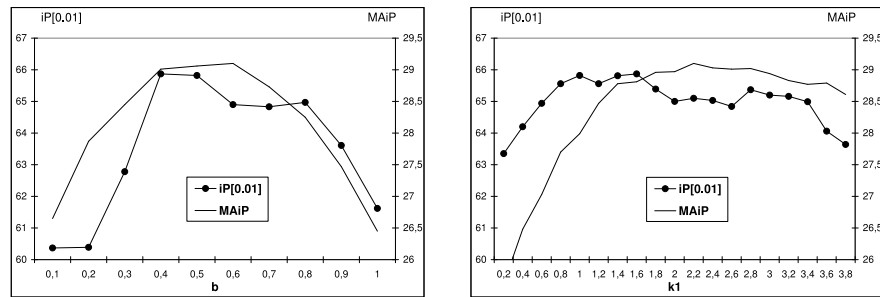
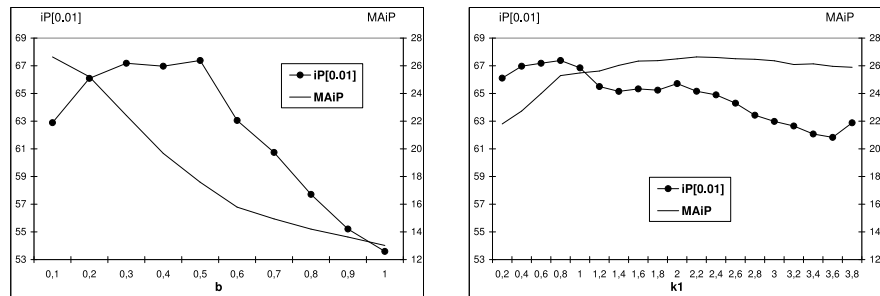


Figure 2 presents the behavior of the BM25 model in focused IR. The best (b, k_1) values are quite different for MAiP $((b, k_1) = (0.1, 2.2))$ than for $iP[0.01]$ $((b, k_1) = (0.5, 0.8))$. The best MAiP is reached with the minimum value $b = 0.1$. The length normalization of BM25 seems to be counterproductive while optimizing recall in focused IR. On the other hand, it is still useful while optimizing precision (best value: $b = 0.5$). The k_1 (tf saturation) seems to be less important for focused IR: either $iP[0.01]$ and MAiP slightly fluctuate with k_1 .

Fig. 2. Focused IR according to b and k_1



The results obtained with the optimal parameter configuration are presented in table 1 according to the $iP[0.01]$ criterion and to the $MAiP$ criterion.

Table 1. Evaluation of 380 runs with the $iP[0.01]$ and $MAiP$ criterion

Run	Granularity	Tags	b	k_1	$iP[0.01]$	#doc	#art	R[1500]	S[1500]
R1	Articles	-	0.4	1.6	0.6587	1,457	1,457	0.8422	8.22
R2	Elements	-	0.5	0.8	0.6738	1,463	1,257	0.4134	1.65
Run	Granularity	Tags	b	k_1	$MAiP$	#doc	#art	R[1500]	S[1500]
R3	Articles	-	0.6	2.2	0.2910	1,457	1,457	0.8216	6.15
R4	Elements	-	0.1	2.2	0.2664	1,459	1,408	0.7476	5.24

5 INEX 2009 results

We present in this section the official results obtained by our system during INEX 2009 on the new INEX 2009 collection. We submitted 17 runs: 5 runs to the Focused task and 4 runs to the Best In Context, the Relevant In Context and the Thorough tasks. One run on each task is based on the BM25 reference run given by the INEX organizers.

5.1 System settings

All our runs have been obtained automatically, and using only the *title* field of INEX topics. Most of the settings given in section 4.1 have been reused for our INEX 2009 runs, except:

- *logical_tags* (elements retrieval): article, list, p, reflat, sec, ss1, ss2, ss3, ss4, table, template (manually chosen);
- b and k_1 : 0.6 and 2.2 for articles retrieval (in order to maximize $MAiP$);
- b and k_1 : 0.5 and 0.8 for elements retrieval (in order to maximize $iP[0.01]$);
- $level_{max}$: 1 for articles retrieval, 100 for elements retrieval;
- df : computed on articles while indexing articles (INEX 2009: $max(df_i) = 2,666,190$) and computed on elements while indexing elements (INEX 2009: $max(df_i) = 444,540,453$).

5.2 Results: Focused task

Table 2 presents the official results of our runs, compared to UWFERBM25F2 (Waterloo University) which was the winning run for the Focused task.

Our system gives very interesting results compared to the best INEX systems. The classical IR achieves better results in terms of precision: $iP[0.01] = 0.6060$ by UJM.15525, against focused IR: 0.5136 (UJM.15518). Moreover, we think

Table 2. Official evaluation of 57 "Focused" task runs

Run	Granularity	Reference run	b	k_1	$iP[0.01]$	Rang
UWFERBM25F2	Element	-	-	-	0.6333	1
UJM_15525	Article	-	0.6	2.2	0.6060	6
UJM_15479	Article	-	0.6	2.2	0.6054	7
UJM_15518	Element	yes	0.5	0.8	0.5136	36
UJM_15484	Element	-	0.5	0.8	0.4296	45

that the classical IR superiority should be confirmed by *MAiP* ranking. This confirms the results obtained during INEX 2008.

It is interesting to see that the BM25 model applied on full articles (UJM_15525 and UJM_15479) outperforms our focused retrieval results (UJM_15518 and UJM_15484) considering $iP[0.1]$, despite the fact that BM25 parameter `nd1` is designed to take into account different documents lengths and thus documents granularities.

5.3 Relevant In Context, Best In Context and Thorough

Our BIC, RIC and Thorough runs have not been computed specifically. We have reordered and filtered some focused runs, in order to consider the RIC, BIC and Thorough order and coverage rules. The following tables present our results to these tasks.

Table 3. Official evaluation of 37 "Best In Context" task runs

Run	Granularity	Reference run	b	k_1	$MAgP$	Rang
BM25bepBIC	Element	-	-	-	0.1711	1
UJM_15490	Element	UJM_15479	0.5	0.8	0.0917	28
UJM_15506	Element	UJM_15479	0.5	0.8	0.0904	30
UJM_15508	Element	yes	0.5	0.8	0.0795	34

Table 4. Official evaluation of 33 "Relevant In Context" task runs

Run	Granularity	Reference run	b	k_1	$MAgP$	Rang
BM25RangeRIC	Element	-	-	-	0.1885	1
UJM_15502	Element	UJM_15479	0.5	0.8	0.1075	21
UJM_15503	Element	yes	0.5	0.8	0.1020	26
UJM_15488	Element	UJM_15479	0.5	0.8	0.0985	27

Runs UJM_15488 and UJM_15490 have been filtered with our best article run (UJM_15479), while UJM_15500, UJM_15502 and UJM_15506 have been filtered and re-ranked with the same article run (UJM_15479).

Table 5. Official evaluation of 30 "Thorough" task runs

Run	Granularity	Reference run	b	k_1	$MAiP$	Rang
LIG-2009-thorough-3T	Element	-	-	-	0.2855	1
UJM_15494	Element	yes	0.5	0.8	0.2435	9
UJM_15500	Element	UJM_15479	0.5	0.8	0.2362	12
UJM_15486	Element	-	0.5	0.8	0.1994	17

6 Conclusion

Our run UJM_15525 is ranked sixth of the competition according to the $iP[0.01]$ ranking. That means that a basic BM25 article retrieval run (classical IR) gives better "precision" results ($iP[0.01]$) than BM25 element retrieval (focused IR), and should also give better "recall" results ($MAiP$).

These results confirm that article retrieval gives very good results against focus retrieval (as in 2008), even considering precision (that was not the case in 2008). However, we don't know if it comes from BM25, which is perhaps not suitable for elements indexing, or if it comes from a wrong parameters settings. It is perhaps not so easy to reuse settings of parameters tuned on a different collection. We have to experiment more deeply on 2009 collection, using the same 2D grid for b and k_1 , but also varying other parameters in order to check where the results come from.

References

1. Ludovic Denoyer and Patrick Gallinari. The wikipedia XML corpus. In *SIGIR forum*, volume 40, pages 64–69, 2006.
2. Mathias Géry, Christine Langeron, and Franck Thollard. Ujm at inex 2008: Pre-impacting of tags weights. In *INEX*, pages 46–53, 2008.
3. J. Kamps, J. Pehcevski, G. Kazai, M. Lalmas, and S. Robertson. INEX 2007 evaluation measures. In *Focused access to XML documents, INEX, 2007*.
4. Jaap Kamps, Maarten de Rijke, and Börkur Sigurbjörnsson. The importance of length normalization for XML retrieval. *Inf. Retr.*, 8(4):631–654, 2005.
5. Jaap Kamps, Shlomo Geva, Andrew Trotman, Alan Woodley, and Marijn Koolen. Overview of the inex 2008 ad hoc track. In *INEX*, pages 1–28, 2008.
6. Wei Lu, Stephen E. Robertson, and Andrew MacFarlane. Field-weighted xml retrieval based on bm25. In *INEX*, pages 161–171, 2005.
7. S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *JASIST*, 27(3):129–146, 1976.
8. Ralf Schenkel, Fabian M. Suchanek, and Gjergji Kasneci. Yawn: A semantically annotated wikipedia xml corpus. In Alfons Kemper, Harald Schning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, and Christoph Brochhaus, editors, *GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web*, volume 103 of *LNI*, pages 277–291. GI, 2007.
9. M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges. Optimisation methods for ranking functions with multiple parameters. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 585–593, New York, NY, USA, 2006. ACM.

A Method of Generating Answer XML Fragment from Ranked Results

Atsushi Keyaki¹, Jun Miyazaki², and Kenji Hatano³

¹ Graduate School of Culture and Information Science, Doshisha University
1-3 Tatara-Miyakodani, Kyotanabe, Kyoto 610-0394, Japan

`keyaki@ilab.doshisha.ac.jp`,

² Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan

`miyazaki@is.naist.jp`,

³ Faculty of Culture and Information Science, Doshisha University
1-3 Tatara-Miyakodani, Kyotanabe, Kyoto 610-0394, Japan

`khatano@mail.doshisha.ac.jp`,

Abstract. In this paper, we propose a method which can obtain more appropriate results for XML search. Since our previous approach has a problem that large-sized XML fragments tend to have large scores in CO topics, it is not suitable for users to return the ranked results in the descendant order of their scores. To cope with this problem, we generate a final search result of XML fragments by integrating smaller-sized ones with higher scores. We apply our approach to the probabilistic scoring model which is often used in the field of XML search area.

1 Introduction

XML (Extensible Markup Language) is one of the structured document formats, which becomes the de facto standard data exchange format. For this reason, a number of XML documents have recently been produced. This surely continues even in the future, and more and more XML documents would be produced. In this situation, information retrieval techniques on XML documents become very important and are strongly expected.

There are so many researches on XML. Focusing on XML search, there are two goals to retrieve XML documents. One is efficient search. In contrast to traditional text document search, XML search needs to retrieve results from more candidates than the traditional one does. This is because an XML document can be treated as a set of documents surrounded by each tag. In other words, a unit of retrieval in XML search is smaller than that in text document search. The unit of retrieval is usually called an *XML fragment*. The purpose of the efficient search is to reduce processing time, hardware requirement, etc. Several attempts of the efficient search could obtain good results. Lowest Common Ancestor, LCA for short [7], is one of the well-known approaches. An LCA is defined as a sub-tree in which all query keywords are contained and its height is the lowest. If we assume that LCAs are the best answer to a given query, the query

can be processed efficiently. However, the problem is that LCAs are not always considered to satisfy user’s information needs.

Recently, in addition to the efforts of efficient search, effective search which is the other goal has actively been studied. Users want to obtain useful results rather even if it spends more time than useless ones faster. We have previously proposed a scoring method for effective search. In this method, we combined two different scores: term weighting and query weighting scores. This scoring method could bring higher precision than one we proposed before. However, we found that the precision of CO topics is still low. In order to investigate the problem of our past approach, we first conduct preliminary experiments, and then, propose a new method to improve the precision of CO topics.

This paper is organized as follows. In Section 2, we explain preliminary experiments in detail. In Section 3, we propose a new method based on the experimental results in Section 2. Related work is introduced in Section 4, followed by concluding remarks and future work in Section 5.

2 Preliminary Experiments

In this section, we investigate why our previous approach brought different effects on the precision between CAS and CO topics.

First, we calculate the average text size of search results ranked by our past scoring method for the Focused task by using several queries. In our past approach, we sum up the following three scores: TF-IPF which is an extension of the TF-IDF term weighting technique, TF-IAF which is a scoring method of query structure, and query content score, QCS for short, which considers the constituent rate of the query keywords in an XML fragment. Considering the QCS, higher score is given to a larger-sized XML fragment. The reason is that the more an XML fragment contains kinds of query keywords, the higher query content score becomes. This property is quite different from TF-IPF [1] and TF-IAF [4], which rarely produce large-sized XML fragments as a search result. Therefore, we compared the text size of the XML fragments which have high score for each scoring method. As a result, it turned out that the size of XML fragments is large if we consider the QCS (see Table 1). This tendency is observed more remarkably in CO topics, though it appears both CAS and CO topics. This is because CAS queries return only the XML fragments satisfying their structural constraints even if large XML fragments have higher score. On the other hand, it is possible that such large fragments become a search result of CO queries which do not have structural constraints. Therefore, it is considered that the text size of XML fragments becomes larger for CO queries. Of course, the XML fragments with large text size might have higher score even in CAS topics if they meet structural constraints. We suppose that the precision becomes higher if we can exclude the XML fragments which have large text size but low score. In order to explore this hypothesis in detail, we conduct some experiments to show the precision in Relevant in Context and Best in Context tasks for CAS, CO, and their mixed queries. The summary of the results is shown as follows.

- In Focused task which retrieves a set of the best single relevant XML fragment from one document to a given query, CAS was more effective than CO.
- In Relevant in Context task which is allowed to retrieve multiple relevant XML fragments from one document, CO is more effective than CAS.
- In Best in Context task which extracts the closer start point to a relevant fragment, CO is more effective than CAS. This tendency was observed more clearly than Relevance in Context task.

From these results, it turns out that our past approach shows higher precision in Relevant in Context task than the Focused one for CO topics. As discussed earlier, the possibility that the retrieved XML fragments have extremely large text size increases. On the other hand, the precision of CAS topics is comparatively low. Hence, it is not always true that retrieving XML fragments with large text size results in low precision.

We first obtain relevant XML fragments to a given query by using our past scoring method, called a ranked result. Then, we reconfigure them to present the best *Answer XML Fragments*, which are a final search result. Each Answer XML Fragment is a list of tree-structured relevant XML fragments in a document, which is constructed by integrating the score of the ranked result, instead of directly calculating the score of Answer XML Fragments. This is because XML fragments in a document are often indirectly dependent each other. Therefore, it is not suitable to directly calculate the score of XML fragments. As we see, this approach is to improve the precision in Relevant in Context and Best in Context tasks. We explain further detailed our methodology in the next section.

	TF-IPF	TF-IAF	QCS
All	1.00	1.00	1.39
CAS	1.00	1.00	1.36
CO	1.00	1.00	1.45

Table 1. Comparison of text size

3 Generating Answer XML Fragments

In this section, we explain our method in detail.

In order to determine a search result, we perform the following two steps. The first step scores each XML fragment and produces ranked results, which makes use of our past approach of XML fragment search. The second step generates Answer XML Fragments as a final search result, after the ranked XML fragments acquired in the first step are grouped by their document ID and reconfigured. The overview of our method is shown in Fig.3.

We explain the second step in more detail. XML search systems usually calculate scores for fragments, which are regarded as independent each other

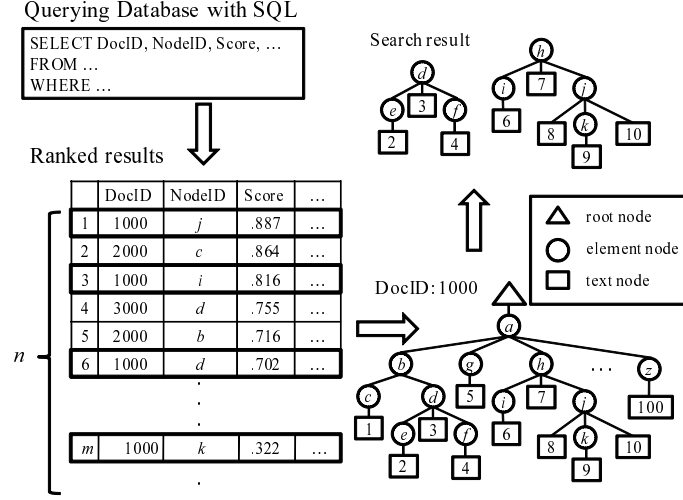


Fig. 1. Processing flow of our method

even in a XML document. Therefore, the document IDs in the ranked results are usually interleaved. Since the objective of our approach is not to show the ranked fragments judged as relevant in the descendant order of their scores but to present ideal fragments as Answer XML Fragments, we arrange the ranked fragments so that they are grouped by document ID. We explain its procedure to generate Answer XML Fragments by using an example shown in Fig.3.

In order to generate a final search result, the ranked fragments grouped by document ID and sorted in the descendant order of the scores are inserted into an *Answer XML Fragment* as long as the size of the Answer XML Fragment is less than a threshold, called *Extraction Limit*, or *EL* for short. The *EL* which is defined as the following expression is utilized so that the text size of fragments does not become too large.

$$EL = \alpha \cdot \text{textsize}(\text{root}) \quad (1)$$

Note that α is an indicator how much ratio an XML document has relevant fragments. We set α to $\frac{1}{3}$ in the example. Since $\text{textsize}(\text{root})$ is identical to the text size of the XML document, which is 300 in this case, *EL* is 100.

We also define connecting cost, *CCost*, as the distance between the node to be inserted and its nearest node in an Answer XML Fragment, where the distance is path length between two nodes. If the connecting cost is less than a threshold, called *Connection Limit* (*CL*), two nodes are joined. The reason why we join nodes is that relevant fragments can often exist around the ones with high score to a given query. Though these ones do not always have higher score, joining nodes might be able to find such relevant fragments which could not be

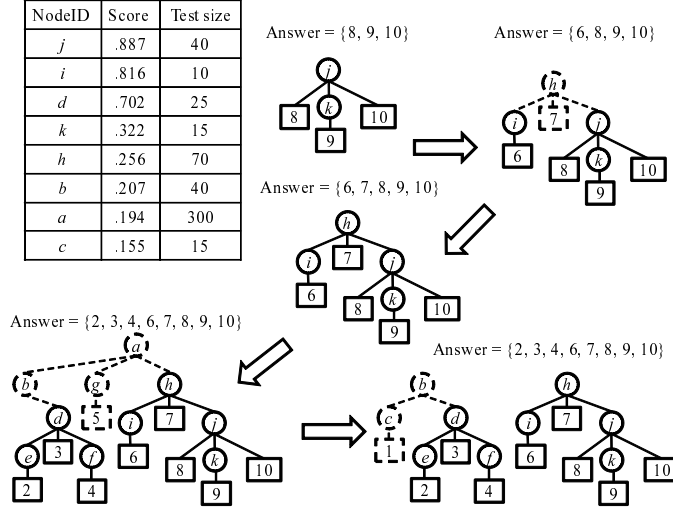


Fig. 2. Generating an Answer XML Fragment

explored before. The CL is a threshold to decide if the distance between two nodes is short enough, that is, whether two nodes are closely related each other or not.

For example, in Fig.3, the node with the highest score, j , can first be inserted into an Answer XML Fragment. This is because the total text size of the text nodes in the Answer XML Fragment, $\{8, 9, 10\}$, is 40, which is less than EL . Next, the node with the second highest score, i , is inserted into the Answer XML Fragment. Though the text nodes in the Answer XML Fragment become $\{6, 8, 9, 10\}$, the total size, i.e., 50, is still less than EL .

At this time, if node i is the nearest from a node in the already inserted into the Answer XML Fragment (node j in this case) and the $CCost$ of node i is less than CL , then these two nodes are joined. If we suppose that CL is 3, the connecting cost, $CCost(i, j)$, is 2, because there is a path from node i to node j by way of node h . Therefore, we can join nodes i and j . As a result, the root node of the Answer XML Fragment becomes node h , which includes text nodes $\{6, 7, 8, 9, 10\}$ and its total text size becomes 70.

After that, node d is inserted into the Answer XML Fragment, because the total text size (95) is still less than EL even if node d is added. Though $CCost(c, h)$ is 3, no join is performed because the total text size exceeds EL if node d is joined. We skip the insertion of nodes k and h , because they have already been added to the Answer XML Fragment. The following insertions of nodes b , a , and c are not performed because the text size is over EL . Hence, we finally obtain the Answer XML Fragment whose root nodes are d and h .

Note that it frequently occurs that a node has only one internal node as a child in the INEX test collection that we use for evaluation, like the fragments on the left and in the middle of Fig.3. Therefore, we calculate $CCost$ only when a node has one or more text nodes in its descendant.

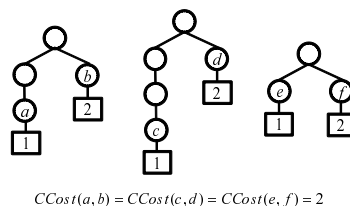


Fig. 3. Connecting Cost

When a final search result is actually returned, the ranked results which can be integrated together are merged as Answer XML Fragments as much as possible⁴. When the number of the ranked fragments being obtained is small, the parameters EL and CL should be smaller. As the ranked fragments are being produced, these parameters should be increased to obtain more number of relevant fragments. In this way, upper ranked results are extracted so that the precision of a final result becomes higher, while lower ranked ones are obtained for higher recall.

4 Related Work

As briefly explained in Section 1, finding LCAs in XML documents is often utilized for searching results related to user's queries in XML search. Since the concept of the LCA is very intuitive, it has widely been adopted in the research field of XML keyword search. For the initial period of time, many researchers proposed methods for efficient searching LCAs in XML documents such as SLCA (Smallest Lowest Common Ancestor) [8], VLCA (Valuable Lowest Common Ancestor) [5], and MCT (Minimum Connecting Tree) [2]. Their contribution was efficient query processing for searching LCAs in XML documents; however, LCA and its extension cannot achieve high precision search, and are not effective in XML search [4]. Therefore, there are also other researches for searching XML fragments based on the concept of LCA.

MLCA (Meaningful LCA) [6] is one of the most famous approach for searching XML fragments whose leaf nodes are closely related each other. The concept of MLCA is very similar to that of the LCA; however, the leaf nodes matching

⁴ The INEX evaluation tool accepts only top 1500 results. Therefore, this step needs only to reduce the number of results.

query keywords are meaningfully related in the case of MLCA while ones are not related in the case of LCA. eXtract [3] can also extract another versions of LCA to process MLCA for presenting an appropriate searching paper, we proposed a method to generate relevant searching results from the ranked XML fragments obtained by using our previously proposed technique.

5 Conclusion

In this paper, in order to solve the problem that large-sized XML fragments tend to obtain large scores in our past approach, we proposed a new method which generates Answer XML Fragments as a search result, which also uses our past scoring scheme. During generating a search result, we used two parameters: EL which restricts the size of fragments extracted, and CL which determines whether related two fragments should be joined or not.

5.1 Future Work

In the future, we need to verify the potential power of our proposed method through some experiments. More precisely, we will compare the precisions of TF-IAF and Answer XML Fragment based approaches. We also need to compare the average size of the fragments in a set of Answer XML Fragments and the ones grouped by each document ID by using TF-IAF to show how much we can reduce the size of result fragments.

The proposed method is supposed to require additional processing cost to reconfigure ranked fragments. Therefore, efficient query processing, which is also taken into account for our previous work, is also one of the important issues to be solved.

Acknowledgements

This work was supported in part by MEXT (Grant-in-Aid for Scientific Research on Priority Areas #20700227 and #21013035).

References

1. Torsten Grabs and Hans-Jörg Schek. PowerDB-XML: A Platform for Data-Centric and Document-Centric XML Processing. In *Proceedings of the First International XML Database Symposium*, volume 2824 of *Lecture Notes on Computer Science*, pages 100–117. Springer-Verlag, September 2003.
2. Vagelis Hristidis and Nick Koudas. Keyword proximity search in xml trees. *IEEE Transaction on knowledge and data engineering*, 18(4):525–539, April 2006.
3. Yu Huang, Ziyang Liu, and Yi Chen. Query Biased Snippet Generation in XML Search. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 315–326, June/July 2008.

4. Atsushi Keyaki, Kenji Hatano, and Jun Miyazaki. A scoring method of xml fragments considering query-oriented statistics. In *Proceedings of the Second International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2009)*, pages 473–478, August 2009.
5. Guoliang Li, Jianhua Feng, Jianyong Wang, and Lizhu Zhou. Effective keyword search for valuable lcas over xml documents. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 31–40, New York, NY, USA, 2007. ACM.
6. Ziyang Liu and Yi Chen. Identifying Meaningful Return Information for XML Keyword Search. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 329–340, June 2007.
7. Albrecht Schmidt, Martin Kersten, and Menzo Windhouwer. Querying xml documents made easy: Nearest concept queries. In *17th International Conference on Data Engineering (ICDE'01)*, page 321, April 2001.
8. Yu Xu and Yunnis Papakonstantinou. Efficient Keyword Search for Smallest LCAs in XML Databases. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 527–538, June 2005.

ENSM-SE at INEX 2009 : scoring with proximity and semantic tag information

Amelie Imafouo², Annabelle Mercier¹ and Michel Beigbeder²

¹ LCIS Lab - Grenoble University
50 rue Barthelemy de Laffemas
BP 54 - 26902 Valence Cedex 9, France

² École Nationale Supérieure des Mines de Saint-Étienne
158 cours Fauriel
F-42023 Saint Etienne Cedex 2, France

Abstract. We present in this paper experiments on the Wikipedia collection used in the INEX 2009 evaluation campaign with an information retrieval method based on proximity. The idea of the method is to assign to each position in the document a fuzzy proximity value depending on its closeness to the surrounding keywords. These proximity values can then be summed on any range of text – including any passage or any element – and after normalization this sum is used as the relevance score for the extent.

1 Introduction

Within the context of structured documents, the idea of adhoc retrieval is a generalization of what it is in the flat document context: searching a static set of documents using a new set of topics, and returning arbitrary XML elements or passages instead of whole documents. In *Focused Retrieval* as stated in the INEX evaluation campaigns, the returned parts or elements of documents in response to a user query must not overlap. For the full document retrieval task, with all the popular models the documents and the queries are represented by bags of words. The ranking scores are computed by adding the contributions of the different query terms to the score of one document.

With the bag of words representation the positions of words in the documents are not used for scoring. While the absolute position of terms does not seem to be significant for scoring, we can have some intuition that the relative positions of the query terms could be relevant. For instance with a query with the two words **roman** and **architecture**, one occurrence of each of these terms in a document gives the same contribution to the score either these two occurrences are close or not in a document. Taking into account the relative positions at least involves that two query terms does appear in a document so that the proximity component does not score to zero.

Section 2 is devoted to a brief state of the art of techniques and models used in flat information retrieval to model the proximity of query term occurrences in the score of documents. Then our scoring model for flat

documents, its extension for hierarchically structured documents and the use of semantic tags are presented in section 3. Experiments with this model are detailed in section 5.

There is few usage of proximity in information retrieval.

2 A brief state of the art of proximity usage

One of the first use of term position appeared within implementations of the boolean model with the NEAR operator. But there are two main problems with such operators, the first one is that the semantics of these operators is not clean and it leads to some inconsistency problems as it was noticed by [1]. The second one is that they are stuck to boolean retrieval: documents verify or not the query and there is no ranking.

As suggested by the example query with the two terms **roman** and **architecture**, there are relationships between the use of proximity and indexing/ranking with phrases. Thus one highly tested idea is to discover phrases in the corpus and then to index the documents with both these phrases and the usual single terms. The conclusion [2] is that this method does not improve retrieval effectiveness except for low quality retrieval methods.

In the phrase discovery works, phrases are only looked for with adjacent words and only some of them are then retained for both indexation and scoring. To relax this constraint about phrases a proposition is to compute the score of a document as the sum of two scores [3], the first one is the usual Okapi BM25 score, and the second one is the proximity score. Later reinvestigated [4], the conclusion was that proximity use is more useful as documents are longer and as the collection is larger.

Another idea to take into account the term proximity with the Okapi BM25 framework circumvents the problem of coherence between word scoring and relaxed phrase scoring [5, 6].

As a synthesis of all these experiments we can derive a conclusion that it is better to consider several terms of the query in the proximity consideration. A second intuition is that their co-occurrence must be considered in a relaxed way by accepting that a quite large number of other words could intermix with the query terms. This second conclusion was also formulated by [7]. She experimented a quite simple method with a usual vectorial ranking but the results were then filtered through a conjunctive filter which impose that all the query terms occur in a returned document and in a more strict version that they occur in a passage shorter than 100 to 300 words.

To take into account proximity as phrase relaxation with a continuous paradigm two ideas were experimented. The first one was developed for the TREC 1995 campaign by two independant teams [8, 9]. Intervals that contain all the query terms are searched. Each selected interval receives a score, higher as the interval is shorter. [10] later experimented their method and concluded that it is quite beneficial for short queries and for the first recall levels.

The last idea is based on the influence that each query term occurrence exercises on its neighbouring [11]. Each position in the text receives an

influence from the occurrences of the query terms, and all these influences are added. Then either the maximum or the summation of this function is the score of the document. This influence idea was reused in a boolean framework, mostly used in a conjunctive way[12]. This last method is developed in the next section as it is the basis for the model used in the structured context.

3 Fuzzy proximity model

3.1 Fuzzy proximity model for flat documents

As in any other information retrieval model, the textual documents are represented with the terms that occur in them. In the sequel, we will call T the set of terms appearing in the collection.

We want to represent each document with the positions of the term occurrences. This can easily be achieved by representing a document d with a finite sequence over \mathbb{N} , the set of positive integers. We will also call d this sequence:

$$\begin{aligned} d : \mathbb{N} &\rightarrow T \\ x &\mapsto d(x) \end{aligned}$$

and $d(x)$ is the term that appears at position x in the document d . With this notation $d^{-1}(t)$ is the set of positions where the term t occurs in the document d , and in a positional inverted file this set is sorted and represented by the list of postings for the term t in the document d .

A collection is a set of documents. Figure 1 displays an example of a collection of four documents (d_0 to d_3) where only two different elements of T , A and B , are showed. In this example, we have for instance: $d_3(2) = A$, $d_3(3) = A$, and $d_3^{-1}(A) = \{2, 3\}$.

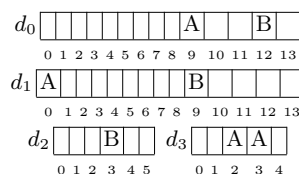


Fig. 1. Example of a collection C , A and B are some elements of T .

Fuzzy proximity to a term Instead of trying to define a proximity measure between the query terms by taking into account the position of the term occurrences as it is done for instance by the length of intervals containing all the query terms, our approach defines a proximity to the query at each position in the document.

Thus the first step is to define a (fuzzy) proximity between a position in the text and one term. Formally, given some $t \in T$, we define $\mu_t^d : \mathbb{Z} \rightarrow [0, 1]$ with

$$\mu_t^d(x) = \max_{i \in d^{-1}(t)} \left(\max \left(\frac{k - |x - i|}{k}, 0 \right) \right),$$

where k is some integral parameter which controls to which extent one term occurrence spreads its influence. The function μ_t^d reaches its maximum (the value 1) where the term t appears and it decreases with a constant slope down to zero on each sides of this maximum. In other terms, this function has a triangular shape at each occurrence of the term t . Fig. 2.a shows $(\mu_A^d)_{d \in C}$ and $(\mu_B^d)_{d \in C}$ for the collection C shown in Fig. 1, with k set to 4.

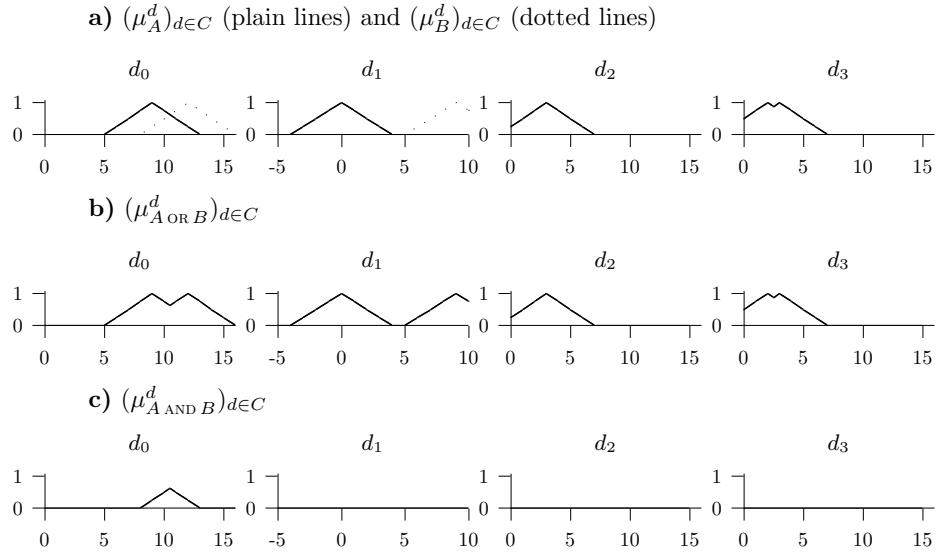


Fig. 2. Different $(\mu_q^d)_{d \in C}$ for the collection C of Fig. 1 with $k = 4$.

This function can be interpreted as the membership degree of a text position x in the document d to a fuzzy set $P(d, t)$.

Fuzzy proximity to a query We will now generalize the fuzzy proximity to a term to a fuzzy proximity to a query. Again it is a local proximity as it is defined for each position in the document. We use boolean queries, which thus can be represented as trees. The leaves of these trees are the terms that appear in the query, and the internal nodes are boolean operators: AND, OR and NOT.

The functions μ_t^d defined in the previous section are associated to the leaves of the query tree. Considering a conjunctive node with two terms A and B , we want to measure for each position how close it is from both terms. This can easily be achieved by using a formula for computing the degree of membership of an intersection in the fuzzy set theory. They are known as *t-norms*. In our experiments we used the minimum t-norm: $\top_{\min}(a, b) = \min(a, b)$. For a disjunctive node, we used the complementary co-norm $\mu_{q_1 \text{ OR } q_2}^d = \max(\mu_{q_1}^d, \mu_{q_2}^d)$. Finally for a complement node, we define $\mu_{\text{NOT } q}^d = 1 - \mu_q^d$.

Thus with these definitions μ_q^d is defined for every boolean query q . In the implementation, these formulas are recursively applied with a post-order tree traversal. The result is the function at the root of the tree, which we call *local fuzzy proximity to the query*. This function can be interpreted as the membership degree of the text positions in the document d to the fuzzy set $P(d, q)$. With a purely conjunctive query, this function have higher values as all the query terms are close to a given position.

Fig. 2.b (resp. Fig. 2.c) plots $\mu_{A \text{ OR } B}^d$ (resp. $\mu_{A \text{ AND } B}^d$) for the documents of the collection C of Fig. 1. Note that the function $\mu_{A \text{ AND } B}^{d_1}$ is uniformly zero, though the document d_1 contains both the terms A and B because their occurrences are not close enough.

Score of documents and passages With the local proximity μ_q^d defined in the previous section it is easy to define a global proximity of any range of positions, either for a full document or for any passage between the positions x_1 and x_2 with

$$\sum_{x_1 \leq x \leq x_2} \mu_q^d(x)$$

To take into account the specificity of the range to the query we then normalize this score by the length of the passage, and finally the score of a passage p between the positions x_1 and x_2 is

$$s(q, p) = \frac{\sum_{x_1 \leq x \leq x_2} \mu_q^d(x)}{x_2 - x_1 + 1}.$$

3.2 Fuzzy proximity model for structured documents

Given the proximity model presented in the previous section, we will now deal with its extension to the structured case. We just deal with the hierarchical aspect of structure. In this hierarchy, the components are the nested sections and their titles. So, we have to model the influence of an occurrence of a query term by taking into account in which type of elements it appears in, either in a section-like element or in a title-like element.

For a term occurrence which appears in a section-like element, the basis is the same as in flat text: A decreasing value in regards to the distance to the occurrence. We add another constraint, the influence is limited to the section in which the occurrence appears.

For term occurrences which appear in title-like elements their influence is extended to the full content of the element and recursively to the elements contained in it. Here the assumption is that the title is descriptive of the content of the section it entitles. In the proximity paradigm, any title term in a title-like element should be close to any term occurrence in the corresponding section. So our choice is that the influence of a title term is set to the maximum value (value 1) over the whole section.

For a single term some of term influence area (triangles) can overlap and should giving the “mountain” aspect in the document representation. Truncated triangles can also be viewed, which indicates that the influence of an occurrence of the term has been limited to the boundaries of the section-like element it belongs to. Finally, rectangles can be seen where an occurrence appears in a title-like element and the influence was uniformly extended to the whole bounding section-like element.

Once the proximity function is computed for a document at the root of the query tree, to answer to the focused task we have to select some elements or some passages and compute their relevance value. The last step consists in choosing some of them so that the non overlapping constraint is verified. Then for a given document an iterative algorithm is applied to the section-like elements. Those elements that are relevant (i.e. their relevance score is not zero) are sorted according to their score in decreasing order. The top element is selected and inserted in the output list. All its descendant and all the elements that belongs to its path to the document tree root are disabled in order to avoid overlapping elements in the output. Then the next element with the highest score and not disabled is chosen and we repeat the process until all the elements are disabled. When all the documents have been processed, the output list is sorted by decreasing relevance score.

3.3 Fuzzy proximity model with semantic tags

In the INEX 2009 Wikipedia collection, the documents are annotated with the 2008-w40-2 version of YAGO. This collection includes semantic annotations for articles and outgoing links, based on the WordNet concepts YAGO assigns to Wikipedia articles. YAGO explicitly labels more than 5800 classes of entities like persons, cities, movies and many more. The proximity model for structured documents presented previously was extended to take these semantic tags into account. The terms contained in the name of a semantic tag are taken into account and the text inside the semantic tag also. The terms contained in the name of a semantic tag are added to the article text (within a new tag `<t>`).

A new operator `<>` is introduced that helps using the semantic tags for the desambiguation of queries. It takes two operands, the first one is a semantic tag (or a list of semantic tags linked by the OR operator) and the second operand is a term (or a combination of terms linked by operators like AND or OR). It works nearly like the AND operator explained previously, but the second operand is considered only if it is in the context of the first operand. This means that the occurrences terms (or the combination of occurrences terms) forming the second operand is considered only if it appears inside the semantic tag (or one of the

semantic tags) forming the first operand. The notation used for a query $op1 \langle \rangle op2$ is $\langle op1 \rangle op2$ for more convenience. For example, let us consider the query $ST \langle \rangle A$ that we choose to note like this $\langle ST \rangle A$, where ST is a semantic tag and A a term. For a given document, all the occurrences of A not appearing in a semantic tag ST are not considered. If an occurrence of A appears inside the semantic tag ST , then it is taken into account as previously : a decreasing value in regards to the distance to the occurrence is computed, the influence of the occurrence term A for this query is limited to the semantic tag ST .

For the semantic tags names with many terms, (e.g. $\langle \text{ethnic group} \rangle$), all the terms must appear in the first operand of a query, otherwise it will not be considered. e.g. : semantic tag $\langle \text{ethnic group} \rangle$, a query like $\text{group} \langle \rangle \text{BigBand}$ (written $\langle \text{group} \rangle \text{BigBand}$ for more convenience) will be scored zero. There is two query examples above.

Query 1

```
((ST1 OR ST2)⟨⟩ A) noted
⟨ST1 | ST2⟩ A is equivalent to
⟨ST1⟩A OR (⟨ST2⟩A)
```

Query 2

```
⟨ST1 | ST2⟩ (A AND (B OR C))$ is equivalent to
((⟨ST1⟩(A AND B)) OR (⟨ST1⟩(A AND C)) OR
(⟨ST2⟩(A AND B)) OR (⟨ST2⟩(A AND C)))
```

4 Document collection transformations

4.1 The semantic tags names transformations

Among the tags provided in the INEX 2009 Wikipedia collection, many needed some transformations to be usable in queries. The first transformations made was the character conversion (*to e*, *to O*, *etc.*). Some tags contain some string showing it was generated from an URL (*http*, *www*, *etc.*) these strings were removed from the tags. The characters like $_$, and the numbers were replaced by the space character, many spaces characters were replaced by one and all the spaces characters at the beginning or at the end of a tag were removed. Here are some examples of initial tags and their transformations : `fictional_character` becomes `fictional character`

4.2 The semantic tags names adding to document text

The YAGO tool adds some semantic information, for example, "Mickey 3D" can be tagged as a $\langle \text{musical group band} \rangle$. For each semantic tag, we add in the text the content of the tag. For the previous example, the text indexed is `" $\langle \text{musical group band} \rangle$ Mickey 3D musical group band".` For several queries, we take into account the possibility to take profit of these tags with (1) the tag itself and (2) the

content from the tag. For example, the query about roman architecture can be extended to ("roman architecture" | "roman building" | "roman architect") | <architect | architecture | architectural> roman

5 Experimentation results and conclusion

For the INEX 2009 campaign, we submit runs given in figure 3. Experimental runs are based on variation of three parameters :

- *Length of k.* The parameter k controls the influence area of a term. The measure unit of k is a number of term. We choose $k = 200$ to match to the length of a paragraph and $k = 1000$ to match to the length of a section or a whole document.
- *Stemming or not.* The collection is indexed with stemming and no stemming. We build also queries with manual stemming that is to say using lexical forms of query terms and synonyms.
- *Propagation or not.* The propagation technique explained in section 3.2 is used for several runs.

#	$iP[0.01]$	Institute	Run	features
21	0.5844	22	emse2009-150	$k = 200$, no stem, title propagation
25	0.5733	22	emse2009-153	$k = 1000$, no stem, propagation
35	0.5246	22	emse2009-151	$k = 200$, stemming, propagation
47	0.3360	22	emse2009-152	$k = 1000$, no stem, without propagation

Fig. 3. Runs submitted for focused task with proximity based method

The best run uses title propagation, no stemming and $k=200$. We can note that the increment of the k value doesn't improve the results. The worst result is done with no stemming, no propagation and the k value too long.

To optimize performances, we have to choose an adapted value of k . With a high value, the model behaviour approximates the behavior of the boolean model ; at the opposite, with a small value, conjunctive queries should give no result. So to retrieve and rank documents with proximity method, we can use a *middle* k value which can represent the length of a paragraph.

We haven't enough runs to compare the stemming effect but we can note that title propagation improves more the results than the stemming (cf. runs 150 and 151).

By comparing run 153 and run 152, we can see the effect of propagation : with k assign to 1000, using propagation increase the result of seventy percent.

As attended, proximity method gives good results at the high recall levels but compared to the other methods (see figure 4), the proximity method has to improve global results at the other levels.

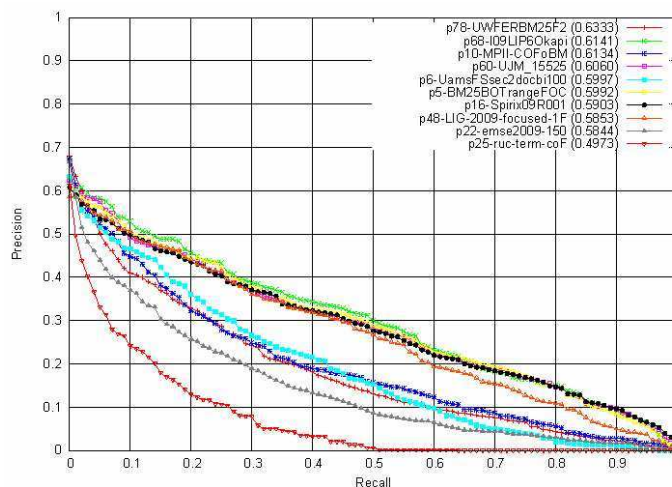


Fig. 4. Results for focused task taken on the INEX 2009 Web site

References

1. Mitchell, P.C.: A note about the proximity operators in information retrieval. In: Proceedings of the 1973 meeting on Programming languages and information retrieval, ACM Press (1974) 177–180
2. Mitra, M., Buckley, C., Singhal, A., Cardie, C.: An analysis of statistical and syntactic phrases. In: Proceedings of RIAO-97, 5th International Conference “Recherche d’Information Assistee par Ordinateur”. (1997) 200–214
3. Rasolofo, Y., Savoy, J.: Term proximity scoring for keyword-based retrieval systems. In: 25th European Conference on IR Research, ECIR 2003. Number 2633 in LNCS, Springer (2003) 207–218
4. Büttcher, S., Clarke, C.L.A., Lushman, B.: Term proximity scoring for ad-hoc retrieval on very large text collections. In: ACM SIGIR ’06, New York, NY, USA, ACM (2006) 621–622
5. Song, R., Taylor, M.J., Wen, J.R., Hon, H.W., Yu, Y.: Viewing term proximity from a different perspective. In: ECIR 2008. Volume 4956 of Lecture Notes in Computer Science., Springer (2008) 346–357
6. Vechtomova, O., Karamuftuoglu, M.: Lexical cohesion and term proximity in document ranking. *Information Processing and Management* **44**(4) (2008) 1485–1502
7. Hearst, M.A.: Improving full-text precision on short queries using simple constraints. In: Proceedings of the 5th Annual Symposium on Document Analysis and Information Retrieval (SDAIR). (1996) 217–232
8. Clarke, C.L.A., Cormack, G.V., Burkowski, F.J.: Shortest substring ranking (multitext experiments for TREC-4). [13]
9. Hawking, D., Thistlewaite, P.: Proximity operators - so near and yet so far. [13]

10. Clarke, C.L.A., Cormack, G.V.: Shortest-substring retrieval and ranking. *ACM Transactions on Information Systems* **18**(1) (January 2000) 44–78
11. de Kretser, O., Moffat, A.: Effective document presentation with a locality-based similarity heuristic. In: *ACM SIGIR '99*, New York, NY, USA, ACM (8 1999) 113–120
12. Beigbeder, M., Mercier, A.: An information retrieval model using the fuzzy proximity degree of term occurrences. In Liebrock, L.M., ed.: *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, New York, NY, USA, ACM Press (2005)
13. Harman, D.K., ed.: *The Fourth Text REtrieval Conference (TREC-4)*. Number 500-236, Department of Commerce, National Institute of Standards and Technology (1995)

Use of Language Model, Phrases and Wikipedia Forward Links for INEX 2009

Philippe Mulhem¹ and Jean-Pierre Chevallet²

¹ LIG - CNRS, Grenoble, France
Philippe.Mulhem@imag.fr

² LIG - Université Pierre Mendès France, Grenoble, France
Jean-Pierre.Chevallet@imag.fr

Abstract. We present in this paper the work of the Information Retrieval Modeling Group (MRIM) of the Computer Science Laboratory of Grenoble (LIG) at the INEX 2009 Ad Hoc Track. Our aim this year was to twofold: first study the impact of extracted noun phrases taken in addition to words as terms, and second using forward links present in Wikipedia to expand queries. For the retrieval, we use a language model with Dirichlet smoothing on documents and/or doxels, and using an Fetch and Browse approach we select rank the results. Our best runs according to doxel evaluation get the first rank on the Thorough task, and according to the document evaluation we get the first rank for the Focused, Relevance in Context and Best in Context tasks.

1 Introduction

This paper describes the approach used by the MRIM/LIG research team for the Ad Hoc Track of the INEX 2009 competition. Our goal here is to experiment enrichment of documents and queries in two directions: first, using the annotation provided by Ralph Schenker and his colleagues [4] we expand the vocabulary to annotated noun phrases, and second, using forward pages extracted from wikipedia (dump processed in July 2009), we expand the user's queries. So, the vocabulary extension comes from internal Wikipedia data (categories for instance) and external data (Wordnet), and the forward information comes from only internal Wikipedia data.

Our work integrates structured documents during retrieval according to a Fetch and Browse framework [1], as retrieval is achieved in two steps: the first one focuses on whole articles, and the second one process integrates the non-article doxels in a way to provide *focused* retrieval according to the retrieved documents parts.

First of all, we define one term: a *doxel* is any part of an XML document between its opening and closing tag. We do not make any kind of difference between a doxel describing the logical structure of the document (like a title or a paragraph) or not, like anchors of links or words that are emphasized), a relation between doxels may come from the structural composition of the doxels, or from any other source. Assume that an xml document is “<A>This

is an `example` of `<C>XML</C>` document``". This document contains 3 doxels: the first is delimited by the tag A, the second is delimited by the tag B, and the third is delimited by the tag C. We also consider that a compositional link relates A to B, and A to C. We will also depict B and C as direct structural components of A.

The remaining of this paper is organized as follows: after commenting shortly in section 2 related works, we describe the vocabulary expansion based on noun phrases in part 3. Then the query expansion according to the forward links extracted from Wikipedia are presented in section 4. Section 6 introduces our *matching* process. Results of the INEX 2009 Ad Hoc track are presented in Section 7, where we present the twelve (three for each of the four tasks) officially submitted runs by the LIG this year. We conclude in part 8.

2 Related works

The language modeling approach to information retrieval exists from the end of the 90s [3]. In this framework, the relevance status value of a document for a given query is estimated by the probability of generating the query from the document. We used last year such model in INEX 2008 competition with some good results. Using noun phrases for information retrieval has also proven to be effective in some contexts like medical documents [2], that is why we chose to experiment extraction of such phrases for Inex 2009.

3 Extraction and use of noun phrases from YAWN

Here, we use the result YAWN [4] provided with the INEX 2009 collection. However, we do not use the tags that characterize the phrases, but the words themselves as additional terms of the vocabulary. For instance, in the document 12000.xml, the following text appears:

```
<music>
<composer wordnetid="109947232" confidence="0.9173553029164789">
<artist wordnetid="109812338" confidence="0.9508927676800064">
<link xlink:type="simple" xlink:href="../434/419434.xml">
Koichi Sugiyama</link></artist>
</composer>
</music>
```

The noun phrase surrounded by YAWN tags is the *Koichi Sugiyama*. In our case, the noun phrase *Koichi Sugiyama* is then used as a term that will be considered in the indexing vocabulary. In fact, the terms *Koichi* and *Sugiyama* are also indexing terms. We keep only the noun phrases that are described by YAWN because we consider that they are trustworthy to be good indexing terms. An inconsistency may appear, though: we noticed that all the noun phrases surrounded by YAWN tags were not always tagged (for instance the phrase “Koichi

Sugiyama” may occur in other documents without YAWN tags). That is why we ran one pass that generates the noun phrases terms for each occurrence of the YAWN tagged noun phrases. In the following, we refer to the initial vocabulary (keywords only) as *KBV*, the phrase-based only vocabulary as *Phrase*, and the union of both vocabulary as *KBV+Phrase*.

4 Extraction and use of Wikipedia forward links

The original wikipedia documents have also a very nice feature worth studying, which is related to forwarding links. To reuse a case taken from the query list of INEX 2009, if on Wikipedia we look for the page “VOIP”, the page is almost empty and redirects to the page titled “Voice over Internet Protocol”. In this case, we clearly see that some very strong semantic relationship exists between the initial term and the meaning of the acronym. We use this point in consideration by storing all these forward links, and by expanding the query expressions by the terms that occur in the redirected page (i.e. the title of the page pointed to). In this case, a query “VOIP” is then translated into “VOIP Voice over Internet Protocol”.

So, this expansion does not impact the IR model used, but only the query expression.

5 Indexing of Doxels

As it has been done in several approaches in the previous years, we choose to select doxels according to their type. In our case, we chose to consider only doxel that contain one title to be potentially relevant for queries. The list called L_d is then: article, ss, ss1, ss2, ss3, ss4.

We generated several indexing of the doxels using a language model that uses a dirichlet smoothing, using the Zettair system. One of the indexing is achieved according to *KBV*, and another one according to *KBV+Phrase*.

In our experiments, we worked on two sets of potential retrieved doxels: the article only doxels, and the doxels of type in L_d .

We integrated also the reference run provided by the INEX Ad-Hoc track organizers, by considering that the vocabulary is *KBV*, that the target doxels are articles only and that the model used is BM25.

6 Matching

We have defined a Fetch and Browse framework that can be processed in three steps:

- The first step generates a ranking according to a subset of doxel types, namely SS1,

- the second step generates a ranking according to another subset of doxel types SS2 (where no intersection exists between SS1 and SS2). Then, the doxels of SS2 are inserted withing the result list of step 1, according to the inclusion of the results of the processing of SS2 in SS1. This inclusion is called a Rerank.
- The third step, dedicated to the results expected for INEX 2009, is dedicated to remove the overlaps in the results. Here, we always give priority on the first results: we remove any of the doxels that overlaps an already seen doxel in the results list.

We note one Fetch and Browse configuration, based on the fact that the first step uses a IR model 1 to process a query on which we apply a preprocessing like a query expansion, applied on the set SS1 on the vocabulary one, and in the step 2 uses a IR model 2 to process a query on which we apply a preprocessing 1 like a query expansion, applied on the set SS1 on the vocabulary 2; then we fuse these results using a Rerank process, and finish by a removal of overlaps if needed as : `Remove_overlap(Rerank((query preprocessing 1, IR model 1, vocabulary 1,SS1),(query preprocessing 2, IR model 2, vocabulary 2,SS2)))`

For our runs we tested in fact the different vocabularies and the use of the forward links as query expansion.

7 Experiments and results

The INEX 2009 Adhoc track consists of four retrieval tasks: the Thourough task, Focused Task, Relevant In Context Task, and Best In Context Task. We submitted 3 runs for each of these tasks.

7.1 Thourough Task

For the Thourough task, we considered

- 1T: No fetch and browse here is processed, we just have a one step processing, (none, LM, KWV, L_d), which ranks all the potential doxels according to their rsv. This result can be considered as a baseline for our LM based approach ;
- 2T: In this run, we applied a Fetch and Browse approach: `No_remove_overlap(Rerankrsv((none, LM, KWV, {article}), (none, LM, KWV, $L_d \setminus \{article\}$)))`. Here the idea is to put priority on the whole article matching, and then to group all the documents doxels according to their rsv. Because we do not use any overlap we use the name *No_remove_overlap* for the removal of overlaps function.
- 3T: For our third Thourough run, the reference run is used as the Fetch step: `No_remove_overlap(Rerankrsv((none, BM25, KWV, {article}), (none, LM, KWV, $L_d \setminus \{article\}$)))`

From the table 1, we see that the language model and the BM25 with fetch and browse outperforms the other runs at recall 0.00. We notice also that the Fetch and Browse approach with language model underperforms the simple process of run 1T, which is not the case with the BM25 fetching.

Table 1. Thorough Task for LIG at INEX2008 Ad Hoc Track

<i>Run</i>	precision at 0.0 recall	precision at 0.01 recall	precision at 0.05 recall	precision at 0.10 recall	MAiP (rank / 30)	MAiP Doc. (rank / 30)
1T	0.575	0.570	0.544	0.496	0.281 (1)	0.344 (3)
2T	0.572	0.567	0.542	0.492	0.273 (4)	0.342 (4)
3T	0.582	0.569	0.535	0.493	0.281 (2)	0.333 (5)

7.2 Focused Task

The INEX 2009 Focused Task is dedicated to find the most focused results that satisfy an information need, without returning “overlapping” elements. In our focused task, we experiment with two different rankings.

We submitted three runs :

- 1F: This run is similar to the run 2T, but we apply the Remove_overlap operation, so the run is described by: `remove_overlap(Rerankrsv((none, LM, KWV, {article}),(none, LM, KWV, Ld\{article})))`;
- 2F: In this second Focused run, we study the impact of using the query expansion using the forward links of Wikipedia and the extended vocabulary in the first step of the matching: `remove_overlap(Rerankrsv((Forward_links_expansion, LM, KBV+Phrase, {article}), (none, LM, KWV, Ld\{article})))`;
- 3F: This run based on the reference run is achieved by removing the overlaps on the run 3T: `Remove_overlap(Rerankrsv((none, BM25, KWV, {article}), (none, LM, KWV, Ld\{article})))`.

Table 2. Focused Task for LIG at INEX2009 Ad Hoc Track

<i>Run</i>	ip[0.00]	ip[0.01] (rank / 57)	ip[0.05]	ip[0.10]	MAiP	MAiP Doc. (rank / 62)
1F	0.574	0.573 (22)	0.534	0.497	0.267	0.351 (1)
2F	0.551	0.532 (31)	0.490	0.453	0.236	0.300 (33)
3F	0.581	0.568 (24)	0.525	0.493	0.269	0.341 (3)

The results obtained by using *doxel* based evaluation show that the use of query expansion and noun phrases underperforms the other approaches, which seems to indicate that the matching of documents (for the fetching) is less accurate using these two expansions. The reason should come from the fact that the query expansion is quite noisy, but we will check this hypothesis in the future. Here again, using the language model outperforms the BM25 baseline provided, except for the ip[0.00] measure.

7.3 Relevant In Context Task

For the Relevant In Context task, we take “default” focused results and reordered the first 1500 doxels such that results from the same document are clustered together. It considers the article as the most natural unit and scores the article with the score of its doxel having the highest RSV. The runs submitted are similar to our Focused runs, but we apply a reranking of the document parts according to the reading order (the $\text{Rerank}_{reading}$ function), before taking care of removing overlapping parts. We submitted three runs :

- *1R*: This run is similar to the run 1F, but we apply the $\text{Rerank}_{reading}$ instead of the Rerank_{rsv} function. So the 1R run is described by: $\text{remove_overlap}(\text{Rerank}_{reading}((\text{none}, \text{LM}, \text{KWV}, \{\text{article}\}), (\text{none}, \text{LM}, \text{KWV}, L_d \setminus \{\text{article}\})))$;
- *2R*: Compared to the run 2F, we apply similar modifications than for the 1R run, leading to: $\text{remove_overlap}(\text{Rerank}_{reading}(\text{Forward_links_expansion}, \text{LM}, \text{KBV}+\text{Phrase}, \{\text{article}\}), (\text{none}, \text{LM}, \text{KWV}, L_d \setminus \{\text{article}\})))$;
- *3R*: Compared to the run 3F, we apply similar modifications than for the 1R run, leading to: $\text{Remove_overlap}(\text{Rerank}_{reading}((\text{none}, \text{BM25}, \text{KWV}, \{\text{article}\}), (\text{none}, \text{LM}, \text{KWV}, L_d \setminus \{\text{article}\})))$.

The results are presented in table 3, and the measures for the *doxel* based evaluations are using generalized precision where the results for the document based evaluation is the MAiP.

Table 3. Relevant In Context Task for INEX2009 Ad Hoc.

<i>Run</i>	gP[5]	gP[10]	gP[25]	gP[50]	MAgP (rank / 33)	MAiP Doc. (rank / 42)
<i>1R</i>	0.295	0.256	0.202	0.152	0.173 (12)	0.351 (1)
<i>2R</i>	0.189	0.171	0.135	0.105	0.091 (28)	0.168 (41)
<i>3R</i>	0.305	0.273	0.216	0.160	0.173 (13)	0.341 (9)

In the case of retrieval in context, even of the best MAgP value is for 1R, the BM25 based run performs better for all the gP values presented in table 3. Here the run 2R that uses query expansion and noun phrases obtains much lower results.

7.4 Best In Context Task

For this task, we take “default” focused results, and we return the best doxel in the document as best entry point by using the `keep_best` function.

We submitted three runs :

- *1B*: This run is similar to the run 1F, but we apply the `keep_best` instead of the `remove_overlap` function. So the 1B run is described by: $\text{keep_best}(\text{Rerank}_{rsv}((\text{none}, \text{LM}, \text{KWV}, \{\text{article}\}), (\text{none}, \text{LM}, \text{KWV}, L_d \setminus \{\text{article}\})))$;

- *2B*: Compared to the run 2F, we apply similar modifications than for the 1B run, leading to: `keep_best(Rerankreading((Forward_links_expansion, LM, KBV+Phrase, {article}),(none, LM, KWV, $L_d \setminus \{article\}$))))`;
- *3B*: Compared to the run 3F, we apply similar modifications than for the 1B run, leading to: `keep_best(Rerankreading((none, BM25, KWV, {article}),(none, LM, KWV, $L_d \setminus \{article\}$))))`.

The results are presented in table 4.

Table 4. Best In Context Task for INEX2009 Ad Hoc.

<i>Run</i>	gP[5]	gP[10]	gP[25]	gP[50]	MAgP (rank / 37)	MAiP Doc. (rank / 38)
<i>1B</i>	0.244	0.226	0.182	0.139	0.154 (17)	0.351 (1)
<i>2B</i>	0.169	0.159	0.128	0.101	0.087 (31)	0.168 (38)
<i>3B</i>	0.271	0.251	0.193	0.144	0.154 (16)	0.341 (7)

Here, for the first time with our runs, the BM25 run outperforms (very slightly: 0.1544 versus 0.1540) the language model one for the official evaluation measure on doxels. Which is not the case for th document based evaluation. Here also, our proposed expansions do not perform well.

7.5 Discussion

In this section, we concentrate on some runs with noun phrases and query expansion in a way to find out why our proposal did not work as planned, and if there is room for improvements.

First of all, on major problem that occur with our proposal is that the query expansion generates too many additional words to be really effective. For instance, the query 2009_005, “chemists physicists scientists alchemists periodic table elements”, is in fact translated into “chemists chemist periodic_table periodic table elements periodic table chemical elements periodic table elements periodic system periodic system elements periodic table mendeleev periodic table table elements periodic properties natural elements element symbol list groups periodic table elements representative element mendeleev periodic chart peroidic table elements mendeleev table periodic table elements periodicity elements organization periodic table periodic table chemical elements fourth period periodic table elements periodic patterns group 2a nuclear symbol”, which is a very long query. In this expanded query *periodic_table* is a noun phrase term. We see in this expansion that we find the word “mendeleev”, which is a very good terme for the query, but on the other side we see many words that are unrelated to the initial query, like “representative” “organization” “system”, “properties”, “fourth”, and so on. So the main problem that we face here is that the expanded queries are not narrowed enough to be really useful, leading to poor results according to the evaluation measure during the INEX 2009 evaluation campaign.

Things are not hopeless, though: consider the seven (i.e., 10% of the 69 INEX 2009 queries) best results obtained for our expansion based *Relevance in Context* run 2R according to MAgP, we get the results presented in table 5. There we see that potential improvement may be achieved through the use of our proposal. The query 091, “Himalaya trekking peak”, has been translated into “trekking_peak himalaya himalayas”: in this case the query result has been clearly improved by using the noun phrase “trekking_peak” instead of using the two words “trekking” and “peak”.

Table 5. Relevant In Context Task for INEX2009 Ad Hoc, query-based comparisons.

Query	gP[10]			AgP		
	R1	R2	R3	R1	R2	R3
2009_001	0.279	0.379	0.279	0.268	0.280	0.314
2009_015	0.686	0.687	0.686	0.395	0.295	0.421
2009_026	0.110	0.170	0.110	0.284	0.273	0.307
2009_029	0.736	0.926	0.636	0.369	0.439	0.287
2009_069	0.299	0.199	0.500	0.289	0.332	0.285
2009_088	0.427	0.391	0.462	0.429	0.450	0.486
2009_091	0.099	0.653	0.182	0.189	0.511	0.221
Average (relative difference)	0.377 (-22.6%)	0.487	0.408 (-16.2%)	0.318 (-13.8%)	0.369	0.332 (-10.0%)

8 Conclusion

In the INEX 2009 Ad Hoc track, we proposed several baseline runs limiting the results to specific types of doxels. From the official INEX 2009 measures we the MRIM/LIG reached the first place on 4 measures, mostly for the evaluations based on documents. The new directions studied that we proposed are based on vocabulary expansion using noun phrases and query expansions using forward links extracted from Wikipedia. The results obtained using these extensions underperformed the more crude approaches we proposed, but we shown here that there is great room for improvements using extracted data from Wikipedia.

References

1. Y. Chiamarella. Information retrieval and structured documents. In *Lecture Notes in Computer Science, Lectures on Information Retrieval, LNCS 1980/2001*, pages 286–309, 2007.
2. C. Lacoste, J.-P. Chevillet, J.-H. Lim, D. L. T. Hoang, X. Wei, D. Racoceanu, R. Teodorescu, and N. Vuillenemot. Inter-media concept-based medical image indexing and retrieval with umls at ipal. In *Lecture Notes in Computer Science, Evaluation of Multilingual and Multi-modal Information Retrieval*, pages 694–701, 2007.

3. J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Research and Development in Information Retrieval*, 1998.
4. R. Schenkel, F. M. Suchanek, and G. Kasneci. Yawn: A semantically annotated wikipedia xml corpus. In *2. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web (BTW 2007)*, pages 277–291, 2007.

Indian Statistical Institute at INEX 2009 Adhoc Focused Task

Sukomal Pal¹, Mandar Mitra¹, and Debasis Ganguly²

¹ Information Retrieval Lab, CVPR Unit
Indian Statistical Institute, Kolkata
India.

{sukomal_r, mandar}@isical.ac.in

² Synopsys,
Bangalore, India
debforit@gmail.com

Abstract. This paper describes the work that we did at Indian Statistical Institute towards XML retrieval for INEX 2009. Since there has been an abrupt quantum jump in INEX corpus size (from 4.6 GB with 659,388 articles to 50.7 GB with 2,666,190 articles), retrieval algorithms and systems are put to ‘stress test’ in INEX 2009 campaign. We tuned our text retrieval system (SMART) based on the Vector Space Model (VSM) that we have been using since INEX 2006. We managed to submit only two VSM-based document-level retrieval runs using blind feedback for the focused task: an initial run (*indsta_VSMpart*) over a small fraction of INEX 09 corpus, and another on full corpus (*indsta_VSMfb*). For both the runs we considered *Content-Only*(CO) retrieval, more specifically, Title and Description fields of the INEX 09 adhoc queries (2009001-2009115). The performance of our document-level runs is not upto the mark. However it clearly shows us the roadmap as to what needs to be done. How to enable our system to handle with this large corpus efficiently is our first priority. We also need to make sure that Language Modelling (LM)-based implementation works with INEX 09 collection, which yielded reasonably good performance at last INEX. Also, with both VSM and LM, we need to tune the system for effective element-level retrieval.

1 Introduction

Traditional Information Retrieval systems return whole documents in response to queries, but the challenge in XML retrieval is to return the most relevant parts of XML documents which meet the given information need. Since INEX 2007 [1], arbitrary passages are permitted as retrievable units, besides the usual XML elements. A retrieved passage consists of textual content either from within an element or spanning a range of elements. Since INEX 2007, the adhoc retrieval task has also been classified into three sub-tasks: a) the FOCUSED task which asks systems to return a ranked list of elements or passages to the user; b) the RELEVANT in CONTEXT task which asks systems to return relevant elements

or passages grouped by article; and c) the BEST in CONTEXT task which expects systems to return articles along with one best entry point to the user.

Along with these, this year INEX sees return of the d) THOROUGH task, where all the relevant items (either passages or elements) from a document are retrieved. Here, overlap among the elements are permitted but they are ranked according to relevance order.

Each of the four subtasks can be again sub-classified based on different retrieval approaches:

- element retrieval versus passage retrieval
- Standard keyword query or Content-Only(CO) retrieval versus structured query or Content-And-Structure (CAS) retrieval
- Standard keyword query (CO) retrieval versus phrase query retrieval.

In the CO task, a user poses a query in free text and the retrieval system is supposed to return the most relevant elements/passages. A CAS query can provide explicit or implicit indications about what kind of element the user requires along with a textual query. Thus, a CAS query contains structural hints expressed in XPath [2] along with an *about()* predicate. Phrase query is a new entity introduced in the INEX 2009 queries using explicit notation of multi-word phrases to make the query more verbose and to see how verbose queries impact on retrieval effectiveness.

This year we submitted two adhoc focused runs, both using a Vector Space Model (VSM) based approach with blind feedback. VSM sees both the document and the query as bags of words, and uses their *tf-idf* based weight-vectors to measure the inner product *similarity* as a measure of closeness between the document and the query. The documents are retrieved and ranked in decreasing order of the similarity-value.

We used a modified version of the SMART system for the experiments at INEX 2009. Since the corpus used for the adhoc track this year is huge compared to the earlier INEXes ([3], [4]), the system was really put to a ‘stress test’ both in terms of robustness and time-efficiency. To make sure that at least one retrieval run was completed within the stipulated time, one of our submissions (*indsta_VSMpart*) was run on a partial data (10060 documents from INEX 09 corpus). The other run (*indsta_VSMfb*) was however on the complete corpus. For both the runs, retrieval was at the whole-document level, using query expansion based on blind feedback after the initial document retrieval. We considered CO queries only using *title* and *description* fields. In the following section, we describe our general approach for the runs, and discuss results and further work in Section 3.

2 Approach

2.1 Indexing

We first shortlisted 74 tags from previous INEX Wikipedia corpus [3] which were reasonably frequently occurring within the corpus (least frequency 2), contain

some useful text (at least 10 characters) and featured in the INEX 07 qrels like: $\langle article \rangle$, $\langle body \rangle$, $\langle caption \rangle$, $\langle center \rangle$, $\langle collectionlink \rangle$, $\langle definitionitem \rangle$, $\langle defintionlist \rangle$, $\langle div \rangle$, $\langle em \rangle$, $\langle figure \rangle$, $\langle gallery \rangle$, $\langle item \rangle$, $\langle outsidelink \rangle$, $\langle p \rangle$, $\langle section \rangle$, $\langle wikipedialink \rangle$, etc. Documents were parsed using the libxml2 parser, and only the textual portions included within the selected tags were used for indexing. Similarly, for the topics, we considered only the *title* and *description* fields for indexing, and discarded the *inex-topic*, *castitle* and *narrative* tags. No structural information from either the queries or the documents was used.

The extracted portions of the documents and queries were indexed using single terms and a controlled vocabulary (or pre-defined set) of statistical phrases following Salton’s blueprint for automatic indexing [5].

Stopwords listed in the standard stop-word list included within SMART were removed from both documents and queries. Words were stemmed using a variation of the Lovins’ stemmer implemented within SMART. Frequently occurring word bi-grams (loosely referred to as phrases) were also used as indexing units. We used the N-gram Statistics Package (NSP)³ on the English Wikipedia text corpus from INEX 2006 and selected the 100,000 most frequent word bi-grams as the list of candidate phrases. Documents and queries were weighted using the *Lnu.ltn* [6] term-weighting formula. For the initial run, we used *slope* = 0.2 and *pivot* = 120 and retrieved 1500 top-ranked XML documents for each of 115 adhoc queries (2009001 - 2009115).

Next we used blind feedback to retrieve whole documents. We applied automatic query expansion following the steps given below for each query (for more details, please see [7]).

1. For each query, collect statistics about the co-occurrence of query terms within the set \mathcal{S} of 1500 documents retrieved for the query by the baseline run. Let $df_{\mathcal{S}}(t)$ be the number of documents in \mathcal{S} that contain term t .
2. Consider the 50 top-ranked documents retrieved by the baseline run. Break each document into overlapping 100-word windows.
3. Let $\{t_1, \dots, t_m\}$ be the set of query terms (ordered by increasing $df_{\mathcal{S}}(t_i)$) present in a particular window. Calculate a similarity score Sim for the window using the following formula:

$$Sim = idf(t_1) + \sum_{i=2}^m idf(t_i) \times \min_{j=1}^{i-1} (1 - P(t_i|t_j))$$

where $P(t_i|t_j)$ is estimated based on the statistics collected in Step 1 and is given by

$$\frac{\# \text{ documents in } \mathcal{S} \text{ containing words } t_i \text{ and } t_j}{\# \text{ documents in } \mathcal{S} \text{ containing word } t_j}$$

This formula is intended to reward windows that contain multiple matching query words. Also, while the first or “most rare” matching term contributes its full idf (inverse document frequency) to Sim , the contribution of any

³ <http://www.d.umn.edu/~tperderse/nsp.html>

subsequent match is deprecated depending on how strongly this match was predicted by a previous match — if a matching term is highly correlated to a previous match, then the contribution of the new match is correspondingly down-weighted.

4. Calculate the maximum *Sim* value over all windows generated from a document. Assign to the document a new similarity equal to this maximum.
5. Rerank the top 50 documents based on the new similarity values.
6. Assuming the new set of top 20 documents to be relevant and all other documents to be non-relevant, use Rocchio relevance feedback to expand the query. The expansion parameters are given below:

number of words = 20
 number of phrases = 5
 Rocchio $\alpha = 4$
 Rocchio $\beta = 4$
 Rocchio $\gamma = 2$.

For each topic, 1500 documents were retrieved using the expanded query.

3 Results

Our performance as reported in the INEX09 website using relevance judgements for 68 topics are shown in Table 1 and Table 2.

Table 1. Subdocument-level (element/passage) evaluation for the FOCUSED, CO task

Run Id	iP@0.01
indsta_VSMfb	0.0078
indsta_VSMpart	0.0003

Table 2. Document-level evaluation for the FOCUSED, CO task

Run Id	MAP
indsta_VSMfb	0.0055
indsta_VSMpart	0.0000

The performance of our system is really dismal. We are working on reproducing the results reported, as the data and programs used for evaluation are really resource-hungry. Once we get the results at different recall level per query level, we will be able to analyse the results and thus improve our performance.

4 Conclusion

The test collection used in this INEX is really resource-hungry and puts the models and their implementations into stress test. We approached the job very late this year and found ourselves inadequately prepared. We therefore submitted two simple runs: one using a minimal portion of corpus and another using the whole corpus. Both the runs were at the document-level retrieval using VSM-based approach. We could not try other approaches in terms of the retrieval model (e.g. LM) or granularity (element/passage level) before the official submission. The results were dismal, but not unexpected. Our immediate task is to make the evaluation program run on our system, reproduce the results reported, analyse the score at per query level and try to better our performance with proper parameter-tuning for the VSM approach, using other approaches like LM and element-level retrieval. There is plenty of work to do, some of which will definitely be attempted and addressed in the coming days.

References

1. INEX: Initiative for the Evaluation of XML Retrieval (2009) <http://www.inex.otago.ac.nz>.
2. W3C: XPath-XML Path Language(XPath) Version 1.0 <http://www.w3.org/TR/xpath>.
3. Denoyer, L., Gallinari, P.: The Wikipedia XML corpus. In Fuhr, N., Lalmas, M., Trotman, A., eds.: Comparative Evaluation of XML Information Retrieval Systems. (2006) 12–19
4. Schenkel, R., Suchanek, F.M., Kasneci, G.: Yawn: A semantically annotated wikipedia xml corpus. In: BTW. (2007) 277–291
5. Salton, G.: A Blueprint for Automatic Indexing. ACM SIGIR Forum **16**(2) (Fall 1981) 22–38
6. Buckley, C., Singhal, A., Mitra, M.: Using Query Zoning and Correlation within SMART: TREC5. In Voorhees, E., Harman, D., eds.: Proc. Fifth Text Retrieval Conference (TREC-5), NIST Special Publication 500-238 (1997)
7. Mitra, M., Singhal, A., Buckley, C.: Improving automatic query expansion. In: SIGIR 98, Melbourne, Australia, ACM (1998) 206–214

Universities of Avignon and Lyon 3 at INEX 2009

Eric SanJuan¹ and Fidelia Ibekwe-SanJuan²

¹ LIA & IUT STID, Université d'Avignon
339, chemin des Meinajaries, Agroparc BP 1228,
84911 Avignon Cedex 9, France.

`eric.sanjuan@univ-avignon.fr`

² ELICO, Université de Lyon 3
4, Cours Albert Thomas, 69008 Lyon, France.
`ibekwe@univ-lyon3.fr`

Abstract. Following our previous participation at INEX 2008 Ad-hoc track, we continue to address both standard and focused retrieval tasks based on comprehensible language models and interactive query expansion (IQE). Query topics are expanded using an initial set of Multi Word Terms (MWTs) selected from top n ranked documents. MWTs are special text units that represent domain concepts and objects. As such, they can better represent query topics than ordinary phrases or n -grams. In this experiment we extract terms from article titles, narrative query field and automatically generated summaries. We also combined the initial set of MWTs obtained in an IQE process as well as with automatic query expansion (AQE) using language models and smoothing mechanism. We chose as baseline the Indri IR engine based on the language model using Dirichlet smoothing. The experiment is carried out on all INEX 2009 Ad-hoc tasks.

1 Introduction

Previous experiments carried out within the framework of TREC [1] tended to conclude that retrieval performance has not been enhanced by adding NLP, especially syntactic level of processing. The problem lies in determining the level of NLP needed, on which text units to implement it, whether to implement NLP on both queries and documents and at what stage (whole collection or only on an initial set of returned documents). Previous research also concluded that a deep syntactic representation of queries and documents is not useful to achieve a state-of-the-art performance in IR [2]. It may on the contrary degrade results. On the other hand, performance can be boosted by better representing queries and documents with longer phrases using shallow NLP. In some cases, even a well-tuned n -gram approach can approximate the extraction of phrases and may suffice to boost retrieval performance.

Up until 2004, the dominant model in IR remained the bag-of-words representation of documents which continued to show superior performances in IR.

However, a series of experiments carried out on several document collections over the past years are beginning to show a different picture. Notwithstanding the apparent success of the bag-of-words representation in some IR tasks, it is becoming clear that certain factors related mostly to query length and document genre (general vs technical) influence the performance of IR systems. For instance, [1, 3] showed that representing queries and document by longer phrases can improve systems' performances since these text units are inherently more precise and will better disambiguate the information need expressed in the queries than lone words.

Furthermore, [1] concluded that the issue of whether or not to use NLP and longer phrases would yield better results if focused on query representation rather than on the documents themselves because no matter how rich and elaborate the document representation, a poor representation of the information need (short queries of 1-2 words) will ultimately lead to poor retrieval performance.

Based on these earlier findings, we investigate the issue of representing queries with a particular type of phrase which are Multiword Terms (MWTs). MWTs is understood here in the sense defined in computational terminology [4] as textual denominations of concepts and objects in a specialized field. Terms are linguistic units (words or phrases) which taken out of context, refer to existing concepts or objects of a given field. As such, they come from a specialized terminology or vocabulary [5]. MWTs are thus terms of length >1 . MWTs, alongside noun phrases, have the potential of disambiguating the meaning of the query terms out of context better than single word terms or statistically-derived n -grams and text spans. In this sense, MWTs cannot be reduced to words or word sequences that are not linguistically and terminologically grounded. A novelty in INEX 2009 is that a new field has been added to queries with samples of relevant MWTs.

An initial selection of MWTs from queries is used in an Interactive Query Expansion (IQE) process to acquire more MWTs from top n -ranked documents. The expanded set is submitted to standard IR Language Models for document ranking. We also test expanding the query automatically (AQE) using the query expansion mechanism in Indri. Our approach was successfully tested on two corpora: the TREC Enterprise track 2007 and 2008 collections, and INEX 2008 Ad-hoc track [6] but only at the document level. This year at Inex 2009 we consider more advanced NLP approaches including automatic multi-document summarizing. We also consider XML element retrieval.

The rest of the paper is structured as follows. Section §2 presents our language model and its application to the IR tasks. Section §3 presents our preliminary results on the Wikipedia collection in the INEX 2009 Ad-hoc track. Finally, section §4 discusses the preliminary results from these experiments.

2 Probabilistic IR Model

2.1 Language Model

Language models are widely used in NLP and IR applications [7, 8]. In the case of IR, smoothing methods play a fundamental role [9]. We shall first describe the probability model that we use.

Document Representation: probabilistic space and smoothing Let us consider a finite collection \mathcal{D} of documents, each document D being considered as a sequence $(D_1, \dots, D_{|D|})$ of $|D|$ terms D_i from a language \mathcal{L} , i.e. \mathcal{D} is an element of \mathcal{L}^* , the set of all finite sequences of elements in \mathcal{L} . Our formal framework is the following probabilistic space $(\Omega, \wp(\Omega), P)$ where Ω is the set of all occurrences of terms from \mathcal{L} in some document $D \in \mathcal{D}$ and P is the uniform distribution over Ω . LMs for IR rely on the estimation of the a priori probability $P_D(q)$ of finding a term $q \in \mathcal{L}$ in a document $D \in \mathcal{D}$. We chose the Dirichlet smoothing method because it can be viewed as a maximum *a priori* (MAP) document probability distribution. Given an integer μ , it is defined as:

$$P_D(q) = \frac{f_{q,D} + \mu \times P(q)}{|D| + \mu} \quad (1)$$

In the present experiment, documents can be full wikipedia articles, sections or paragraphs. Each of them defining a different probabilistic space that we shall combine in our runs.

Query Representation and ranking functions Like in INEX 2008, our purpose is to test the efficiency of MWTs in standard and focused retrieval compared to a bag-of-words model and statistically-derived phrases. For that, we shall consider phrases (instead of single terms) and a simple way of combining them. Given a phrase $s = (s_0, \dots, s_n)$ and an integer k , we formally define the probability of finding the sequence s in the corpus with at most k insertions of terms in the following way. For any document D and integer k , we denote by $[s]_{D,k}$ the subset of $D_i \in D$ such that: $D_i = s_1$ and there exists n integers $i < x_1, \dots, x_n \leq i + n + k$ such that for each $1 \leq j \leq n$ we have $s_j = D_{x_j}$.

We can now easily extend the definition of probabilities P and P_D to phrases s by setting $P(s) = P([s]_{.,k})$ and $P_D(s) = P_D([s]_{D,k})$. Now, to consider queries that are set of phrases, we simply combine them using a weighted geometric mean as in [10] for some sequence $w = (w_1, \dots, w_n)$ of positive reals. Unless stated otherwise, we shall suppose that $w = (1, \dots, 1)$, i.e. the normal geometric mean. Therefore, given a sequence of weighted phrases $Q = \{(s_1, w_1), \dots, (s_n, w_n)\}$ as query, we shall rank documents according to the following scoring function $\Delta_Q(D)$ defined by:

$$\Delta_Q(D) = \prod_{i=1}^n (P_D(s_i))^{\frac{w_i}{\sum_{j=1}^n w_j}} \quad (2)$$

$$\stackrel{\text{rank}}{=} \sum_{i=1}^n \left(\frac{w_i}{\sum_{j=1}^n w_j} \times \log(P_D(s_i)) \right) \quad (3)$$

This plain document ranking can easily be computed using any passage information retrieval engine. We chose for this purpose the Indri engine [11] since it combines a language model (LM) [7] with a bayesian network approach which can handle complex queries [10]. However, in our experiments, we use only a very small subset of the weighting and ranking functionalities available in Indri.

2.2 Query Expansion

We propose a simple QE process starting with an approximate short query $Q_{T,S}$ of the form (T, \mathcal{S}) where $T = (t_1, \dots, t_k)$ is an approximate document title consisting of a sequence of k words, followed by a possibly empty set of phrases: $\mathcal{S} = \{S_1, \dots, S_i\}$ for some $i \geq 0$. In our case, each S_i will be a MWT.

Baseline document ranking function By default, we shall rank documents according to

$$\Delta_{T,S} = \Delta_T \times \Delta_{i=1}^{|\mathcal{S}|} S_i \quad (4)$$

Therefore, the larger \mathcal{S} is, the less the title part T is taken into account. Indeed, \mathcal{S} consists of coherent set of MWTs found in a phrase query field or chosen by the user. If the query can be expanded by coherent clusters of terms, then we are no more in the situation of a vague information need and documents should be ranked according to precise MWTs. For our baseline, we shall generally consider \mathcal{S} to be made of the phrases given in the query.

Interactive Multiword Term Selection The IQE process works in the following manner. We consider the top twenty ranked documents of Δ_Q ranking. These documents are split into sentences and a multi-document summary is produced. MWTs are then extracted from this summary based on shallow parsing and proposed as possible query expansions. The user selects all or a subset \mathcal{S}' of them. This leads to acquiring sets of synonyms, abbreviations, hypernyms, hyponyms and associated terms with which to expand the original query terms. The selected multiword terms S'_i are added to the initial set \mathcal{S} to form a new query $Q' = Q_{T,S \cup \mathcal{S}'}$ leading to a new ranking $\Delta_{Q'}$ computed as previously in §2.2.

We also extract MWTs from the narrative and title fields of articles.

Automatic Query expansion To compare the interactive QE (IQE) process with an automatic one, we also experimented with the automatic query expansion (AQE). In our model, it consists in the following. Let D_1, \dots, D_K be the top ranked documents by the initial query Q . Let $C = \cup_{i=1}^K D_i$ be the concatenation of these K top ranked documents. Terms c occurring in D can be ranked

according to $P_C(c)$ as defined by equation (1). We consider the set E of the N terms $\{c_1, \dots, c_N\}$ having the highest probability $P_C(c_i)$. We then consider the new ranking function Δ'_Q defined by $\Delta'_Q = \Delta_Q^\lambda \times \Delta_E^{1-\lambda}$ where $\lambda \in [0, 1]$.

Unless stated otherwise we shall take $K = 4$, $N = 50$ and $\lambda = 0.1$. We now explore in which context IQE based on MWTs is efficient. Our baseline is an automatic document retrieval based on equation 2 in §2.1.

3 Results

We submitted four runs for the official INEX evaluation: two to the focused task and two to the thorough task. We compare these runs to two additional ones that were not submitted but were generated after the official results based on the preliminary qrels released by the organizers.

Each of these runs is a combination of the following strategies:

Xml XML elements can be retrieved and not only complete documents. Each element is evaluated in the probabilistic space of all elements sharing the same tag. Elements are then ranked by decreasing probability. The following elements were considered: b, bdy, category, causal_agent, country, entry, group, image, it, list, location, p, person, physical_entity, sec, software, table, title.

Doc Only full articles are retrieved.

QE Automatic Query Expansion (AQE) is used.

mwt Interactive Query Expansion (IQE) is used based on a selection of Multi Word Terms suggested to the user by the system.

Ti Elements in documents whose title overlaps the initial query or its expansion are favoured.

We consider the following runs:

Xml+Ti+QE (Lyon3LIAautolmnt) submitted to the thorough task.

Xml+Ti+QE+mwt (Lyon3LIAMANlmnt) submitted to the thorough task.

Doc+Ti+QE (Lyon3LIAautoQE) submitted to the focused task.

Doc+Ti+QE+mwt (Lyon3LIAMANQE) submitted to the focused task.

Doc baseline run not submitted.

Doc+mwt baseline run with IQE based on MWTs, not submitted.

Figure 1 shows the Interpolated generalized precision curves based on thorough (or focus) evaluation measures.

As in the INEX 2008 corpus, it appears that QE based on MWTs can improve document or XML element retrieval. However, it appears that AQE does not, and this is a surprising difference with last year's results. Contrary to our last year's runs, this year we used AQE in all our submitted runs because it previously gave better results. It was therefore surprising to observe that our non submitted baseline runs not implementing any query expansion appear to perform better than our submitted runs implementing one form of query expansion (AQE or IQE).

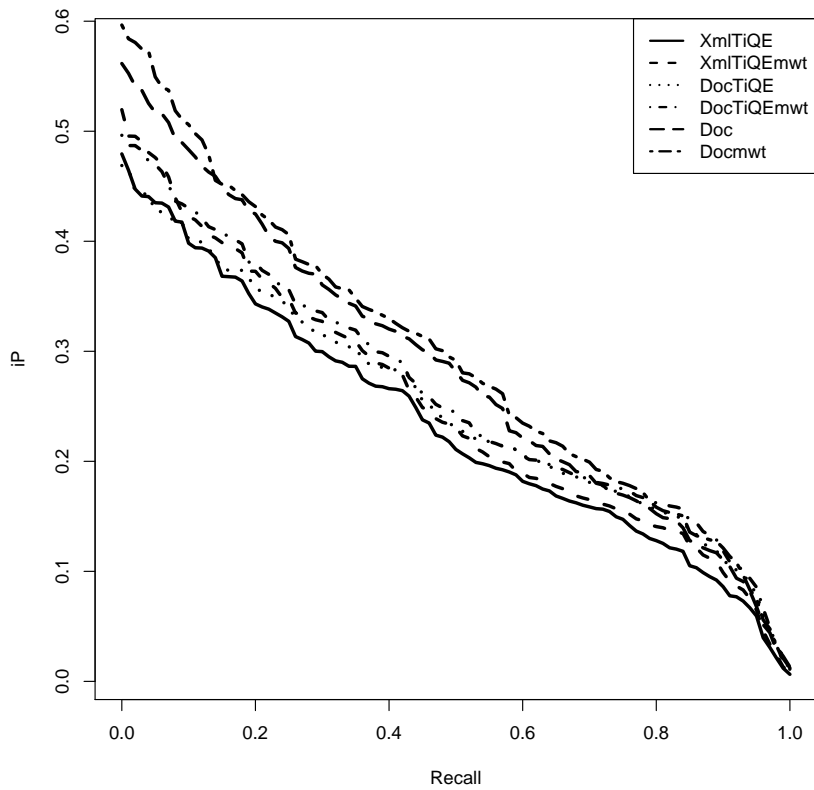


Fig. 1. Interpolated generalized precision curves at INEX 2009.

Moreover, it appears on these baseline runs that the difference between our automatic baseline run and the one with IQE is not statistically significant. In fact with an $I_p[0.01]$ of 0.56 and a MAiP of 0.28 our baseline run performs much better than last year meanwhile the score of our MWT runs is unchanged. This requires further experiments. The reason could be the availability this year in the topics of a new query field with phrases that is used by all of our runs, including the baseline. Thus making the input to the baseline runs somewhat similar to the runs using MWTs.

4 Discussion

We used Indri with Dirichlet smoothing and we combined two language models, one on the documents and one on elements.

The results from both models are then merged together and ranked by decreasing probability.

For queries we used NLP tools (summarizer and terminology extraction). We started from the topic phrase and title, then we added related Multi Word Terms (MWT) extracted from the other topic fields and from an automatic summary of the top ranked documents by this initial query. We also used standard Query Expansion when applied to the document model.

Other features are the indri operators to allow insertions in the MWTs and favoring documents in which the MWTs appear in the title.

As observed last year on the previous INEX coprus, IQE based on MWTs still improves retrieval effectiveness. On the contrary, automatic query expansion (AQE) using the same parameters has the reverse effect. This needs further investigation and tests with different parameters.

References

1. Perez-Carballo, J., Strzalkowski, T.: Natural language information retrieval: progress report. *Information Processing and Management* **36**(1) (2000) 155 – 178
2. Smeaton, A.F.: Using nlp and nlp resources for information retrieval tasks. Kluwer Academic Publishers (1999) 99–109
3. Mishne, G., de Rijke, M.: Boosting web retrieval through query operations. *Lecture Notes in Computer Sciences* **3408** (2006) 502 – 516
4. Kageura, K.: *The dynamics of Terminology: A descriptive theory of term formation and terminological growth*. John Benjamins, Amsterdam (2002)
5. Ibekwe-SanJuan, F.: Constructing and maintaining knowledge organization tools: a symbolic approach. *Journal of Documentation* **62** (2006) 229–250
6. Ibekwe, F., SanJuan, E.: ”use of multiword terms and query expansion for interactive information retrieval”. In Geva, S., Kamps, J., Trotman, A., eds.: *INEX 2008 (selected papers)*, LNCS 5631, Berlin Heidelberg, Springer-Verlag (2008) 54 – 64
7. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: *SIGIR ’98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, ACM (1998) 275–281
8. Jones, K.S., Walker, S., Robertson, S.E.: A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.* **36**(6) (2000) 779–840
9. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.* **22**(2) (2004) 179–214
10. Metzler, D., Croft, W.B.: Combining the language model and inference network approaches to retrieval. *Information Processing and Management* **40**(5) (2003) 735–750
11. Strohman, T., Metzler, D., Turtle, H., Croft, W.B.: *Indri: A language-model based search engine for complex queries (extended version)*. IR 407, University of Massachusetts (2005)

How well does Best in Context reflect ad hoc XML retrieval? Reprise

James A. Thom

RMIT University, Melbourne, Australia
james.thom@rmit.edu.au

Abstract. This paper describes the participation of RMIT in the 2009 ad hoc track for INEX. We repeated experiments done in 2007 that showed article level retrieval does surprisingly well on the Best in Context task.

Introduction

In 2007 we questioned how useful the best in context task reflected ad hoc retrieval [1]. Our experiments in 2007 used the Zettair¹ search engine to do article retrieval for the Focused, Relevant in Context, and Best in Context tasks; for the Best in Context task, article level retrieval did surprisingly well. In 2009 we report similar results for the Best in Context task on the new INEX collection with the 2009 ad hoc topics.

Approach and Results

Zettair is an open source search engine developed at RMIT. We used a stable released version of Zettair (version 0.9.3) which we ran on a MacBook computer with 2.16 GHz Intel Core 2 Duo running Mac OS X version 10.5.8 with 4 MB of L2 Cache and 2 GB memory. In the queries we used the CO titles of the topics and two similarity measures: `RMIT09title` used the default similarity measure in Zettair (a language model with Dirichlet-smoothing), and `RMIT09title0` used Okapi BM25. The best-entry-point was set at the start of the article.

As shown in the table below when evaluated with the MAgP metric the run `RMIT09title` performed better than the run `RMIT09title0`, whereas when evaluated as document level runs using MAP the order of the runs was reversed.

Discussion

Once again it appears that the Best in Context task is a challenging task, as in the vast majority of cases the best entry point for an INEX answer is very close to the start of the article. Thus, once again, it is very difficult for systems to identify when the best entry should be set anywhere else apart from the start of the document.

¹ <http://www.seg.rmit.edu.au/zettair/>

Table 1. Results

run	MAgP score	MAgP rank	MAP score	MAP rank
RMIT09title	0.1608	12	0.3540	2
RMIT09title0	01710	2	0.3181	22

References

1. J. A. Thom and J. Pehcevski. How well does best in context reflect ad hoc XML retrieval? In *Preproceedings INEX 2007*, 2007.

Exploiting Semantic Tags in XML Retrieval

Qiuyue Wang^{1,2}, Qiushi Li^{1,2}, Shan Wang^{1,2}, Xiaoyong Du^{1,2},
Yina Geng^{1,2}, Zuoyan Qin^{1,2}

¹ School of Information, Renmin University of China,

² Key Laboratory of Data Engineering and Knowledge Engineering, MOE,
Beijing 100872, P. R. China

{qiuyuew, liqiushi, swang, duyong, ygeng, qinzuoyan}@ruc.edu.cn

Abstract. With the new semantically annotated Wikipedia XML corpus, we attempt to investigate the following two research questions. Do the structural constraints in CAS queries help in retrieving an XML document collection containing semantically rich tags? How to exploit the semantic tag information to improve the CO queries as most users prefer to express the simplest forms of queries? In this paper, we describe and analyze the work done on comparing CO and CAS queries over the document collection at INEX 2009 ad hoc track, and we propose a method to improve the effectiveness of CO queries by enriching the element content representations with semantic tags. Our results show that the approaches of enriching XML element representations with semantic tags are effective in improving the early precision, while on average precisions, strict interpretation of CAS queries are generally superior.

1 Introduction

With the growth of XML, there has been increasing interest in studying structured document retrieval. A key characteristic that distinguishes the XML retrieval task from a traditional retrieval task is the existence of structural information in the former one. The structural information not only enables the system to retrieve the document fragments rather than the whole documents relevant to users' queries, but also provides new dimensions to be exploited to improve the retrieval performance.

For example, a common approach to exploit the hierarchical structure in XML documents is to score the leaf elements that directly contain terms and propagate the scores up to their ancestors. Thus the scores of elements up in the tree are calculated as weighted combinations of their descendants' scores. Such a score propagation strategy can reflect the hierarchical level of the elements (the lower elements are considered as more specific than the upper elements), and also the weights can be set to reflect the importance of different element types. Another well-accepted idea of utilizing the structural information to improve the retrieval performance is to formulate more precise queries by specifying structural conditions in the query. For example, by specifying that the return element type should be movie, or John should be a director, the retrieval precision can be greatly improved. Structural conditions add more semantics into the query as discussed in [1] by *specifying target information*

type, disambiguating keywords, specifying search term context, and/or relating search query terms. They can be explicitly specified by the user or automatically inferred by the system.

There are various structured query languages, e.g. XML fragments [2], NEXI [3], field-based structured query languages as that used in Lemur/Indri [4], XQuery Full-Text [5] and etc. They differ in their expressive power on specifying structural constraints, from the simplest one (with only the return element type) to the most complex one (with the support of full-fledged XQuery). For most users, however, complex structured queries are hard to construct. Moreover, once such queries were incorrectly formulated or inferred, possibly due to the imprecise knowledge about the document structures or the semantic gap between the query and data, strict matching of these queries would greatly harm the retrieval precision instead of improving it. To overcome this problem, we can make the hard constraints soft by treating the structural conditions as “hints” rather than “constraints”. The structural hints however won’t help much in XML retrieval as analyzed from the previous INEX Ad hoc tracks [6]. We think the reasons partially lie in the document collections used in previous INEX tracks. Both the collection of IEEE journal articles used from INEX 2002 to 2005 and the Wikipedia document collection used from INEX 2006 to 2008 contain very few semantic tags, such as movie, director, but mostly structural tags, like article, sec, p and etc. When expressing queries with only structural tags, the user intends to constrain the size of results rather than make the query semantically clearer. For example, when the user specifies the return element type to be a section or a paragraph, it has nothing to do with the topic relevance of the query. Users are in general bad at giving such structural hints [6]. Thus the performance improvement is not significant.

In INEX 2009, document collection is changed to a semantically annotated Wikipedia XML corpus [7]. In this corpus, the Wikipedia pages, as well as the links and templates in the pages, are annotated with semantically rich tags using the concepts from WordNet and etc. For Example, Fig. 1 shows an excerpt of an example XML document (4966980.xml) in the collection. With this new semantics-annotated document collection, we attempt to investigate the following research questions:

1. Do the structural constraints in CAS queries help in retrieving such an XML document collection containing semantically rich tags?
2. How to exploit the semantic tag information to improve the CO queries as most users prefer to express the simplest forms of queries?

In this paper, we describe and analyze the work done on comparing CO and CAS queries on such a semantically annotated XML corpus, and propose to improve the effectiveness of CO queries by enriching element content representations with semantic tags. Our experimental results show that the approaches of enriching XML element representations with semantic tags are effective in improving the early precision, while on average precisions, strict interpretation of CAS queries are generally superior.

The paper is organized as follows. Section 2 describes the baseline retrieval models for CO and CAS queries in XML retrieval used in our comparisons. In Section 3, we discuss the methodology for exploiting semantic tags in evaluating CO queries for XML retrieval. The experiment results are presented in Section 4. Finally, Section 5 concludes the paper and describes future work.

```

<article xmlns:xlink="http://www.w3.org/1999/xlink">
  <physical_entity confidence="0.8" wordnetid="100001930">
    <communicator confidence="0.8" wordnetid="109610660">
      <person confidence="0.8" wordnetid="100007846">
        <causal_agent confidence="0.8" wordnetid="100007347">
          <writer confidence="0.8" wordnetid="110794014">
            <dramatist confidence="0.8" wordnetid="110030277">
              <header>
                <title>Sam Ukala</title>
                <id>4966980</id>
                <revision>
                  <timestamp>2007-08-12T14:57:18Z</timestamp>
                  <contributor><username>Cydebot</username></contributor>
                </revision>
                <categories>
                  <category>Nigerian dramatists and playwrights</category>
                </categories>
              </header>
              <bdy>
                <b>Sam Ukala</b> is a
                <link xlink:type="simple" xlink:href="../383/21383.xml">Nigerian</link>
                playwright, poet, short story writer, actor, theatre director and academic. He has been Professor of Drama and Theatre Arts at a number of Nigerian universities, including
                <region wordnetid="108630985" confidence="0.8">
                  <administrative_district wordnetid="108491826" confidence="0.8">
                    <location wordnetid="100027167" confidence="0.8">
                      <district wordnetid="108552138" confidence="0.8">
                        <country wordnetid="108544813" confidence="0.8">
                          <link xlink:type="simple" xlink:href="../627/2227627.xml">Edo State</link>
                        </country>
                      </district>
                    </location>
                  </administrative_district>
                </region> University and .....
              </bdy>
            </dramatist>
          </writer>
        </causal_agent>
      </person>
    </communicator>
  </physical_entity>
</article>

```

Fig. 1. An excerpt of document 4966980.xml in the semantically annotated Wikipedia collection.

2 Baseline Approaches

Language modeling is a newly developed and promising approach to information retrieval. It has a solid statistical foundation, and can be easily adapted to model various kinds of complex and special retrieval problems, such as structured document retrieval. In particular, mixture models [8] and hierarchical language models [9][10][11] are proposed to be applied in XML retrieval. We base our work on language modeling approaches for XML retrieval. In this section, we describe the baseline XML retrieval models for both CO and CAS queries compared in our experiments.

We model an XML document as a node-labeled tree, where each node in the tree corresponds to an element in the document and the node label corresponds to the tag name of the element. The hierarchical structure represents the nesting relationship

between the elements. The content of each element can be modeled using its *full content* or *weighted combination of leaf contents*. The full content of an element consists of all the text contained in the subtree rooted at the element, while the leaf content of an element consists of all the text directly contained in the element. The weighted combination of leaf contents of an element refers to the weighted combination of the leaf contents of all the elements in the subtree rooted at the element. The full content is in fact a special case of weighted combination of leaf contents, where all the weights are equal to 1. In this paper, we represent the element content by its full content. How to set optimal weights for combining leaf contents to improve the retrieval performance is orthogonal to the techniques addressed in this paper, and worth further study.

The basic idea of language modeling approaches in information retrieval is to estimate a language model for each document (θ_D) and the query (θ_Q), and then rank the document in one of the two ways: by estimating the probability of generating the query string with the document language model, i.e. $P(Q|\theta_D)$, as in equation (1), or by computing the Kullback-Leibler divergence of the query language model from the document language model, i.e. $D(\theta_Q \parallel \theta_D)$, as in equation (2).

$$P(Q|\theta_D) = \sum_{w \in Q} P(w|\theta_D). \quad (1)$$

$$\begin{aligned} -D(\theta_Q \parallel \theta_D) &= -\sum_{w \in V} P(w|\theta_Q) \log \frac{P(w|\theta_Q)}{P(w|\theta_D)}. \\ &\propto \sum_{w \in V} P(w|\theta_Q) \log P(w|\theta_D) \end{aligned} \quad (2)$$

On the surface, the KL-divergence model appears to be quite different from the query likelihood method. However, it turns out that the KL-divergence model covers the query likelihood method as a special case when we use the empirical distribution to estimate the query language model, i.e. maximum-likelihood estimate. By introducing the concept of query language model, the KL-divergence model offers opportunities of leveraging feedback information to improve retrieval accuracy. This can be done by re-estimating the query language model with the feedback information [12]. In this paper, we do not consider the effect of feedback information. So we adopt the query likelihood method in our experiments.

Thus, the entire retrieval problem is reduced to the problem of estimating document language models. The most direct way to estimate a language model given some observed text is to use the maximum likelihood estimate, assuming an underlying multinomial model. However, the maximum likelihood estimate assigns zero probability to the unseen words. This is clearly undesirable. Smoothing plays a critical role in language modeling approaches to avoid assigning zero probability to unseen words and also to improve the accuracy of estimated language models in general. Traditionally, most smoothing methods mainly use the global collection information to smooth a document language model [13][14]. Recently, corpus graph structures, e.g. the similarity between documents, have been exploited to provide

more accurate smoothing of document language models [15]. Such a local smoothing strategy has been shown to be effective.

In XML retrieval, the element language model is usually smoothed by interpolating it with the global information such as the whole collection model or the language model specific to the element type, and further more, possibly with other related element language models in the tree structure, e.g. its parent, children, or even descendants and ancestors if the smoothing is done recursively on the tree [8][9][10][11]. However, according to the previous study [10][16], the recursive smoothing methods exploiting the hierarchical structure of the XML tree only improve the retrieval effectiveness slightly. Thorough experimental study on effective smoothing methods for XML retrieval is needed, and we leave it to our future work. As for the baseline approaches in this paper, we adapt the two-stage smoothing method proposed in [14] to XML retrieval as it was shown to be effective in our previous experiments [16]. In the first stage, the element language model is smoothed using a Dirichlet prior with the document language model as the reference model. In the second stage, the smoothed element language model is further interpolated with a query background model. With no sufficient data to estimate the query background model, the collection language model is assumed to be a reasonable approximation of the query background model. Thus, we get the estimation of each element language model as shown in equation (3), where $tf(w, e)$ is the term frequency of w in the element e , $len(e)$ is length of e , μ is the scale parameter for Dirichlet smoothing and λ is the interpolation parameter for Jelinek-Mercer smoothing.

$$P(w | \theta_e) = (1 - \lambda) \frac{tf(w, e) + \mu \cdot P(w | \theta_D)}{len(e) + \mu} + \lambda P(w | \theta_C). \quad (3)$$

2.1 CO Queries

CO queries at INEX ad hoc track are given in the title fields of topics. We remove all the signs, i.e. +, -, and quotes, i.e. "" in the title field. That is, a CO query is simply a bag of keywords in our system, $Q = \{w_1, w_2, \dots, w_m\}$. We estimate a language model for each element in the collection using its full content and two-stage smoothing method. Each element is scored independently based on the query likelihood as stated above, and a ranked list of elements is returned.

We submitted results to the four tasks of the ad hoc track, i.e. focused, best in context, relevant in context and thorough. For the first three tasks, overlap in the result list has to be removed. We adopt the simplest strategy of removing overlap, i.e. keeping only the highest ranked element on each path. For the in context tasks, all the elements from the same document are clustered together, and the clusters (corresponding to documents) are ordered by their maximum element scores. For the best in context task, all the elements except the max-scored element in each document are removed from the result list. That is, a ranked list of documents is returned for best in context task. The best entry point in each document is set to be the max-scored element in the document. For the thorough task, no overlapping is removed from the result list.

2.2 CAS Queries

In INEX ad hoc track, CAS queries are given in the castitle fields of topics. The queries are expressed in the NEXI query language [3]. For example, consider the CAS query of topic 85,

```
//article[about(., operating system)]//sec[about(./p,  
mutual exclusion)]
```

which requests section components in which some paragraph is about “mutual exclusion” and such sections are in an article about “operating system”. There can be strict and vague interpretations for the structural constraints expressed in the query. As how to vaguely interpret the structural constraints properly remains a challenge problem, in this paper for the comparisons, we adopt the strict interpretation strategy as implemented in the Lemur/Indri system [4].

When evaluating the above example query, for each section that appears in an article, a score depending on its relevance to the query condition *about(./p, mutual exclusion)* is first calculated; to calculate this score, a list of relevance scores of all paragraphs in the section to the keyword query “*mutual exclusion*” are computed using the query likelihood method, and then the section’s score is set to be its best matching paragraph’s score. Finally, this score is combined (probabilistic AND) with the relevance score of the article containing this section to the keyword query “*operating system*” to form the final score of the section. A ranked list of sections is returned based on the final scores.

As in CO queries, we ignore all the phrases and +, - signs in CAS queries. Removing overlaps and presenting the results as required in context tasks are handled in the same way as that for CO queries as described in Section 2.2.

3 Exploiting Semantic Tags in Evaluating CO Queries

With the semantically rich tags present in the XML document, we assume that providing more structural information in queries could be more effective, as discussed in previous studies [1]. However, most users are not willing or not good at providing such structural conditions in queries. Many users in general are only willing to submit the simplest forms of queries, i.e. keyword queries.

To assist its users, an XML retrieval system can shift the burden of specifying effective structured queries from the user to the system. That is, the system automatically infers or generates CAS queries from the CO queries submitted by users [17]. This process is typically divided into three steps: first, generating all possible structured queries from the input unstructured query by incorporating the knowledge of schemas, data statistics, heuristics, user/pseudo relevance feedback and etc.; second, ranking the structured queries according to their likelihood of matching user’s intent; third, selecting the top-k queries and evaluating them. However, if the inferred structured queries are not intended by the user, we can not expect to get the right result. Another line of research work done in this direction is to infer some structural hints, not necessarily complete structured queries, from the input keyword query, such as in [18][19].

In this paper, we investigate the problem of how to exploit the semantic tag information to improve the performance of CO queries not from the point of view of modifying queries but from the point of view of enriching element representations with semantic tags. The idea is similar to that of enriching web document representations with aggregated anchor text [20], even though we are in different settings, web documents versus XML elements and aggregated anchor text versus semantic tags. Although enriching XML element representations with the semantic tags of all its related elements or even the ones from other linked documents would be an interesting research issue, we leave it to our future work. In this paper we investigated two ways of enriching element content representations with its own semantic tags. One is text representation and the other is new field representation.

3.1 Text Representation

When a user issues a keyword query, it often contains keywords matching the semantic tags of the relevant elements, e.g. to specify the target information type or to specify the context of search terms. For example, among many others, topic 5 “chemists physicists scientists alchemists periodic table elements” is looking for *chemists, physicists, scientists, alchemists* who studied elements and the periodic table, and topic 36 “notting hill film actors” requests all the actors starring in the film “Notting Hill”, where *actor* gives the target information type and *film* specifies the search context of “Notting Hill”.

Thus, the terms in a keyword query has to match not only the text content of an element but also the semantic tags. The simplest way of enriching the element representation to match the query with both the content and semantic tags of the element is to propagate all the semantic tags of the element to its raw text content. Note that there could be more than one semantic tags added to the raw text content of an element. For example, in Fig. 1, the text content of the *article* element will be augmented with additional terms, *article, physical_entity, communicator, person, causal_agent, writer, dramatist*; the text content of the second link element will be augmented to be “*region, administrative-district, location, district, country, link, Edo State*”.

There are altogether 32311 distinct tags in the Wikipedia collection of INEX 2009. Only a small portion of them (less than 0.1%) are for structural or formatting uses, e.g. *article, sec, p, bdy, b, it*, while all others are of semantic use. Among all the semantic tags, only about 5245 of them are from the WordNet concepts, i.e. with the “*wordnetid*” attribute in the start tag. Since most tags have semantic meanings, we did not differentiate them, but chose to propagate all of them to their respective elements.

3.2 New Field Representation

When evaluating a query, it may be helpful if we assign different weights to the case when a search term matches a semantic tag and to the case when it matches the raw text of an element. This can be achieved by the new field representation, in which all the semantic tags of an element are added to a new subelement of this element. The

new subelement is given special tag name “*semantics*”, which is not among the existing 32311 distinct tag names.

When evaluating a CO query over the new field representation, the generative model for each element is first estimated using the equation (3). Next, the model for each element is further smoothed by interpolating it with all its children’s smoothed language models as shown in equation (4). This can be done non-recursively or recursively from the bottom up to the top of the tree as discussed in the hierarchical language modeling approaches [10][11]. If by non-recursive smoothing, $P(w|\theta_c)$ in equation (4) should be changed to $P(w|\theta_c)$.

$$P(w|\theta_e) = \frac{1}{\lambda_e + \sum_{c \in \text{children}(e)} \lambda_c} [\lambda_e P(w|\theta_e) + \sum_{c \in \text{children}(e)} \lambda_c P(w|\theta_c)] \quad (4)$$

In such a model, we can set different weights for different fields. By default, all the fields have equal weights that are equal to 1. To stress the “*semantics*” field, we can set a higher weight on it, e.g. in our experiments, we set $\lambda_{\text{semantics}}$ to be 2. In this representation, the matching of terms with semantic tags and with raw text can be differently weighted, which may be useful.

4 Experiments

To compare CO and CAS queries and evaluate the methodology we proposed to exploit semantic tags in executing CO queries, we planned to submit four runs for each task at the ad hoc track of INEX 2009. These four runs, named as *base_co*, *base_cas*, *text_co* and *semantics_co*, correspond to the baseline CO, baseline CAS, text representation, and new field representation approaches respectively. Due to the limit of time, not all runs were successfully submitted before the deadline. We ignore the Thorough task as it may have similar results as the Focused task and present the results on the other three tasks in this section.

4.1 Setup

We implemented the four retrieval strategies inside the Lemur/Indri IR system [5], which is based on language modeling approaches. Since in this paper we intend to investigate whether the semantic tags in XML documents could be useful, and many other retrieval strategies are orthogonal to our approaches, e.g. by incorporating the element length priors, the proximity of keywords, contextualization and etc., we did not employ them in this set of experiments. We expect the retrieval performance be further improved by incorporating the above mentioned techniques.

At the ad hoc track of INEX 2009, the data collection consists of 2,666,190 semantically annotated Wikipedia XML documents. There are more than 762M elements and 32311 distinct tags in the collection. We indexed all the elements in the XML documents, and the index was built using the Krovetz stemmer and the shortest

list of stop words that contains only three words {"a", "an", "the"}. To make Lemur/Indri possible to index 32311 different fields, we made changes on its internal data structures. Among the list of 32311 tags, there are many cases that two tags only differ on the letter cases, e.g. "Mission" and "mission", or a list of tags share the same prefix while their different suffixes form a sequence of numbers, like "country1", "country2", "country3", "country4", and etc. We specify some rules to conflate these tags when indexing the documents. For example, all the tags with capital letters are conflated to its small cases version, and multiple tags with same word prefix but different number suffixes are conflated to their shared word prefix. Thus, "country1", "country2", and etc. are all conflated to "country". This can improve the retrieval performance of CAS queries when matching the tag names exactly. So this has similar effect as the tag equivalence strategy, but is done at the indexing time.

The parameters in the retrieval models are set as its default values in Lemur/Indri system, e.g. μ and λ in equation (3) are set to be 2500 and 0.4 respectively, and we set the $\lambda_{semantics}$ in equation (4) to be 2 while all other λ s in equation (4) are set to be 1. For each run, our system returns the top 1500 elements.

The measures used in INEX 2009 are the same as that in INEX 2007. For Focused task, interpolated precisions at 101 recall levels, i.e. $iP(i)$, $i=0.0, 0.01, 0.02, \dots, 1.0$, and mean average interpolated precision ($MAiP$) are computed. For In-Context tasks, generalized precisions and recalls at different ranks, i.e. $gP[r]$ and $gR[r]$, $r=1, 2, 3, 5, 10, 25, 50$, and mean average generalized precision ($MAgP$) are computed.

4.2 Results

The results for the four approaches at different tasks are shown in Table 1. The results annotated with "*" are not official runs submitted to the INEX 2009 but were evaluated with the INEX 2009 assessments afterwards. Due to some implementation issues, we did not finish all the semantics_co runs. However, from the results of best in context task, we can draw similar conclusions for this approach in other tasks. It does not perform better than other approaches, especially the text representation approach. This may be due to that we did not tune the parameter $\lambda_{semantics}$, and we leave it in our future work.

Table 1. Experimental results of different retrieval strategies at INEX 2009.

Tasks Runs	Focused ($iP[0.01]$)	Focused ($MAiP$)	Relevant in context ($gP[10]$)	Relevant in context ($MAgP$)	Best context ($gP[10]$)	Best context ($MAgP$)
base_co	0.4322	0.0565	0.1934	0.0645	0.1465*	0.0958*
base_cas	0.4876	0.1472	0.1946	0.1028	0.1444*	0.0971*
text_co	0.4973	0.0741	0.2381	0.0807	0.1610	0.1013
semantics_co	-	-	-	-	0.1484	0.0923

From Table 1, we can make the following observations. The text representation approach is useful in retrieving the relevant results earlier, while the baseline CAS

retrieval strategy, i.e. strict interpretation of the structural constraints, performs better on average precisions.

To analyze these observations more deeply, we classify the 68 assessed CAS queries in INEX 2009 into three categories:

1. CAS queries with no structural constraints, i.e. identical to CO queries. For example, *//*[about(., Bermuda Triangle)]*. There are 15 of them.
2. CAS queries with only structural tags. For example, *//article[about(., Nobel prize)]*, *//article[about(., IBM)]//sec[about(., computer)]*, and *//article[about(., operating system)]//sec[about(./p, mutual exclusion)]*. There are 25 of them.
3. CAS queries with semantic tags. For example, *//vehicles[about(., fastest speed)]*, *//article[about(., musician)]//music_genre[about(., Jazz)]*, and *//article[about(./language, java) OR about(., sun)]//sec[about(./language, java)]*. There are 28 of them.

For each class of CAS queries, we compared their results as shown in Table 2, Table 3 and Table 4 respectively.

Table 1. Results over the CAS queries with no structural constraints.

Tasks Runs	Focused (<i>iP[0.01]</i>)	Focused (<i>MAiP</i>)	Relevant context (<i>gP[10]</i>)	Relevant context (<i>MAgP</i>)	Best context (<i>gP[10]</i>)	Best context (<i>MAgP</i>)
base_co	0.4864	0.0718	0.2263	0.0814	0.1194*	0.0923*
base_cas	0.4865	0.0718	0.2263	0.0814	0.1194*	0.0933*
text_co	0.5302	0.0974	0.2818	0.1024	0.1219	0.0855
semantics_co	-	-	-	-	0.1039	0.0752

For the first class of CAS queries, CAS queries are identical to their CO versions. From Table 2, we can observe that text representation performs better than the baseline CO/CAS queries.

Table 3. Results over the CAS queries with only structural tags.

Tasks Runs	Focused (<i>iP[0.01]</i>)	Focused (<i>MAiP</i>)	Relevant context (<i>gP[10]</i>)	Relevant context (<i>MAgP</i>)	Best context (<i>gP[10]</i>)	Best context (<i>MAgP</i>)
base_co	0.4170	0.0580	0.1807	0.0669	0.1629*	0.0996*
base_cas	0.5582	0.2296	0.2054	0.1242	0.1627*	0.1099*
text_co	0.4545	0.0771	0.2032	0.0814	0.1778	0.1099
semantics_co	-	-	-	-	0.1675	0.1042

For the second class of CAS queries, baseline CAS approach performs better than other approaches both in terms of early precisions and in terms of average precisions. This may be because that for querying INEX Wikipedia collection, most of time the whole article is relevant. CAS queries constraining the results to be articles or

sections would avoid returning many small subelements, thus returning more relevant components in the top 1500 returned elements.

Table 4. Results over the CAS queries with semantic tags.

Tasks	Focused (iP[0.01])	Focused (MAiP)	Relevant context (gP[10])	Relevant context (MAgP)	Best context (gP[10])	Best context (MAgP)
base_co	0.4166	0.0648	0.1872	0.0534	0.1465*	0.0942*
base_cas	0.4155	0.1141	0.1679	0.0951	0.1415*	0.0876*
text_co	0.5179	0.0589	0.2459	0.0684	0.1670	0.1020
semantics_co	-	-	-	-	0.1553	0.0908

For the third class of queries, we can observe the same trend as on the whole query set. The reason why CAS queries are not always beneficial may be that if the CAS query is badly formed, strict matching of the query would hurt the performance greatly. For example, *//*[about(., notting hill actors) AND about(./category, film)]* does not clarify that “notting hill” should occur in the “film” context and “actor” be the desired result element type, so the baseline CAS performance for this query is much worse than other approaches. Another example is *//article[about(., rally car)]//driver[about(., female) OR about(., woman)]*, the user submit this query intends to retrieve “cars”, however the badly formulated CAS queries will return “drivers”. To avoid such problems, it is better for the system to approximately match the CAS queries and to infer some hints from user submitted CO queries instead of asking the user to specify these hints. It is hard for them to formulate good queries.

5 Conclusions and Future Work

In this paper, we did experiments over the semantically annotated Wikipedia XML corpus at INEX 2009 ad hoc track, attempting to investigate the two research questions:

1. Do the structural constraints in CAS queries help in retrieving an XML document collection containing semantically rich tags?
2. How to exploit the semantic tag information to improve the CO queries as most users prefer to express the simplest forms of queries?

The results show that CAS queries are helpful in retrieving more relevant elements. When the query was badly formed, however, the performance could be hurt greatly. The simplest way of enriching element representations with semantic tags can improve the performance slightly, especially in terms of the early precisions.

As our future work, we are going to study how to infer structural hints from CO queries and match them with data approximately. We are also interested in how to evaluate top-k queries with complex scoring models efficiently.

6 Acknowledgements

The research work is supported by the 863 High Tech. Project of China under Grant No. 2009AA01Z149.

References

1. J. Chu-Carroll, J. Prager, K. Czuba, D. Ferrucci, P. Duboue, "Semantic Search via XML Fragments: A High-Precision Approach to IR", SIGIR 2006.
2. D. Carmel, Y.S. Maarek, M. Mandelbrod, et al., "Searching XML documents via XML fragments", SIGIR 2003.
3. A. Trotman and B. Sigurbjoernsson, "Narrowed Extended XPath I (NEXI)", INEX 2004.
4. Lemur/Indri. <http://www.lemurproject.org>.
5. XQuery Full-Text. <http://www.w3.org/TR/xpath-full-text-10/>.
6. A. Trotman and M. Lalmas, "Why Structural Hints in Queries do not Help XML-Retrieval?", SIGIR 2006.
7. R. Schenkel, F. Suchanek, G. Kasneci, "YAWN: A Semantically Annotated Wikipedia XML Corpus", BTW 2007.
8. D. Hiemstra, "Statistical Language Models for Intelligent XML Retrieval", Intelligent Search on XML data, H. Blanken et al. (Eds.), 2003.
9. P. Ogilvie, J. Callan, "Language Models and Structured Document Retrieval", INEX 2003.
10. P. Ogilvie, J. Callan, "Hierarchical Language Models for XML Component Retrieval", INEX 2004.
11. P. Ogilvie, J. Callan, "Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval", INEX 2005.
12. C. Zhai, "Statistical Language Models for Information Retrieval: A Critical Review", Foundations and Trends in Information Retrieval, Vol. 2, No. 3 (2008).
13. C. Zhai, J. Lafferty, "A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval", SIGIR 2001.
14. C. Zhai, J. Lafferty, "Two-Stage Language Models for Information Retrieval", SIGIR 2002.
15. Q. Mei, D. Zhang, C. Zhai, "A General Optimization Framework for Smoothing Language Models on Graph Structures", SIGIR 2008.
16. Q. Wang, Q. Li, S. Wang, "Preliminary Work on XML Retrieval", Pre-Proceedings of INEX 2007.
17. D. Pektova, W.B. Croft, Y. Diao, "Refining Keyword Queries for XML Retrieval by Combining Content and Structure", ECIR 2009.
18. J. Kim, X. Xue, W.B. Croft, "A Probabilistic Retrieval Model for Semistructured Data", ECIR 2009.
19. Z. Bo, T.W. Ling, B. Chen, J. Lu, "Effective XML Keyword Search with Relevance Oriented Ranking", ICDE 2009.
20. D. Metzler, J. Novak, H. Cui, S. Reddy, "Building Enriched Document Representations using Aggregated Anchor Text", SIGIR 2009.

Overview of the INEX 2009 Book Track

Gabriella Kazai¹, Antoine Doucet², Marijn Koolen³, and Monica Landoni⁴

¹ Microsoft Research, United Kingdom
gabkaz@microsoft.com

² University of Caen, France
doucet@info.unicaen.fr

³ University of Amsterdam, Netherlands
m.h.a.koolen@uva.nl

⁴ University of Lugano
monica.landoni@unisi.ch

Abstract. This paper provides an overview of the INEX 2009 Book Track. The main goal of the track is to evaluate approaches for supporting users in reading, searching, and navigating the full texts of digitized books. The investigation is focused around four tasks: 1) the Book Retrieval task aims at comparing traditional and book-specific retrieval approaches, 2) the Focused Book Search task evaluates focused retrieval approaches for searching books, 3) the Structure Extraction task tests automatic techniques for deriving structure from OCR and layout information, and 4) the Active Reading task aims to explore suitable user interfaces for eBooks enabling reading, annotation, review, and summary across multiple books. We report on the setup and status of the track.

1 Introduction

The INEX Book Track was launched in 2007, prompted by the numerous mass-digitization projects [1], e.g., the Million Book project⁵, the Open Content Alliance⁶, and the Google Books Library project⁷. As a result of these efforts the full texts of digitized books have become available by the thousands on the Web and in digital libraries. The unprecedented scale of these efforts, the unique characteristics of the digitized material, as well as the unexplored possibilities of user interactions present exciting research challenges and opportunities, see e.g. [3].

The overall goal of the INEX Book Track is to promote inter-disciplinary research investigating techniques for supporting users in reading, searching, and navigating the full texts of digitized books and to provide a forum for the exchange of research ideas and contributions. Toward this goal, the track set up tasks to provide opportunities for investigating research questions around three broad topics:

- IR techniques for searching collections of digitized books,

⁵ <http://www.ulib.org/>

⁶ www.opencontentalliance.org/

⁷ <http://books.google.com/>

- Users’ interactions with eBooks and collections of digitized books,
- Mechanisms to increase accessibility to the contents of digitized books.

Based around these main themes, four specific tasks were defined:

1. The Book Retrieval (BR) task, framed within the user task to build a reading list for a given topic of interest, aimed at comparing traditional document retrieval methods with domain-specific techniques exploiting book-specific features, such as the back of book index or associated metadata, like library catalogue information,
2. The Focused Book Search (FBS) task aimed to test the value of applying focused retrieval approaches to books, where users expect to be pointed directly to relevant book parts,
3. The Structure Extraction (SE) task aimed to evaluate automatic techniques for deriving structure from OCR and layout information for building hyper-linked table of contents, and
4. The Active Reading task (ART) aimed to explore suitable user interfaces enabling reading, annotation, review, and summary across multiple books.

In this paper, we discuss the setup and current status of each of these tasks at INEX 2009. First, in Section 2, we give a brief summary of the participating organisations. In Section 3, we describe the corpus of books that forms the basis of the test collection. The following three sections detail the four tasks: Section 4 summarises the BR and FBS tasks, Section 5 reviews the SE task, and Section 6 discusses ART. We close in Section 7 with a summary and further plans.

2 Participating Organisations

A total of 84 organisations registered for the track (up from 54 in 2008, and 27 in 2007), of which 15 took part actively throughout the year (same as in 2008, and up from 9 in 2007), see Table 1. For the full list of participants, please refer to the INEX web site at <http://www.inex.otago.ac.nz/people/participants.asp>.

In total, 7 groups contributed 16 search topics with a total of 37 aspects, 4 groups submitted runs to the Structure Extraction task, 3 to the Book Retrieval task, and 3 groups submitted runs to the Focused Book Search task. Two groups participated in the Active Reading task, but did not submit results.

3 The Book Corpus

The track builds on a collection of 50,239 digitized out-of-copyright books⁸, digitized by Microsoft. The corpus is made up of books of different genre, including history books, biographies, literary studies, religious texts and teachings, reference works, encyclopedias, essays, proceedings, novels, and poetry. 50,099 of

⁸ The collection, although in a different XML format, can also be found on the Internet Archive.

ID	Organisation	Topics	Runs	Assessed topics
6	University of Amsterdam	8, 11	2 BR, 4 FBS	
7	Oslo University College	1, 2	10 BR, 10 FBS	
14	University of California, Berkeley		10 BR, ART	
41	University of Caen	7, 9	3 SE	SE
43	Xerox Research Centre Europe		3 SE	SE
52	Kyungpook National University	3, 4	ART	
54	Microsoft Research Cambridge	10, 16		
78	University of Waterloo	5, 6	4 FBS	
86	University of Lugano	12, 13, 14, 15		
125	Microsoft Development Center Serbia		1 SE	
335	Fraunhofer IAIS			SE
339	Universita degli Studi di Firenze			SE
343	Noopsis Inc.		1 SE	
471	Peking University, ICST			SE

Table 1. Active participants of the INEX 2009 Book Track, contributing topics, runs, and/or relevance assessments (BR = Book Retrieval, FBS = Focused Book Search, SE = Structure Extraction, ART = Active Reading Task)

the books also come with an associated MACHine-Readable Cataloging (MARC) record, which contains publication (author, title, etc.) and classification information.

The OCR text of the books has been converted from the original DjVu format to an XML format referred to as BookML, developed by Microsoft Development Center Serbia. BookML provides additional structure information, including markup for table of contents entries. The basic XML structure of a typical book in BookML (ocrml.xml file extension) is a sequence of pages containing nested structures of regions, sections, lines, and words ([coords] represents coordinate attributes, defining the position of a bounding rectangle for a region, line or word, or the width and height of a page):

```

<document>
  <page pageNumber='1' label='PT_CHAPTER' [coords] key='0' id='0'>
    <region regionType='Text' [coords] key='0' id='0'>
      <section label='SEC_BODY' key='408' id='0'>
        <line [coords] key='0' id='0'>
          <word [coords] key='0' id='0' val='Moby' />
          <word [coords] key='1' id='1' val='Dick' />
        </line>
        <line [...]><word [...] val='Melville' />[...]</line>[...]
```

BookML provides a set of labels (as attributes) indicating structure information in the full text of a book and additional marker elements for more

complex texts, such as a table of contents. For example, the label attributes in the XML extract above indicate that a new chapter starts on page 1 (label="PT_CHAPTER") and that the section element is part of the main body of text on the page (label="SEC_BODY"). Other semantic units include headers (SEC_HEADER), footers (SEC_FOOTER), back of book index (SEC_INDEX), table of contents (SEC_TOC). A page may be labeled as a table of contents page (PT_TOC), an empty page (PT_EMPTY), a back of book index page (PT_INDEX), or as a new chapter page (PT_CHAPTER), etc. Marker elements provide detailed markup, e.g., for table of contents, indicating entry titles (TOC_TITLE), and page numbers (TOC_CH_PN), etc.

The full corpus, which totals around 400GB, was distributed on USB HDDs (at a cost of 70GBP). In addition, a reduced version (50GB, or 13GB compressed) was made available for download. The reduced version was generated by removing the word tags and propagating the values of the `val` attributes as text content into the parent (i.e., line) elements.

4 Information Retrieval Tasks

Focusing on IR challenges, two search tasks were investigated: 1) Book Retrieval (BR), in which users search for whole books in order to build a reading list on a given topic, and 2) Focused Book Search (FBS), in which users search for information in books on a given topic and expect to be pointed directly at relevant book parts. Both these tasks used the corpus of over 50,000 books described in Section 3, and the same set of test topics (see Section 4.3).

A summary of the tasks, the test topics, and the online relevance assessment system are described in the following sections. The relevance assessment collection phase is not yet underway, thus evaluation results will be published only after the INEX workshop.

4.1 The Book Retrieval (BR) Task

This task was set up with the goal to compare book-specific IR techniques with standard IR methods for the retrieval of books, where (whole) books are returned to the user. The user scenario underlying this task is that of a user searching for books on a given topic with the intent to build a reading or reference list, for example to append at the end of an article, such as a Wikipedia article. The reading list may be for research purposes, or in preparation of lecture materials, or for entertainment, etc.

Participants of this task were invited to submit either single runs or pairs of runs. A total of 10 runs could be submitted, each run containing the results for all 16 test topics. A single run could be the result of either generic (non-specific) or book-specific IR methods. A pair of runs had to contain both types, where the non-specific run served as a baseline, which the book-specific run extended upon by exploiting book-specific features (e.g., back-of-book index, citation statistics, book reviews, etc.) or specifically tuned methods. One automatic run (i.e., using

only the topic title part of a test topic for searching and without any human intervention) was compulsory. A run could contain, for each test topic, a maximum of 1000 books (identified by their 16 character long bookID⁹), ranked in order of estimated relevance.

A total of 22 runs were submitted by 3 groups (2 runs by University of Amsterdam (ID=6); 10 runs by University of California, Berkeley (ID=14); and 10 runs by Oslo University College (ID=7)), see Table 1.

4.2 The Focused Book Search (FBS) Task

The goal of this task was to investigate the application of focused retrieval approaches to a collection of digitized books. The task was thus similar to the INEX ad hoc track's Relevant in Context task, but using a significantly different collection while also allowing for the ranking of book parts within a book. The user scenario underlying this task was that of a user searching for information in a library of books on a given subject. The information sought may be 'hidden' in some books (i.e., it forms only a minor theme) while it may be the main focus of some other books. In either case, the user expects to be pointed directly to the relevant book parts. Following the focused retrieval paradigm, the task of a focused book search system is then to identify and rank (non-overlapping) book parts that contain relevant information and return these to the user, grouped by the books they occur in.

Participants could submit up to 10 runs, where one automatic and one manual run was compulsory. Each run could contain, for each of the 37 topic aspects, a maximum of 1000 books estimated relevant to the given aspect, ordered by decreasing value of relevance. For each book, a ranked list of non-overlapping XML elements, passages, or book page results estimated relevant were to be listed in decreasing order of relevance. A minimum of one book part had to be returned for each book in the ranking. A submission could only contain one type of results, i.e., only XML elements or only passages; result types could not be mixed.

A total of 18 runs were submitted by 3 groups (4 runs by the University of Amsterdam (ID=6); 10 runs by Oslo University College (ID=7); and 4 runs by the University of Waterloo (ID=78)), see Table 1.

4.3 Test Topics

Topics are representations of users' information needs and may comprise of several aspects or sub-topics. An information need may be generic or specific. Reflecting this, a topic may be of varying complexity and may comprise one or multiple aspects or sub-topics. We encouraged participants to create multiple aspects for their topics, where aspects should be focused (narrow) with limited number of relevant book parts (e.g., pages).

⁹ The bookID is the name of the directory that contains the book's OCR file, e.g., A1CD363253B0F403

Participants were encouraged to use Wikipedia at different stages when preparing topics. The intuition behind the introduction of Wikipedia is twofold. First, Wikipedia articles often contain a reading list of books relevant to the general topic of the article, while they also often cite related books relevant to a specific statement in the article. Thus, topics linked to Wikipedia articles have a real world application. Second, we anticipated that browsing through Wikipedia entries could provide participants with suggestions about topics and their specific aspects of interest, and at the same time provide them with insights and relevant terminology to be used for better searches and refinements that should lead to a better mapping between topics and collection.

Participants were asked to create and submit 2 topics, ideally with at least 2 aspects each, using an online Book Search system (see Section 4.4).

A total of 16 new topics (ID: 1-16), containing 37 aspects, were contributed by 7 participating groups (see Table 1). An example topic is shown in Figure 1.

The collected topics were used for retrieval in the BR task, while the topic aspects were used in the FSB task.

4.4 Relevance Assessment System

The Book Search system (<http://www.booksearch.org.uk>), developed at Microsoft Research Cambridge, is an online web service that allows participants to search, browse, read, and annotate the books of the test corpus. Screenshots of the assessment system are shown in Figures 2 and 3.

In 2008, a game called the Book Explorers' Competition was developed to collect relevance assessments, where assessors competed for prizes. The competition involved reading books and marking relevant content inside the books for which assessors were rewarded points [4].

Based on what we learnt in 2008, we are modifying the game this year to consist of two separate stages: 1) In the first stage assessors are asked to find books relevant to the 16 topics and rank the top 10 most relevant books for each topic, then 2) in the second stage, assessors will again compete as explorers and reviewers, providing page level judgements for the 37 topic aspects.

We expect the assessment phase to start in mid December and conclude by the end of January 2010. Results of the evaluation will be published soon after the assessments have been collected.

5 The Structure Extraction (SE) Task

As in 2008, the goal of this task was to test and compare automatic techniques for extracting structure information from digitized books and building a hyper-linked table of contents (ToC). The task was motivated by the limitations of current digitization and OCR technologies that produce the full text of digitized books with only minimal structure markup: Pages and paragraphs are usually identified, but more sophisticated structures, such as chapters, sections, etc., are typically not recognised.

The first round of the structure extraction task, in 2008, ran as a pilot test and permitted to set up appropriate evaluation infrastructure, including guidelines, tools to generate ground truth data, evaluation measures, and a first test set of 100 books. The second round was run both at INEX 2009 and at the International Conference on Document Analysis and Recognition (ICDAR) 2009 [2]. This round built on the established infrastructure with an extended test set of 1,000 digitized books.

Participants of the task were provided a sample collection of 1000 digitized books of different genre and styles in DjVu XML format. Unlike the BookML format of the main corpus, the DjVu files only contain markup for the basic structural units (e.g., page, paragraph, line, and word); no structure labels and markers are available. In addition to the DjVu XML files, participants were distributed the PDF of books.

Participants could submit up to 10 runs, each containing the generated table of contents for the 1000 books in the test set.

A total of 8 runs were submitted by 4 groups (1 run by Microsoft Development Center Serbia (MDCS), 3 runs by Xerox Research Centre Europe (XRCE), 1 run by Noopsis Inc., and 3 runs by the University of Caen).

5.1 Evaluation Measures and Results

For the evaluation of the SE task, the ToCs generated by participants were compared to a manually built ground-truth. This year, the annotation of a minimum number of books was required to gain access to the combined ground-truth set.

To make the creation of the ground-truth set for 1,000 digitized books feasible, we 1) developed a dedicated annotation tool, 2) made use of a baseline annotation as starting point and employed human annotators to make corrections to this, and 3) shared the workload across participants.

The annotation tool was specifically designed for this purpose and developed at the University of Caen, see Figure 4. The tool takes as input a generated ToC and allows annotators to manually correct any mistakes.

Performance was evaluated using recall/precision like measures at different structural levels (i.e., different depths in the ToC). Precision was defined as the ratio of the total number of correctly recognized ToC entries and the total number of ToC entries; and recall as the ratio of the total number of correctly recognized ToC entries and the total number of ToC entries in the ground-truth. The F-measure was then calculated as the harmonic of mean of precision and recall. For further details on the evaluation measures, please see <http://www.inex.otago.ac.nz/tracks/books/INEXBookTrackSEMeasures.pdf>. The ground-truth and the evaluation tool can be downloaded from <http://www.inex.otago.ac.nz/tracks/books/Results.asp#SE>.

The evaluation results are given in Table 2. The best performance ($F = 41.51\%$) was obtained by the MDCS group, who extracted ToCs by first recognizing the page(s) of a book that contained the printed ToC [5]. Noopsis Inc. used a similar approach, although did not perform as well. The XRCE group and the University of Caen relied on title detection within the body of a book.

ParticipantID+RunID	Participant	F-measure
MDCS	MDCS	41.51%
XRCE-run2	XRCE	28.47%
XRCE-run1	XRCE	27.72%
XRCE-run3	XRCE	27.33%
Noopsis	Noopsis	8.32%
GREYC-run1	University of Caen	0.08%
GREYC-run2	University of Caen	0.08%
GREYC-run3	University of Caen	0.08%

Table 2. Evaluation results for the SE task (complete ToC entries)

6 The Active Reading Task (ART)

The main aim of ART is to explore how hardware or software tools for reading eBooks can provide support to users engaged with a variety of reading related activities, such as fact finding, memory tasks, or learning. The goal of the investigation is to derive user requirements and consequently design recommendations for more usable tools to support active reading practices for eBooks. The task is motivated by the lack of common practices when it comes to conducting usability studies of e-reader tools. Current user studies focus on specific content and user groups and follow a variety of different procedures that make comparison, reflection, and better understanding of related problems difficult. ART is hoped to turn into an ideal arena for researchers involved in such efforts with the crucial opportunity to access a large selection of titles, representing different genres and appealing to a variety of potential users, as well as benefiting from established methodology and guidelines for organising effective evaluation experiments.

ART is based on the large evaluation experience of EBONI [6], and adopts its evaluation framework with the aim to guide participants in organising and running user studies whose results could then be compared.

The task is to run one or more user studies in order to test the usability of established products (e.g., Amazon’s Kindle, iRex’s Ilaid Reader and Sony’s Readers models 550 and 700) or novel e-readers by following the provided EBONI-based procedure and focusing on INEX content. Participants may then gather and analyse results according to the EBONI approach and submit these for overall comparison and evaluation. The evaluation is task-oriented in nature. Participants are able to tailor their own evaluation experiments, inside the EBONI framework, according to resources available to them. In order to gather user feedback, participants can choose from a variety of methods, from low-effort online questionnaires to more time consuming one to one interviews, and think aloud sessions.

6.1 Task Setup

Participation requires access to one or more software/hardware e-readers (already on the market or in prototype version) that can be fed with a subset of the INEX book corpus (maximum 100 books), selected based on participants' needs and objectives. Participants are asked to involve a minimum sample of 15/20 users to complete 3-5 growing complexity tasks and fill in a customised version of the EBONI subjective questionnaire, usually taking no longer than half an hour in total, allowing to gather meaningful and comparable evidence. Additional user tasks and different methods for gathering feedback (e.g., video capture) may be added optionally. A crib sheet is provided to participants as a tool to define the user tasks to evaluate, providing a narrative describing the scenario(s) of use for the books in context, including factors affecting user performance, e.g., motivation, type of content, styles of reading, accessibility, location and personal preferences.

Participants are encouraged to integrate questionnaires with interviews and think aloud sessions when possible, and adapt questionnaires to fit into their own research objectives whilst keeping in the remit of the active reading task. We also encourage direct collaboration with participants to help shape the tasks according to real/existing research needs.

Our aim is to run a comparable but individualized set of studies, all contributing to elicit user and usability issues related to eBooks and e-reading.

The task has so far only attracted 2 groups, none of whom submitted any results at the time of writing.

7 Conclusions and plans

The Book Track this year has attracted a lot of interest and has grown considerably from last year. However, active participation remained a challenge for most of the participants who signed up to the track. A reason for this may be the high initial set up costs (e.g., building infrastructure to search books). Most tasks also require advance planning and preparations, e.g., for setting up a user study. At the same time, the Structure Extraction task run at ICDAR 2009 (International Conference on Document Analysis and Recognition) has been met with great interest and created a specialist community.

Our immediate plans are to commence the relevance assessment gathering stage for the BR and FBS tasks from mid December. We aim to have the evaluation results published by mid February 2010.

Our plans for the longer term future are to work out ways in which the initial participation costs can be reduced, allowing more of the 'passive' participants to take an active role.

Acknowledgements

The Book Track in 2008 was supported by the Document Layout Team of Microsoft Development Center Serbia. The team contributed to the track by pro-

viding the BookML format and a tool to convert books from the original OCR DjVu files to BookML. They also contributed to the Structure Extraction task by helping us prepare the ground-truth data and by developing the evaluation tools.

References

1. K. Coyle. Mass digitization of books. *Journal of Academic Librarianship*, 32(6):641–645, 2006.
2. Antoine Doucet, Gabriella Kazai, Bodin Dresevic, Aleksandar Uzelac, Bogdan Radakovic, and Nikola Todic. ICDAR 2009 Book Structure Extraction Competition. In *Proceedings of the Tenth International Conference on Document Analysis and Recognition (ICDAR'2009)*, pages 1408–1412, Barcelona, Spain, July 2009.
3. Paul Kantor, Gabriella Kazai, Natasa Milic-Frayling, and Ross Wilkinson, editors. *BooksOnline '08: Proceeding of the 2008 ACM workshop on Research advances in large digital book repositories*, New York, NY, USA, 2008. ACM.
4. Gabriella Kazai, Natasa Milic-Frayling, and Jamie Costello. Towards methods for the collective gathering and quality control of relevance assessments. In *SIGIR '09: Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, 2009.
5. Aleksandar Uzelac, Bodin Dresevic, Bogdan Radakovic, and Nikola Todic. Book layout analysis: TOC structure extraction engine. In Shlomo Geva, Jaap Kamps, and Andrew Trotman, editors, *INEX*, Lecture Notes in Computer Science. Springer Verlag, Berlin, Heidelberg, 2009.
6. R. Wilson, M. Landoni, and F. Gibb. The web experiments in electronic textbook design. *Journal of Documentation*, 59(4):454–477, 2003.

```

<topic id='10' cn_no='60'>
<task>Find relevant books and pages to cite from the Wikipedia article on
Cleopatra's needle</task>
<title>Cleopatra needle obelisk london paris new york</title>
<description>I am looking for reference material on the obelisks known as
Cleopatra's needle, three of which have been erected: in London,
Paris, and New York.</description>
<narrative>I am interested in the obelisks' history in Egypt, their transportation,
their physical descriptions, and current locations. I am, however, not
interested in the language of the hieroglyphics.</narrative>
<wikipedia-title>Cleopatra's needle</wikipedia-title>
<wikipedia-url>http://en.wikipedia.org/wiki/Cleopatra's_Needle</wikipedia-url>
<wikipedia-text>Cleopatra's Needle is the popular name for each of three Ancient
Egyptian obelisks [...] </wikipedia-text>
<aspect aspect_id='10.1'>
<aspect-title>Description of the London and New York pair</aspect-title>
<aspect-narrative>I am looking for detailed physical descriptions of the London and
New York obelisks as well as their history in Egypt. When and
where they were originally erected and what happened to them when
they were moved to Alexandria.</aspect-narrative>
<aspect-wikipedia-text>The pair are made of red granite, stand about 21 meters
(68 ft) high, weigh [...] </aspect-wikipedia-text>
</aspect>
<aspect aspect_id='10.2'>
<aspect-title>London needle</aspect-title>
<aspect-narrative>I am interested in details about the obelisk that was moved to
London. When and where was it moved, the story of its
transportation. Information and images of the needle and the two
sphinxes are also relevant.</aspect-narrative>
<aspect-wikipedia-text>The London needle is in the City of Westminster, on the
Victoria Embankment [...] </aspect-wikipedia-text>
</aspect>
<aspect aspect_id='10.3'>
<aspect-title>New York needle</aspect-title>
<aspect-narrative>I am looking for information and images on the obelisk that was
moved to New York. Its history, its transportation and
description of its current location.</aspect-narrative>
<aspect-wikipedia-text>The New York needle is in Central Park. In 1869, after the
opening of the Suez Canal, [...] </aspect-wikipedia-text>
</aspect>
<aspect aspect_id='10.4'>
<aspect-title>Paris needle</aspect-title>
<aspect-narrative>Information and images on the Paris needle are sought. Detailed
description of the obelisk, its history, how it is different from
the London and New York pair, its transportation and current
location are all relevant.</aspect-narrative>
<aspect-wikipedia-text>The Paris Needle (L'aiguille de Cleopatre) is in the Place
de la Concorde. The center [...] </aspect-wikipedia-text>
</aspect>
</topic>

```

Fig. 1. Example topic from the JNEX 2009 Book Track test set.



Fig. 2. Screenshot of the relevance assessment module of the Book Search system: List of books in the assessment pool for a selected topic.

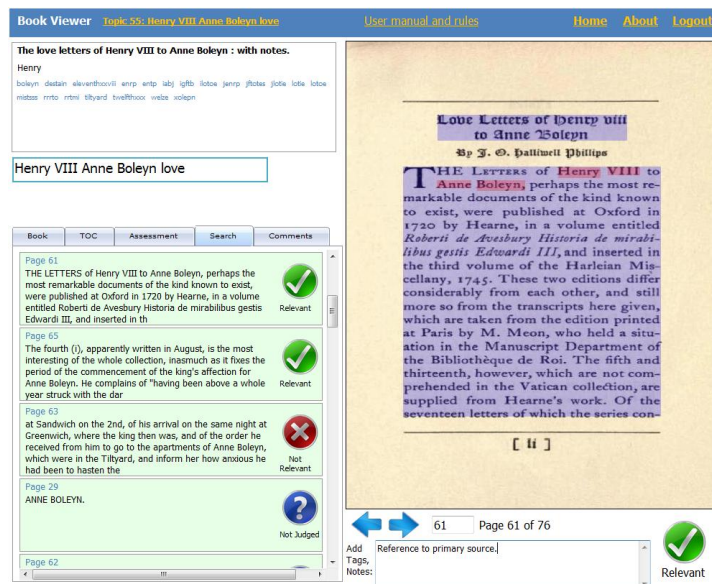


Fig. 3. Screenshot of the relevance assessment module of the Book Search system: Book Viewer window with Assessment tab showing, listing pooled pages to judge.

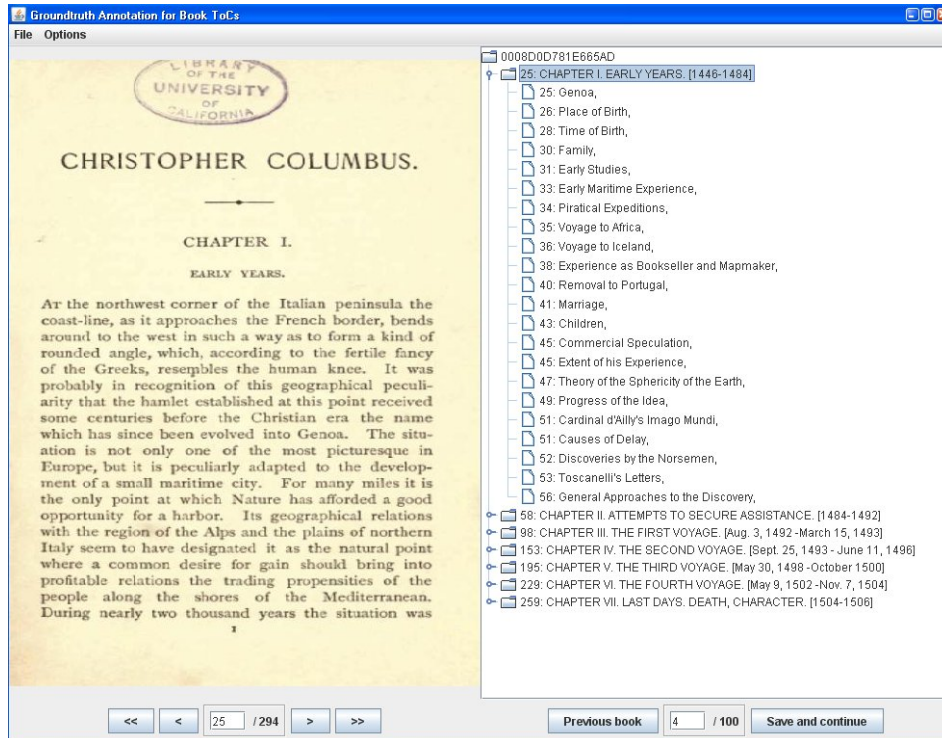


Fig. 4. A screenshot of the ground-truth annotation tool. In the application window, the right-hand side displays the baseline ToC with clickable (and editable) links. The left-hand side shows the current page and allows to navigate through the book. The JPEG image of each visited page is downloaded from the INEX server at www.booksearch.org.uk and is locally cached to limit bandwidth usage.

A Methodology for Producing Improved Focused Elements

Carolyn J. Crouch, Donald B. Crouch, Dinesh Bhirud, Pavan Poluri,
Chaitanya Polumetla, Varun Sudhakar

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

Abstract. This paper reports the results of our experiments to produce superior (i.e., highly ranked) focused elements in response to the Focused Task of the INEX Ad Hoc Track. The results of these experiments, performed using the 2008 INEX collection, confirm that our current methodology (described herein) is able to produce such elements for this collection. Our goal for 2009 is to apply this methodology to the new, extended 2009 INEX collection to determine its viability in this environment. (These experiments are currently underway.) Our system uses our method for dynamic element retrieval [2], working with the semi-structured text of Wikipedia [3], to produce a rank-ordered list of elements in the context of focused retrieval. It is based on the Vector Space Model [7]; basic functions are performed using the Smart experimental retrieval system [6]. Experimental results are reported for both the 2008 and 2009 INEX Ad Hoc Tracks.

1. Introduction

In 2008, our INEX investigations centered on integrating our methodology for the dynamic retrieval of XML elements [2] with traditional article retrieval to facilitate in particular the Focused Task of the Ad Hoc Track. Our goal was to produce what we refer to as *good focused elements*—i.e., elements which when evaluated were competitive with others in the upper ranges of the 2008 rankings. Our best results for these experiments, as reported in [4], accomplished this goal but also

produced indications that further, significant overall improvements were possible with more investigation. Thus, our efforts in 2009 are directed at producing not merely good but rather exceptionally good focused elements (which rank above those listed in the official rankings). This paper reports the results of these experiments, as performed using the INEX 2008 collection. Our goal for 2009 is to produce equivalent results—i.e., superior focused elements—using the new, larger INEX 2009 collection. But as our system requires tuning, this work is just getting underway at the present time

Dynamic element retrieval—i.e., the dynamic retrieval of elements at the desired degree of granularity—has been the focus of our INEX investigations for some time [2, 3]. We have shown that our method works well for both structured [1] and semi-structured text [3] and that it produces a result identical to that produced by the search of the same query against the corresponding all-element index [5]. In [3], we show that dynamic element retrieval (with terminal node expansion) produces a result considerably higher than that reported by the top-ranked participant for the INEX 2006 Thorough task. However, when the task changed to focused retrieval, our results fell into the mid-range of participant scores [4]. This paper (1) describes our current methodology for producing focused elements and (2) reports the results of experiments run on the INEX 2008 collection which clearly establish that it produces superior focused elements in this environment. The remainder of the paper is directed towards establishing whether the same methodology can be successfully applied in the environment of INEX 2009—i.e., whether it is also able to produce superior focused elements for this much larger, scaled up version of Wikipedia.

References

- [1] Bapat, S.: Improving the results for focused and relevant-in-context tasks. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth, (2008) <http://www.d.umn.edu/cs/thesis/bapat.pdf>
- [2] Crouch, C.: Dynamic element retrieval in a structured environment. *ACM TOIS* 24(4), 437-454 (2006)
- [3] Crouch, C., Crouch, D., Kamat, N., Malik, V., Mone, A. Dynamic element retrieval in the Wikipedia collection. In: Fuhr, N., Kamps, J.,

- Lalmas, M., Trotman, A. (eds.) *INEX 2007*, LNCS, vol. 4862, pp. 70-29 (2008)
- [4] Crouch, C., Crouch, D., Bapat, S., Mehta, S., Paranjape, D.: Finding good elements for focused retrieval. In: Geva, S., Kamps, J., Trotman, A. (eds.) *INEX 2008*, LNCS, vol. 5631, pp. 33-38 (2009)
- [5] Mone, A.: Dynamic element retrieval for semi-structured documents. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth, (2007) <http://www.d.umn.edu/cs/thesis/mone.pdf>
- [6] Salton, G., (ed.): *The Smart Rretrieval System—Experiments in Automatic Document Processing*. Prentice-Hall (1971)
- [7] Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. *Comm. ACM* 18 (11), pp. 613-620 (1975)

Resurgence for the Book Structure Extraction Competition

Emmanuel Giguët*, Alexandre Baudrillart °, Nadine Lucas*

GREYC Cnrs EnsiCaen Caen University
BP 5186 F-14032 CAEN Cedex France
* name.surname@info.unicaen.fr
° abaudril@etu.info.unicaen.fr

Abstract. The GREYC Island team participated in the Structure Extraction Competition part of the INEX Book track for the first time, with the Resurgence software. We used a strategy primarily based on top-down document representation. The main idea is to use a model describing relationships for elements in the document structure. Chapters are represented and implemented by frontiers between chapters. Page is also used. The periphery center relationship is calculated on the entire document and reflected on each page. However, the strong points of the approach are that it deals with the entire document, not only the table of contents (ToC); it handles books without ToCs, and titles that are not represented in the ToC (e. g. preface) ; it is not dependent on lexicon, hence tolerant to OCR errors and language independent ; it is simple and fast.

Introduction

The GREYC Island team participated for the first time in the Book Structure Extraction Competition held at ICDAR in 2009 and part of the INEX evaluations [1]. The Resurgence software was modified to this end. This software was designed to handle various document formats, in order to process academic articles (mainly in pdf format) and news articles (mainly in HTML format) in various text parsing tasks [2]. We decided to join the INEX competition because the team was also interested in handling textbooks.

The experiment was run from pdf documents to ensure the control of the entire process [3]. The evaluation rules were not thoroughly studied, for we wished to see if we were able to handle large corpora of voluminous documents, not to win a competition. We focused on chapter detection.

Very few principles were tested in this experiment. The model used previously for articles is a periphery center dichotomy. The periphery center relationship is calculated on the entire document and reflected on each page. It aims at retrieving the page body in a page, surrounded by margins [2]. In the following we explain our stand and discuss the results on the INEX book corpus.

The Book Structure Extraction Task

Strategy

The strategy in Resurgence is primarily based on document positional representation, rather than starting from the table of contents (ToC). This means that the whole document is considered first. Then document constituents are considered top-down (by successive subdivision), with focus on the middle part (main body). The document is thus the unit that can be broken down ultimately to pages. The main idea is to use a model describing relationships for elements in the document structure. However, for this first experiment, we focused on chapter title detection so that the program detects only one level, i. e. chapter titles.

Chapter title detection throughout the document was conducted using a four-page sliding window. It is used to detect chapter transitions. The underlying idea is that the chapter begins after a blank, at least a blank at the top of page. The half page fill rate is the simple cue used to decide on chapter transition. The beginning of a chapter is detected by one of the two patterns below, where i is the page when a chapter starts:

- top of page $i-2$ and bottom of page equally filled
 - bottom of page $i-1$ less filled than top of page
 - top of page i less filled than bottom of page
 - top of page $i+1$ and bottom of page equally filled
-
- empty page $i-1$
 - top of page i less filled than bottom of page
 - top of page $i+1$ and bottom of page equally filled

Chapter title extraction is made from the first third of the beginning page. We applied rules that ensured that the extracted span of text contained the title. The beginning of the title is detected but the end was not carefully looked for. The title was grossly delineated by a rule allowing a number of lines containing at most 40 words.

Experiment

The program detected only chapter titles. No effort was exerted to find the sub-titles. The three runs were not very different.

Run 1 was based on minimal rules as stated above.

Run 2 was the same + trim (remove white spaces at the beginning and end of the title)

Run 3 was the same + trim + lower case (a rule saying that the lower-case lines should be pruned, when following a would-be title in higher-case).

Commented Results

The entire corpus was handled. The results were equally very bad for the three runs. This was due to a page numbering bug where $p = p-1$. The intriguing value above zero 0,08% came from rare cases where the page contained two chapters (two poems).

Table 1. Book Structure Extraction official evaluation ¹

RunID Participant	F-measure (complete entries)
MDCS Microsoft Development Center Serbia	41,51%
XRCE-run2 Xerox Research Centre Europe	28,47%
XRCE-run1 Xerox Research Centre Europe	27,72%
XRCE-run3 Xerox Research Centre Europe	27,33%
Noopsis Noopsis inc.	8,32%
GREYC-run1 GREYC - University of Caen, France	0,08%
GREYC-run2 GREYC - University of Caen, France	0,08%
GREYC-run3 GREYC - University of Caen, France	0,08%

Table 2. Detailed results for GREYC

	Precision	Recall	F-Measure
Titles	19,83%	13,60%	13,63%
Levels	16,48%	12,08%	11,85%
Links	1,04%	0,14%	0,23%
Complete entries	0,40%	0,05%	0,08%
Entries disregarding depth	1,04%	0,14%	0,23%

The results were recomputed with correction on the unfortunate page number shift in the INEX grid (Table 3).

Table 3. GREYC results with page numbering correction

	precision	recall	F-measure
run-1	10,41	7,41	7,66
run-2	10,56	7,61	7,85
run-3	11,22	7,61	8,02

¹ <http://users.info.unicaen.fr/~doucet/StructureExtraction2009/>

The alternative evaluation grid suggested by [4], was applied. In table 4, for the GREYC result, the corrected run p= p-1 is computed under the name "GREYC-1"²².

Table 4. Alternative evaluation

	XRCE Link-based Measure				
	Links			Accuracy (for valid links)	
	Precision	Recall	F1	Title	Level
MDCS	65.9	70.3	66.4	86.7	75.2
XRCE-run3	69.7	65.7	64.6	74.4	68.8
XRCE-run2	69.2	64.8	63.8	74.4	69.1
XRCE-run1	67.1	63.0	62.0	74.6	68.9
Noopsis	46.4	38.0	39.9	71.9	68.5
GREYC-1	59.7	34.2	38.0	42.1	73.2
GREYC	6.7	0.7	1.2	13.9	31.4

The results still suffered from insufficient provision made for the evaluation rules. Notably, the title hierarchy is not represented, which impairs recall. Titles were not segmented on the right side, which impairs precision. However, level accuracy is quite encouraging.

Corrections after official competition

A simple corrective strategy was applied in order to better compare methods. A new feature boosted precision.

In a supplementary run (run 4) the title right end is detected, by calculating the line height disruption. This correction results in a better precision as shown in Table 5 (line GREYC-2) with the XRCE link-based measure. The recall rate is not improved because the subtitles are still not looked for.

Table 5. Corrected run (GREYC-2) with better title extraction compared with previous results

	XRCE Link-based Measure				
	Links			Accuracy (for valid links)	
	Precision	Recall	F1	Title	Level
GREYC-2	64.3	37.1	41.1	45.1	73.1
GREYC-1	59.7	34.2	38.0	42.1	73.2
GREYC	6.7	0.7	1.2	13.9	31.4

²² <http://users.info.unicaen.fr/~doucet/StructureExtraction2009/AlternativeResults.html>

Discussion

The experiment was preliminary. We were pleased to be able to handle the entire corpus. The results were very bad as expected. The low recall is due to the fact that the hierarchy of titles was not addressed as mentioned earlier. This will be addressed in the future.

The experiment was conducted in two phases. The first strategy was more classical; it started from the page up to the chapter and relied on title “literal” detection. The second was the document to chapter “positional” strategy explained above.

Reflections on the experiment

Although the results are bad, they showed some strong points of the Resurgence program, based on relative position and differential principles. We intend to further explore this way. The advantages are the following. The program deals with the entire document, not only the table of contents; it handles books without ToCs, and titles that are not represented in the ToC (e. g. preface). It is dependent on typographical position, which is very stable in the corpus; it is not dependent on lexicon, hence tolerant to OCR errors and language independent. Last, it is simple and fast.

Some examples below underline our point. They illustrate problems that were met in the first “literal” attempt but avoided by the “positional” solution.

Example 1. Varying forms for the string “chapter” due to OCR errors handled by

Resurgence

CHAPTEK

CHAPTEE

CH^PTEE

CHAP TEE

CHA 1 TKR

C II APT Kit

(MI A I TKII

C II A P T E II

C H A P T E H

C H A P T E R

C II A P T E U

Oil A PTKR

Since no expectations bear on language-dependent forms, the extracted strings are for example the following. A reader can detect *a posteriori* that this is being written in French (first series) or in English (second series).

- TABLE DES MATIÈRES
- DEUXIEME PARTIE
- CHAPITRE
- TROISIÈME PARTIE
- QUATRIÈME PARTIE

- PREFACE
- CHAPTER
- TABLE OF CONTENTS
- INTRODUCTION
- APPENDIX

Since no list of expected and memorized forms is used, but position instead, fairly common strings are extracted, such as CHAPTER or SECTION, but also uncommon ones, such as PSALM or SONNET. When chapters have no numbering and no prefix such as *chapter*, they are found as well, for instance a plain title “Christmas Day”.

Resurgence did not rely on numbering of chapters: this is an important source of OCR errors, like in the following series. Hence they were retrieved as they were by our robust extractor.

- II
- HI
- IV
- V
- VI

- SECTION VI
- SECTION VII
- SECTION YTTT
- SECTION IX

- SKOTIOX XMI
- SECTION XV
- SECTION XVI

- THE FIRST SERMON
- THE SECOND SERMON
- THE THIRD SERMON
- THE FOURTH SERMON

- CHAPTEE TWELFTH
- CHAPTER THIRTEENTH
- CHAPTER FOURTEENTH

Proposals

Concerning evaluation rules, generally speaking, it is unclear whether the groundtruth depends on the book or on the ToC. If the ToC is the reference, it is an error to extract prefaces, for instance.

Concerning details, the reason why the beginning and end of the titles are overrepresented is not clear and a more straightforward edit distance for extracted titles should be provided.

It should be clear whether or when the prefix (*Chapter, Section*, and so on) and the numbering should be part of the extracted result. The groundtruth is not clear either on the extracted title case: sometimes the case differs in the ToC and in the actual title in the book.

It would be very useful to provide results by title depth (level) as suggested by [4], because it seems that providing complete results for one or more level(s) would be more satisfying than missing some items at all levels. It is important to get coherent and comparable text spans for many tasks, such as indexing, or helping navigation.

There is also a bias introduced by a semi-automatically constructed groundtruth. It is understood that manual annotation is still to be conducted to improve the groundtruth quality. We had technical difficulties to meet that requirement this summer, which unfortunately were not solved by the organizing committee. It might be a better idea to open annotation to a larger audience and for longer periods of time.

The corpus provided for the INEX Book track was very valuable, it is the only available corpus offering full books and it was interesting for it provided various examples of layout.

References

1. Doucet, A. and Kazai, G. ICDAR 2009 Book Structure Extraction Competition. *10th International Conference on Document Analysis and Recognition ICDAR 2009*, Barcelona, Spain, IEEE. pp. 1408-1412. (2009).
2. Giguët, E., Lucas, N. & Chircu, C. *Le projet Resurgence: Recouvrement de la structure logique des documents électroniques*. JEP-TALN-RECITAL'08 Avignon (2008).
3. pdf2xml open source software, Déjean, H. last visited November 2009 <http://sourceforge.net/projects/pdf2xml/>
4. Déjean, H. and Meunier, J.-L. "XRCE Participation to the Book Structure Task" in *Advances in Focused Retrieval*, Berlin, Springer. LNCS 5631/2009: 124-131. doi: 10.1007/978-3-642-03761-0 (2009).

XRCE Participation to the Book Structure Task

Hervé Déjean, Jean-Luc Meunier

Xerox Research Centre Europe
6 Chemin de Maupertuis, F-38240 Meylan
Firstname.lastname@xrce.xerox.com

Abstract. We present here the XRCE participation to the Structure Extraction task of the INEX Book track 2009. After briefly explaining the four methods used for detecting the book structure in the book body, we explain how we composed them to address the book structure task. We then discuss the Inex evaluation method and propose another measure together with the corresponding software. We then report on each individual method. Finally, we report on our evaluation of the results of all participants.

1. Introduction

We present in this paper our participation to the Structure Extraction task of the INEX Book 2009. Our objective was to experiment with the use of multiple unsupervised methods to realize the task. This article will therefore briefly describe each of them before focusing on the evaluation of our results as well as those of the other participants. We use here the metric we proposed in 2008, whose software implementation is now available at:

<http://users.info.unicaen.fr/~doucet/StructureExtraction2009/AlternativeResults.html>

Most of the document conversion methods we discuss here are freely accessible at <http://rossinante.xrce.xerox.com:8090> with `inex/colorcube` as login/password.

This work is supported by the Large Scale Integrating Project SHAMAN, co-funded under the EU 7th Framework Programme (<http://shaman-ip.eu/shaman/>).

2. Pre-processing

The first step simply consists in reformatting the XML INEX format into our internal format, mostly renaming elements and adding some attributes (such as unique IDs). This was performed using XSLT technology.

A second step consists in detecting pages headers and footers, which often introduce noisy for our table of contents detector (see [1]).

A third step consists in recognizing the page numbering of the document (see [3]), in order to associate each physical page with zero or one logical page number, the latter being a piece of text. This is again an unsupervised method.

3. The Four Methods

The first method aims at parsing the ToC page so as to segment it into entries, each entry being formed by a label and a reference to a page number. Our second method is dedicated at parsing an index page. The third method is our classical Toc detection

method (see [2]). The fourth method uses some characteristic page layout in the document body.

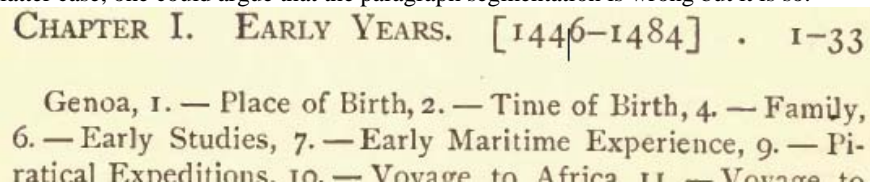
None of these methods aims at determining the entry level, so it is arbitrarily set to 1.

3.1. Parsing the ToC Pages

Parsing the ToC pages involves first finding them. For this purpose, we tried first with a simple heuristic that consists in looking for the keyword ‘contents’ in the few first line of each page, under the hypothesis of the presence of one ToC, possibly split over consecutive pages at the beginning or end of the book. We look for a series of pages containing this word and tolerate a certain number of misses.

Somehow to our own surprise, this method led to a F1, in term of ToC page retrieval task, in the range 90-95% over the 2008 INEX dataset. However, retrieving the page of the ToC of a document is not enough.

We then need to parse the contiguous pages deemed to be the ToC. The segmentation in paragraph is unfortunately not adequate, since a valid paragraph may both correspond to multiple entries, as shown below, or to part of one entry. In the latter case, one could argue that the paragraph segmentation is wrong but it is so.



CHAPTER I. EARLY YEARS. [1446-1484] . 1-33

Genoa, 1. — Place of Birth, 2. — Time of Birth, 4. — Family, 6. — Early Studies, 7. — Early Maritime Experience, 9. — Piratical Expeditions, 10. — Voyage to Africa, 11. — Voyage to

Figure 1: excerpt from the 2008 book #3 (0008D0D781E665AD).

We decided to use the reference to page numbers as ToC entry segmenter, as already mentioned in the literature [44,55]. Once the ToC is segmented, the referred to page is considered as the target of the entry and the entry text becomes the title. Here we take benefit of the recognition of the page numbering, performed in the pre-processing step, and which associates each physical page with zero or one logical page number.

3.2. Parsing the Index Pages

In a similar way, it is possible to look for some index pages, using the ‘index’ keyword, and to segment it based on the appearance of a page number at the end of the lines. The text in-between two page numbers is deemed to be the title of some part of the document, starting at the indicated page number.

3.3. “Classical” Detection of the ToCs

The method is detailed in [1], [2] and in this section we will only sketch its outline. The design of this method has been guided by the interest in developing a generic method that uses very intrinsic and general properties of the object known as a table of contents. In view of the large variation in shape and content a ToC may display, we believe that a descriptive approach would be limited to a series of specific collections. Therefore, we instead chose a functional approach that relies on the functional properties that a ToC intrinsically respects. These properties are:

1. Contiguity: a ToC consists of a series of contiguous references to some other parts of the document itself;
2. Textual similarity: the reference itself and the part referred to share some level of textual similarity;
3. Ordering: the references and the referred parts appear in the same order in the document;
4. Optional elements: a ToC entry may include (a few) elements whose role is not to refer to any other part of the document, e.g. decorative text;
5. No self-reference: all references refer outside the contiguous list of references forming the ToC.

Our hypothesis is that those five properties are sufficient for the entire characterization of a ToC, independently of the document class and language. In the Evaluation and Discussion section, we will discuss the cases where these hypotheses were not valid.

3.4. Trailing Page Whitespace

Here we are exploiting a widespread convention: main book parts or chapters are often preceded by a page break, i.e. an incomplete page. In consequence, there is some extra blank space between the end of the previous and the start of the next part. So detecting such blank space, also called trailing page whitespace, leads to discover some of the parts of the book. In addition, the title must also be extracted and we currently rely on a heuristic to locate it in the few first lines of the page.

We conjecture that this method has the potential for finding the top level structure of many books while inner structure may remain hidden to it.

4. The Three Runs

We combined in three different ad-hoc ways the methods described earlier and submitted three results.

4.1. First Run

To start with, we applied the keyword heuristic, explained at the beginning of the section 3.1, to all books. Those with a ‘contents’ keyword were then processed as said in 3.1. Those without ‘contents’ but with ‘index’ were processed as said in 3.2. In both case, if the method failed to find resp. a ToC or an Index, the classical method (3.3) was then applied.

The rationale for this process was the assumption that the presence of a ‘contents’ keyword indicated a ToC. Of course the found ToC might be not parsable based on the page number, for instance because many secondary entries have no associated page number.

768 books were deemed to have a ToC indicated by a keyword ‘contents’, among which 722 could be segmented based on the occurrence of page numbers. This left 46 documents for the ‘classical’ method.

21 books had the ‘index’ keyword but not the ‘contents’, among which 16 were processed by the 3.2 method. This left 5 more for the ‘classical’ method, which eventually processed 51 books plus the 211 books with neither ‘contents’ nor ‘index’ in them (262 books).

4.2. Second Run

Our second run was an extension of the first one: if the ‘classical’ method fails to detect a ToC, then we apply the ‘trailing space’ method (3.4) to structure the book.

We found that among the 262 books processed by the ‘classical’ method, 50 of them got no result at all, and went then trough the ‘trailing space’ method.

4.3. Third Run

Our third run was a variation of the previous ones. If the ‘contents’ keyword is found, we put the method 3.1 and 3.3 in competition by running each of them separately and taking the best output. The best one is supposed to be the one with the longest textual contents, assuming it “explains” better the ToC pages.

For the other books, we applied the ‘index’ method if the appropriate keyword was found, and in last chance we ran the ‘trailing space’ method.

5. Evaluation of XRCE results

5.1. Inex Metric

According to the Inex metric, the 3 runs show little difference, with a “complete entries” F1 measure at about 28%.

In more detail, let us look on the first run for instance. The title precision is at 48.23% and the link precision is at 45.22%. This decrease is indicative of the proportion of entries with a valid title that also have a valid link. It means that 94% of the entries with a valid title, have a valid link as well, although the INEX links metric indicate a precision of 45.22%. In our view, this again points at a room for improvement of the INEX links measure. It is desirable in our view to separate the title quality measure from the link quality measure.

5.2. Proposed Link Metric

As initiated in our 2008 paper, we advocate for a complementary metric that would qualify the quality of the links in first intention, rather than conditionally to the title validity.

Such a metric is now available as a Python software at:

<http://www.xrce.xerox.com/Research-Development/Publications/2009-078> .

It computes:

- the precision, recall and F1 measure of the links, by considering a submission as a list of page breaks, ordered by the page number of the breaks. So it counts valid, wrong and missed links. The software actually can display this low-level information per book in a submission.
- for each valid link, it computes the similarity of the proposed title with the ground truth one, using the INEX weighted Levenshtein distance:

$$simil(s1, s2) = 1 - \frac{weightedLevenshtein(s1, s2)}{\max(weight(s1), weight(s2))}$$

Averaging over all entries with a valid link gives a title accuracy measure for each book.

- In the same manner, a level accuracy per book is computed by considering all entries with a valid link.
- A INEX-like link measure, by matching the title of valid links against the ground truth. So it is similar to the INEX link apart that the link validity is checked before the title validity. If both are valid, the entry is valid. Our experiments validate the closeness of the two measures. A slight difference may occur if several links point to the same page. Since we do not look for the optimal alignment of the corresponding title, the INEX-like measure can be lower.

According to this metric, our 3 runs are also similar to each other, the third one being the better.

The table below summarizes all these values.

%	Inex - Links			Proposed link measure				
	Prec.	Rec.	F1	Prec.	Rec.	F1	Title acc.	Level acc.
xrx1	45.22	41.40	42.13	67.1	63.0	62.0	74.6	68.9
xrx2	46.39	42.47	43.15	69.2	64.8	63.8	74.4	69.1

xrx3	45.16	41.70	42.28	69.7	65.7	64.6	74.4	68.8
------	-------	-------	-------	------	------	------	------	------

Table 1 : XRCE results according to the Inex links measure and to the proposed measure.

We observe that the proposed link measure indicates significantly better results, in the range 60-70%, which matches better our feeling when looking at the results.

Those valid links have a textual similarity around 75% with the ground truth. This is why a part of them are invalidated by the official INEX links measure. Again, we question this behavior; indeed, should the title “I Introduction” be invalidated because the groundtruth specifies “Chapter I Introduction”, for instance?

Regarding the levels, the table below gives the proportion of each level in the groundtruth.

Level	1	2	3	4	5	6	7
Proportion	39%	41%	16%	4%	1%	0.1%	0.02%

Table 1 : Proportion of each level in the groundtruth.

Our methods set by construction all levels to 1, since we did not try to determine the level. We get 7166 valid levels out of 17309 valid links, which means that 41% of the valid links belong to level 1. So our combined method does pretty equally well on all levels. On the other hands, the level accuracy per book is higher on smaller books, this is why the micro-accuracy of 41% is much lower than the macro-accuracy of 68.9% reported in the table 1 above (a book with few entries as the same importance than a larger book with many entries in the macro-accuracy measure).

5.3. Evaluation of the Four XRCE Methods

We will use the proposed measure in the rest of this document.

Using the ground truth, we then evaluated separately all the four methods. We discarded the books 0F7A3F8A17DCDA16 (#40) and E0A9061DCCBA395A (#517) because of massive errors in their respective ground truth. We also decided to only use the ground truth data built collectively in 2009, with the hope to benefit from a higher quality groundtruth, thanks to the use of the nice annotation tool provided to us.

This led to a ground truth of 427 books (only 98 books from 2008 were in the 2009 ground truth).

%	Inex-LIKE - Links			(Proposed) link measure				
	Prec.	Rec.	F1	Prec.	Rec.	F1	<i>Title acc.</i>	<i>Level acc.</i>
ToC page parsing	37.2	35.2	35.5	54.8	53.2	52.1	76.3	72.3
Index page parsing	0.0	0.0	0.0	0.5	0.3	0.2	28.5	53.7
Classical ToC	24.6	19.5	20.8	60.3	49.2	50.8	70.4	71.2
Trailing	22.9	24.4	21.4	56.1	62.6	52.4	59.1	78.8

whitespace								
------------	--	--	--	--	--	--	--	--

Table 2 : The four XRCE methods measured individually.

The first observation is that the use of the index pages, if any, largely fails. Of course, on some documents, like the 491st of the ground truth, the book structuration would benefit from a merge of the ToC and of the index (p15-18).

The second observation is a threshold effect on the title accuracy. It seems that an accuracy below ~75% importantly penalizes the Inex-like result, at almost equal F1 (proposed) measure. Or said differently, given a certain F1 measure, a title accuracy below 75% seems to penalize heavily the INEX F1 measure.

Interestingly, the combination of the 4 methods for the runs leads to a 7 points gain on the F1, reaching 59.3 on these 427 books.

There is room for improvement here since by taking for each book, the max of the F1 of the 4 methods, the averaged F1 reaches 75% with 49% of the books over 0.9 in F1, and 63% over 0.8. On the other hand, 16% of the document would remain below 0.5 in F1. This is the optimal reachable F1 one can obtain by choosing at best which method to trust. Such choice is however difficult to make automatically. Another approach could consist in mixing the results of the four methods for each book, rather than choosing one of them, in some way to be researched.

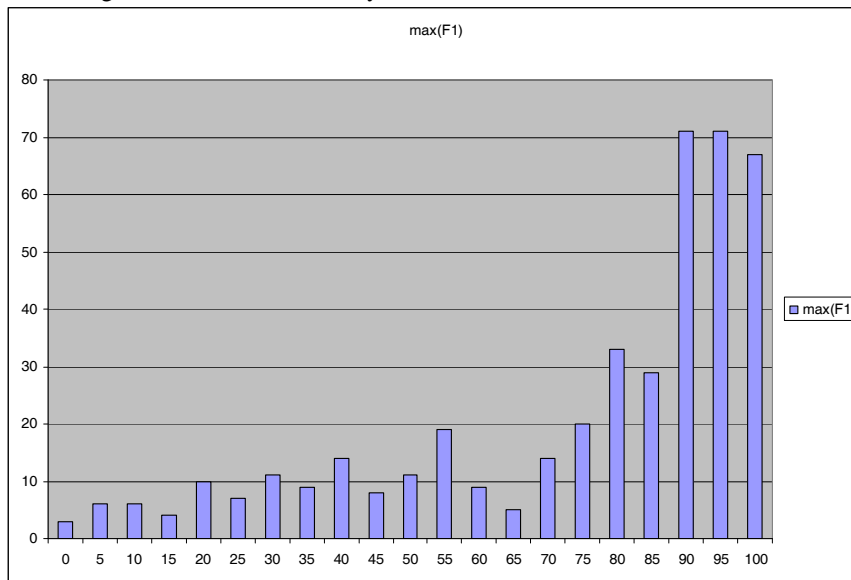


Figure 2: The histogram of the distribution of the F1 when manually choosing optimally which method to apply to each document. This is theoretical data. (*The vertical scale indicates a number of documents.*)

6. Evaluation of All Participant Results

We again use the proposed new measure to evaluate the results of all the participants, in term of links, titles and levels.

This evaluation is performed using the 2009 official ground truth, which include 527 books. For the GREYC result, we observed an unfortunate shift of +1 on all page numbers so we corrected their result before evaluating it under the name GREYC-1.

%	Inex-LIKE - Links			(Proposed) link measure				
	Prec.	Rec.	F1	Prec.	Rec.	F1	<i>Title acc.</i>	<i>Level acc.</i>
MDCS	52.4	54.6	52.8	65.9	70.3	66.4	86.7	75.2
XR3	44.1	40.8	41.4	69.7	65.7	64.6	74.4	68.8
XR2	45.3	41.6	42.3	69.2	64.8	63.8	74.4	69.1
XR1	44.2	40.6	41.3	67.1	63.0	62.0	74.6	68.9
NOOPSIS	15.1	12.0	12.9	46.4	38.0	39.9	71.9	68.5
GREYC-1	10.2	7.2	7.7	59.7	34.2	38.0	42.1	73.2
GREYC	0.0	0.0	0.0	6.7	0.7	1.2	13.9	31.4

Table 3 : The results of all participants measured using the proposed metric.

In term of F1 measure of the links, we observe two groups of result, one in the 65% the other in the 40%. The title accuracy ranges from 87% to 42%, with again an important impact on the initial INEX links measure.

As for the level accuracy, MDCS and GREYC do slightly better than the two other participants who simply set all level to 1, gaining about 7% in accuracy, i.e. the proportion a valid level among the valid links. We tend to think that it is indicative of a very difficult task.

We now show the histogram of the link F1 per participant as well as the same histogram computed with the maximum F1 reached by one participant for each book. Such result is achievable is one knows how to choose at best for each book the result to choose among the 4 participants.

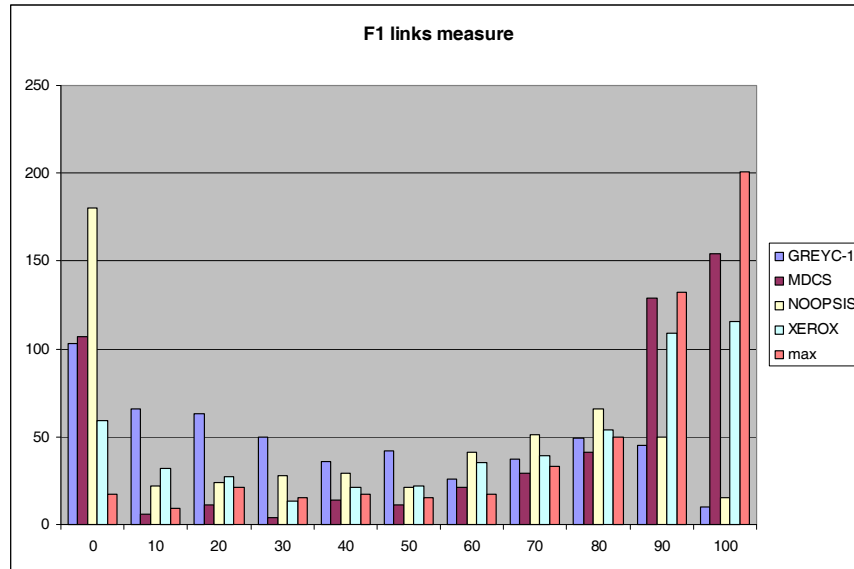


Figure 3: The histogram of the distribution of the F1 when choosing optimally which method to apply to each document. This is theoretical data.

Choosing optimally the best result per document would lead to a 79% average F1, with 63% of the books over 0.9 in F1, and 73% over 0.8. On the other hand, 20% of the document would remain below 0.5 in F1.

Unfortunately, this measure does not distinguish a non-response from a wrong response, since both get a 0% in precision and recall. Indeed, non-responses are tolerable in many applications while wrong responses are problematic. So we computed the number of non-responses per participant:

GREYC=80 MDCS=101 NOOPSIS=171 XRCE=45

It turns out that the combination “by maximization” leaves only 14 books without answers. It would be interesting to look in more details at those books.

7. Conclusion

We have experimented with combining multiple methods to perform the task, except the level determination for which we have not method in place. Our method for combining was ad-hoc and it provided a limited gain, ~7%, over each individual method. Among those methods, we have found disappointing the method for parsing the index pages while the method looking at trailing page whitespace is promising and was under-exploited.

On the other hand, if one knows how to choose the best method for each book, a gain over 20% in F1 can be achieved, at a computationally modest cost. This is an attractive path to pursue.

We have also proposed another metric to evaluate the results and made it available in software form. We are grateful to Antoine Doucet for his support in providing us with the submission of all participants and for publishing those new results together with the evaluation software.

Finally, we have found the task of making the ground truth quite labor intensive, despite the very convenient tool provided to us. This plays in favor of finding efficient automatic methods.

References

1. Déjean H., Meunier J.-L., Structuring Documents according to their Table of Contents. In: Proceedings of the 2005 ACM symposium on Document engineering, pp. 2—9. ACM, New York, NY, USA (2005)
2. Déjean H., Meunier J.-L., On Tables of Contents and how to recognize them. In: *International Journal of Document Analysis and Recognition (IJ DAR)*, 2008.
3. Déjean H., Meunier J.L., Versatile page numbering analysis, Document Recognition and Retrieval XV, part of the IS&T/SPIE International Symposium on Electronic Imaging, San Jose, California, USA, 26-31 January 2008. DRR08
4. Belaïd, A., Recognition of table of contents for electronic library consulting. In: *International Journal of Document Analysis and Recognition (IJ DAR)*, 2001.
5. Lin, X. Detection and analysis of table of contents based on content association. In: *International Journal of Document Analysis and Recognition (IJ DAR)*, 2005.

Ranking and Fusion Approaches for XML Book 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 Book track and all of our submissions were for that track. We tried a variety of different approaches for our Book Track runs, the TREC2 logistic regression probabilistic model used in previous INEX Book Track submissions as well as various fusion approaches including use of the Okapi BM-25 algorithm. No results are yet available for the track, so this paper focusses on the approaches used in the various runs submitted.

1 Introduction

In this paper we will first discuss the algorithms and fusion operators used in our official INEX 2008 Book Track runs. Then we will look at how these algorithms and operators were used in combination with indexes for various parts of the book contents in our submissions for this track, and finally we will discuss possible directions for future research.

2 The Retrieval Algorithms and Fusion Operators

This section largely duplicates earlier INEX papers in describing the probabilistic retrieval algorithms used for both the Adhoc and Book track in INEX this year. Although These are the same algorithms that we have used in previous years for INEX and in other evaluations (such as CLEF), including a blind relevance feedback method used in combination with the TREC2 algorithm, we are repeating the formal description here instead of referring to those earlier papers alone. In addition we will again 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 [9, 8, 7] which also supports a number of other algorithms for distributed search and operators for merging result lists from ranked or Boolean sub-queries.

2.1 TREC2 Logistic Regression Algorithm

Once again the principle algorithm used for our INEX runs is based on the *Logistic Regression* (LR) algorithm originally developed at Berkeley by Cooper, et al. [5]. The version that we used for Adhoc tasks was the Cheshire II implementation of the “TREC2” [4, 3] that provided good Thorough retrieval performance in the INEX 2005 evaluation [9]. As originally formulated, the LR model of probabilistic IR attempts to estimate the probability of relevance for each document based on a set of statistics about a document collection and a set of queries in combination with a set of weighting coefficients for those statistics. The statistics to be used and the values of the coefficients are obtained from regression analysis of a sample of a collection (or similar test collection) for some set of queries where relevance and non-relevance has been determined. More formally, given a particular query and a particular document in a collection $P(R | Q, D)$ is calculated and the documents or components are presented to the user ranked in order of decreasing values of that probability. To avoid invalid probability values, the usual calculation of $P(R | Q, D)$ uses the “log odds” of relevance given a set of S statistics derived from the query and database, such that:

$$\begin{aligned} \log O(R|C, Q) &= \log \frac{p(R|C, Q)}{1 - p(R|C, Q)} = \log \frac{p(R|C, Q)}{p(\bar{R}|C, Q)} \\ &= c_0 + c_1 * \frac{1}{\sqrt{|Q_c|} + 1} \sum_{i=1}^{|Q_c|} \frac{qt f_i}{ql + 35} \\ &+ c_2 * \frac{1}{\sqrt{|Q_c|} + 1} \sum_{i=1}^{|Q_c|} \log \frac{tf_i}{cl + 80} \\ &- c_3 * \frac{1}{\sqrt{|Q_c|} + 1} \sum_{i=1}^{|Q_c|} \log \frac{ct f_i}{N_i} \\ &+ c_4 * |Q_c| \end{aligned}$$

where C denotes a document component and Q a query, R is a relevance variable, and

$p(R|C, Q)$ is the probability that document component C is relevant to query Q ,

$p(\bar{R}|C, Q)$ the probability that document component C is not relevant to query Q , (which is $1.0 - p(R|C, Q)$)

$|Q_c|$ is the number of matching terms between a document component and a query,

$qt f_i$ is the within-query frequency of the i th matching term,

tf_i is the within-document frequency of the i th matching term,

$ct f_i$ is the occurrence frequency in a collection of the i th matching term,

ql is query length (i.e., number of terms in a query like $|Q|$ for non-feedback situations),

cl is component length (i.e., number of terms in a component), and N_t is collection length (i.e., number of terms in a test collection). c_k are the k coefficients obtained through the regression analysis.

Assuming that stopwords are removed during index creation, then ql , cl , and N_t are the query length, document length, and collection length, respectively. If the query terms are re-weighted (in feedback, for example), then qtf_i is no longer the original term frequency, but the new weight, and ql is the sum of the new weight values for the query terms. Note that, unlike the document and collection lengths, query length is the relative frequency without first taking the log over the matching terms.

The coefficients were determined by fitting the logistic regression model specified in $\log O(R|C, Q)$ to TREC training data using a statistical software package. The coefficients, c_k , used for our official runs are the same as those described by Chen[1]. These were: $c_0 = -3.51$, $c_1 = 37.4$, $c_2 = 0.330$, $c_3 = 0.1937$ and $c_4 = 0.0929$. Further details on the TREC2 version of the Logistic Regression algorithm may be found in Cooper et al. [4].

2.2 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)[14] and TREC-8 (Voorhees and Harman 1999)[15].)

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[2] 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 [6] provides a survey of relevance feedback techniques that have been used.

Obviously there are important choices to be made regarding the number of top-ranked documents to consider, and the number of terms to extract from

those documents. For this year, having no truly comparable prior data to guide us, we chose to use the top 10 terms from 10 top-ranked documents. The terms were chosen by extracting the document vectors for each of the 10 and computing the Robertson and Sparck Jones term relevance weight for each document. This weight is based on a contingency table where the counts of 4 different conditions for combinations of (assumed) relevance and whether or not the term is, or is not in a document. Table 1 shows this contingency table.

Table 1. Contingency table for term relevance weighting

	Relevant	Not Relevant	
In doc	R_t	$N_t - R_t$	N_t
Not in doc	$R - R_t$	$N - N_t - R + R_t$	$N - N_t$
	R	$N - R$	N

The relevance weight is calculated using the assumption that the first 10 documents are relevant and all others are not. For each term in these documents the following weight is calculated:

$$w_t = \log \frac{\frac{R_t}{R - R_t}}{\frac{N_t - R_t}{N - N_t - R + R_t}} \quad (1)$$

The 10 terms (including those that appeared in the original query) with the highest w_t are selected and added to the original query terms. For the terms not in the original query, the new “term frequency” ($qt f_i$ in main LR equation above) is set to 0.5. Terms that were in the original query, but are not in the top 10 terms are left with their original $qt f_i$. For terms in the top 10 and in the original query the new $qt f_i$ is set to 1.5 times the original $qt f_i$ for the query. The new query is then processed using the same TREC2 LR algorithm as shown above and the ranked results returned as the response for that topic.

2.3 Okapi BM-25 Algorithm

The version of the Okapi BM-25 algorithm used in these experiments is based on the description of the algorithm in Robertson [12], and in TREC notebook proceedings [13]. As with the LR algorithm, we have adapted the Okapi BM-25 algorithm to deal with document components :

$$\sum_{j=1}^{|Q_c|} w^{(1)} \frac{(k_1 + 1) t f_j}{K + t f_j} \frac{(k_3 + 1) q t f_j}{k_3 + q t f_j} \quad (2)$$

Where (in addition to the variables already defined):

K is $k_1((1 - b) + b \cdot dl/avcl)$

k_1 , b and k_3 are parameters (1.5, 0.45 and 500, respectively, were used),

$avcl$ is the average component length measured in bytes

$w^{(1)}$ is the Robertson-Sparck Jones weight:

$$w^{(1)} = \log \frac{\left(\frac{r+0.5}{R-r+0.5}\right)}{\left(\frac{n_{t_j}-r+0.5}{N-n_{t_j}-R-r+0.5}\right)}$$

r is the number of relevant components of a given type that contain a given term,

R is the total number of relevant components of a given type for the query.

Our current implementation uses only the *a priori* version (i.e., without relevance information) of the Robertson-Sparck Jones weights, and therefore the $w^{(1)}$ value is effectively just an IDF weighting. The results of searches using our implementation of Okapi BM-25 and the LR algorithm seemed sufficiently different to offer the kind of conditions where data fusion has been shown to be most effective [10], and our overlap analysis of results for each algorithm (described in the evaluation and discussion section) has confirmed this difference and the fit to the conditions for effective fusion of results.

2.4 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 components of a document.

For the Adhoc Focused and Best In Context runs we used a merge/reweighting operator based on the ‘‘Pivot’’ method described by Mass and Mandelbrod[11] to combine the results for each type of document component considered. In our case the new probability of relevance for a component is a weighted combination of the initial estimate probability of relevance for the component and the probability of relevance for the entire article for the same query terms. Formally this is:

$$P(R | Q, C_{new}) = (X * P(R | Q, C_{comp})) + ((1 - X) * P(R | Q, C_{art})) \quad (3)$$

Where X is a pivot value between 0 and 1, and $P(R | Q, C_{new})$, $P(R | Q, C_{comp})$ and $P(R | Q, C_{art})$ are the new weight, the original component weight, and article weight for a given query. Although we found that a pivot value of 0.54 was most effective for INEX04 data and measures, we adopted the ‘‘neutral’’ pivot value of 0.4 for all of our 2008 adhoc runs, given the uncertainties of how this approach would fare with the new database.

3 Database and Indexing Issues

For the Book Track data this year, we attempted to use multiple elements or components that were identified in the Books markup as Tables of Contents and Indexes as well as the full text of the book, since the goal of the main Books Adhoc task was to retrieval entire books and not elements, the entire book was retrieved regardless of the matching elements. We created the same indexes for the Books and for their associated MARC data that we created last year (the MARC fields are shown in Table 5, for the books themselves we used a single index of the entire document content. We did not use the Entry Vocabulary Indexes used in previous years Book track runs, since their performance was, in general, less effective than using the full contents.

Table 2. Book-Level Indexes for the INEX Book Track 2009

Name	Description	Contents	Vector?
topic	Full content	//document	Yes
toc	Tables of Contents	//section@label="SEC.TOC"	No
index	Back of Book Indexes	//section@label="SEC.INDEX"	No

Table 2 lists the Book-level (/article) indexes created for the INEX Books database and the document elements from which the contents of those indexes were extracted.

Table 3. Components for INEX Book Track 2009

Name	Description	Contents
COMPONENT_PAGE	Pages	//page
COMPONENT_SECTION	Sections	//section

Cheshire system permits parts of the document subtree to be treated as separate documents with their own separate indexes. Tables 3 & 4 describe the XML components created for the INEX Book track and the component-level indexes that were created for them.

Table 3 shows the components and the paths used to define them. The first, referred to as COMPONENT_PAGE, is a component that consists of each identified page of the book, while COMPONENT_SECTION identifies each section of the books, permitting each individual section or page of a book to be retrieved separately. Because most of the areas defined in the markup as “section”s are actually paragraphs, we treat these as if they were paragraphs for the most part.

Table 4 describes the XML component indexes created for the components described in Table 3. These indexes make the individual sections (such as 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

Table 4. Component Indexes for INEX Book Track 2009

Component or Index Name	Description	Contents	Vector?
COMPONENT_SECTION			
para_words	Section Words	* (all)	Yes
COMPONENT_PAGES			
page_words	Page Words	* (all)	Yes

with proximity searching. Individual figures (COMPONENT_FIG) are indexed by their captions.

Table 5. MARC Indexes for INEX Book Track 2009

Name	Description	Contents	Vector?
names	All Personal and Corporate names	//FLD[1670]00, //FLD[1678]10, //FLD[1670]11	No
pauthor	Personal Author Names	//FLD[170]00	No
title	Book Titles	//FLD130, //FLD245, //FLD240, //FLD730, //FLD740, //FLD440, //FLD490, //FLD830	No
subject	All Subject Headings	//FLD6..	No
topic	Topical Elements	//FLD6.., //FLD245, //FLD240, //FLD4.., //FLD8.., //FLD130, //FLD730, //FLD740, //FLD500, //FLD501, //FLD502 //FLD505, //FLD520, //FLD590	Yes
lcclass	Library of Congress Classification	//FLD050, //FLD950	No
doctype	Material Type Code	//USMARC@MATERIAL	No
localnum	ID Number	//FLD001	No
ISBN	ISBN	//FLD020	No
publisher	Publisher	//FLD260/b	No
place	Place of Publication	//FLD260/a	No
date	Date of Publication	//FLD008	No
lang	Language of Publication	//FLD008	No

The indexes used in the MARC data are shown in Table 5. Note that the tags represented in the “Contents” column of the table are from Cheshire’s MARC to XML conversion, and are represented as regular expressions (i.e., square brackets indicate a choice of a single character). We did not use the MARC data this year in our submitted runs.

3.1 Indexing the Books XML Database

Because the structure of the Books database was derived from the OCR of the original paper books, it is primarily focused on the page organization and

layout and not on the more common structuring elements such as “chapters” or “sections”. Because this emphasis on page layout goes all the way down to the individual word and its position on the page, there is a very large amount of markup for page with content. For this year’s original version of the Books database, there are actually NO text nodes in the entire XML tree, the words actually present on a page are represented as attributes of an empty word tag in the XML. The entire document in XML form is typically multiple megabytes in size. A separate version of the Books database was made available that converted these empty tags back into text nodes for each line in the scanned text. This provided a significant reduction in the size of database, and made indexing much simpler. The primary index created for the full books was the “topic” index containing the entire book content.

We also created page-level “documents” as we did last year. As noted above the Cheshire system permits parts of the document subtree to be treated as separate documents with their own separate indexes. Thus, paragraph-level components were extracted from the page-sized documents. Because unique object (page) level identifiers are included in each object, and these identifiers are simple extensions of the document (book) level identifier, we were able to use the page-level identifier to determine where in a given book-level document a particular page or paragraph occurs, and generate an appropriate XPath for it.

Indexes were created to allow searching of full page contents, and component indexes for the full content of each of individual paragraphs on a page. Because of the physical layout based structure used by the Books collection, paragraphs split across pages are marked up (and therefore indexed) as two paragraphs. Indexes were also created to permit searching by object id, allowing search for specific individual pages, or ranges of pages.

The system problems encountered last year have been (temporarily) corrected for this years submissions. Those problems were caused by the numbers of unique terms exceeding the capacity of the integers used to store them in the indexes. For this year, at least, moving to unsigned integers has provided a temporary fix for the problem but we will need to rethink how statistical summary information is handled in the future – perhaps moving to long integers, or even floating point numbers and evaluating the tradeoffs between precision in the statistics and index size (since moving to Longs could double index size).

4 INEX 2008 Book Track Runs

We submitted nine runs for the Book Search task of the Books track,

As Table 6 shows, a number of variations of algorithms and search elements were tried this year. In Table 6 the first column is the run name (all official submissions had names beginning with “BOOKS09” which has been removed from the name), the second column is a short description of the run, since each topic included not only a title, but one or more “aspects” each of which had an aspect title, column four also indicates whether title only or title and aspects combined were used in the submitted queries. The third column shows which

Table 6. Berkeley Submissions for the INEX Book Track 2009

Name	Description	Algorithm	Aspects?
T2FB_TOPIC_TITLE	Uses topic index with title and blind feedback	TREC2 +BF	No
T2FB_TOPIC_TA	Uses topic index with title and aspects with blind feedback	TREC2 +BF	Yes
T2_INDEX_TA	Uses Back of Book index with title and aspects	TREC2	Yes
T2_TOC_TA	Uses Table of Contents index with title and aspects	TREC2	Yes
OK_TOPIC_TA	Uses Topic index with title and aspects	Okapi BM-25	Yes
OK_TOC_TA	Uses Tables of Contents index with title and aspects	Okapi BM-25	Yes
OK_INDEX_TA	Uses Back of Book indexes with title and aspects	Okapi BM-25	Yes
FUS_TA	Fusion of Topic, Table of Contents, and Back of Book Indexes - title and aspects	TREC2 +BF	Yes
FUS_TITLE	Fusion of Topic, Table of Contents, and Back of Book Indexes - title only	TREC +BF	No

algorithms where used for the run, TREC2 is the TREC2 Logistic regression algorithm described above, “BF” means that blind relevance feedback was used in the run, and OKAPI BM-25 means that the OKAPI algorithm described above was used.

5 Conclusions and Future Directions

The results of the Books track are not yet available, but a few observations can be made about the runs. We suspect that the fusion of full contents with specific table of contents matching and back of the book indexes will probably perform best. This is because the table of contents and back of the book indexes alone are likely to miss relevant items due to the lack of such areas in many of the books. We hope the topic index with blind feedback will provide a good baseline that can be enhanced by matches in tables of contents and back of book indexes. But the actual effectiveness of fusion methods is often slightly less than a single effective method alone. Only the provision of evaluation results will see if that is the case for this database and collection of methods.

References

1. A. Chen. Multilingual information retrieval using english and chinese queries. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum, CLEF-2001, Darmstadt, Germany, September 2001*, pages 44–58. Springer Computer Science Series LNCS 2406, 2002.

2. A. Chen. *Cross-Language Retrieval Experiments at CLEF 2002*, pages 28–48. Springer (LNCS #2785), 2003.
3. A. Chen and F. C. Gey. Multilingual information retrieval using machine translation, relevance feedback and decomposing. *Information Retrieval*, 7:149–182, 2004.
4. W. S. Cooper, A. Chen, and F. C. Gey. Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression. In *Text REtrieval Conference (TREC-2)*, pages 57–66, 1994.
5. W. S. Cooper, F. C. Gey, and D. P. Dabney. Probabilistic retrieval based on staged logistic regression. In *15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, June 21-24*, pages 198–210, New York, 1992. ACM.
6. D. Harman. Relevance feedback and other query modification techniques. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 241–263. Prentice Hall, 1992.
7. R. R. Larson. A logistic regression approach to distributed IR. In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pages 399–400. ACM, 2002.
8. R. R. Larson. A fusion approach to XML structured document retrieval. *Information Retrieval*, 8:601–629, 2005.
9. R. R. Larson. Probabilistic retrieval, component fusion and blind feedback for XML retrieval. In *INEX 2005*, pages 225–239. Springer (Lecture Notes in Computer Science, LNCS 3977), 2006.
10. J. H. Lee. Analyses of multiple evidence combination. In *SIGIR '97: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 27-31, 1997, Philadelphia*, pages 267–276. ACM, 1997.
11. Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement for xml retrieval. In *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX2004*, pages 73–84. Springer (LNCS #3493), 2005.
12. S. E. Robertson and S. Walker. On relevance weights with little relevance information. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 16–24. ACM Press, 1997.
13. S. E. Robertson, S. Walker, and M. M. Hancock-Beauliee. OKAPI at TREC-7: ad hoc, filtering, vlc and interactive track. In *Text Retrieval Conference (TREC-7), Nov. 9-1 1998 (Notebook)*, pages 152–164, 1998.
14. E. Voorhees and D. Harman, editors. *The Seventh Text Retrieval Conference (TREC-7)*. NIST, 1998.
15. E. Voorhees and D. Harman, editors. *The Eighth Text Retrieval Conference (TREC-8)*. NIST, 1999.

Twente University at INEX 2009: Ad-hoc Track

Rongmei Li

Department of Computer Science,
University of Twente, P.O.Box 217 7500AE
Enschede, The Netherlands
`lir@cs.utwente.nl`

Abstract. In this paper we describe the University of Twente's participation in the INEX 2009 ad-hoc track. We participated in all four retrieval tasks (thorough, focused, relevant-in-context, best-in-context) and report initial findings based on a single set of measure for all tasks. In this first participation, we test two ideas: (1) evaluate the performance of standard IR engines used in full document retrieval and XML element retrieval; (2) investigate if document structure can lead to more accurate and focused retrieval result. We find: 1) the full document retrieval outperforms the XML element retrieval using simple mixture language model; 2) the element relevance score itself can be used to remove overlapping element results effectively.

1 Introduction

INEX offers a framework for cross comparison among content-oriented XML retrieval approaches given same test collections and evaluation measures. The INEX ad-hoc track is to evaluate system performance in retrieving relevant document components (e.g. XML elements or passages) for a given topic of request. The relevant results should discuss the topic exhaustively and have as little non-relevant information as possible (specific for the topic). The ad-hoc track includes four retrieval tasks: the Thorough task, the Focused task, the Relevant in Context task, and the Best in Context task.

The 2009 collection is the English Wikipedia with XML format. The ad-hoc topics are created by participants to represent real life information need. Each topic consists of five fields. The `<title>` field (CO query) is same as the standard keyword query. The `<castitle>` field (CAS query) adds structural constraints to the CO query by explicitly specifying where to look and what to return. The `<phrasetitle>` field (Phrase query) presents explicitly marked up query phrase. The `<description>` and `<narrative>` fields provide more information about topical context. Especially the `<narrative>` field is used for relevance assessment.

The paper documents our first participation in the INEX 2009 ad-hoc track. Our aims are to: 1) evaluate the performance of standard IR engines (Indri search engine) used in full document retrieval and XML element retrieval; 2) investigate if document structure can lead to more accurate and focused retrieval result. We

adopt the language modeling approach [1] and tailor the estimate of query term generation from a document to an XML element according to the user request. The retrieval results are evaluated as: 1) XML element or passage retrieval; 2) full document retrieval.

The rest of the paper describes our experiments in the ad-hoc track. The pre-processing and indexing are given in section 2. Section 3 explains how to convert user query to Indri structured query. The retrieval model and strategies are summarized in section 4. We present our results in section 5 and conclude this paper with discussion in section 6.

2 Pre-processing and Indexing

The original English XML Wikipedia is not stopped or stemmed before indexing. The 2009 collection has 2,666,190 documents taken on 8 October 2008. It is annotated with the 2008-w40-2 version of YAGO ([2]).

We index mainly the queried XML fields as follows: `category`, `actor`, `actress`, `adversity`, `aircraft`, `alchemist`, `article`, `artifact`, `bdy`, `bicycle`, `caption`, `catastrophe`, `categories`, `chemist`, `classical_music`, `conflict`, `director`, `dog`, `driver`, `group`, `facility`, `figure`, `film_festival`, `food`, `home`, `image`, `information`, `language`, `link`, `misfortune`, `mission`, `missions`, `movie`, `museum`, `music_genre`, `occupation`, `opera`, `orchestra`, `p`, `performer`, `person`, `personality`, `physicist`, `politics`, `political_party`, `protest`, `revolution`, `scientist`, `sec`, `section`, `series`, `singer`, `site`, `song`, `st`, `theory`, `title`, `vehicles`, `village`.

3 Query Formulation

We use CO queries for full article retrieval. For CAS queries, we adopt two different strategies to formulate our Indri structured query ([3]) for retrieving full article or XML elements respectively. The belief operator `#combine` is used for all cases. Neither CO queries nor CAS queries are stemmed or stopped.

- CAS queries for full article retrieval: we extract all `<castitle>` terms within “about” and XML tags that have semantic meaning as our query terms. Boolean operators (e.g. “-” or “+”) in `<castitle>` are ignored. For instance, the INEX query (id=2009009) has `<castitle>` as follows:

```
<castitle>//(p|sec)[about(.//(political_party|politics),election
+victory_australian_labor_party_state_council_federal)]</castitle>
```

After extraction, our query terms (2009009) are `election`, `victory`, `australian`, `labor`, `party`, `state`, `council`, `federal`.

- CAS queries for XML element retrieval: we extract all `<castitle>` terms within “about” and use Indri belief operators (`#not`) to exclude (“-”) certain terms. At the same time, we add XML element constraints (e.g. `<song>`, `<p>`)

and phrase constraints (e.g. “wonder girls” in 2009021) to our new Indri queries.

4 Retrieval Model and Strategies

We use the run of full article retrieval as our baseline for both CO and CAS queries. The retrieval model for the baseline runs is the mixture language model. It is defined as follows:

$$score(D) = \sum_{i=1}^l [P_{ml}(t_i|\theta_Q) \cdot \log(\lambda P_{ml}(t_i|\theta_D) + (1 - \lambda)P_{ml}(t_i|\theta_C))] \quad (1)$$

where l is the length of the query, $P_{ml}(t_i|\theta_Q)$, $P_{ml}(t_i|\theta_D)$, and $P_{ml}(t_i|\theta_C)$ are the Maximum Likelihood (ML) estimate of query model, document model, and the collection model.

For XML element retrieval, we compute the relevance score ($score(E)$) of queried XML field (E) in regard to the given CAS query. The smoothed document model (expressed in the log function) will compute the ML estimate of XML element model $P_{ml}(t_i|\theta_E)$ and document model $P_{ml}(t_i|\theta'_D)$.

We set up our language model and model parameters based on the experimental results of similar tasks for INEX 2008. Here λ is considered to be 0.9.

4.1 Baselines

Baseline runs retrieve full articles for CO and CAS queries. Only #combine operator is used. We submitted the results of CAS query for the Thorough and the Focused tasks and the results of CO query for the Relevant in Context and the Best in Context tasks. Due to overlapping problem, only the result for the Thorough task (Run 637: utCASartT09) is accepted and is ranked six among participating groups. The baseline performance indicates the performance of Indri search engine in the setting of XML element retrieval.

4.2 Strategies for Overlapping Removal

Within the ad-hoc XML retrieval track, there are four sub-tasks:

- **Thorough Task** asks systems to estimate the relevance of elements in the collection. It returns elements or passages ranked in relevance order (where specificity is rewarded). Overlap is permitted.
- **Focused Task** asks systems to return a ranked list of elements or passages to the user. Overlap is removed. If equally relevant users prefer shorter results over longer ones.
- **Relevant in Context Task** asks systems to return relevant elements or passages clustered per article to the user. For each article, it returns an unranked set of results, covering the relevant material in the article. Overlap is not permitted.

- **Best in Context Task** asks systems to return articles with one best entry point to the user. Overlapping is not allowed.

Because of the hierarchical structure of an XML document, sometimes a parent element is also considered relevant if its child element is highly relevant to a given topic. As a result, we obtain a number of overlapping elements. To fulfill the overlap-free requirement for the Focused task, the Relevant in Context task and the Best in Context task, we adopt the following strategies to remove overlapping element paths based on the result of the Thorough task:

- **Relevance Score:** The result of the Thorough task is scanned from most relevant to less relevant. When overlapped element path is found within a document, the element path with lower relevance score is removed. (see Figure 1).

docID	relevance score	element path
7575091	5.33165	/article[1]/physicist[1]/scientist[1]/chemist[1]
63140	5.30598	/article[1]/scientist[1]/chemist[1]
63140	5.10164	/article[1]/scientist[1]/chemist[1]/chemist[1]/
14353374	5.00247	/article[1]/physicist[1]/scientist[1]/chemist[1]
63140	3.39020	/article[1]/person[1]/alchemist[1]

Fig. 1. Example result of the Focused task (qid=2009005)

The overlap-free result is then grouped by article. For each query, the articles are ranked based on their highest relevance score. For each article, the retrieved element paths keep the rank order of relevance (see Figure 2).

docID	relevance score	element path
7575091	5.33165	/article[1]/physicist[1]/scientist[1]/chemist[1]
63140	5.30598	/article[1]/scientist[1]/chemist[1]
63140	3.39020	/article[1]/person[1]/alchemist[1]
14353374	5.00247	/article[1]/physicist[1]/scientist[1]/chemist[1]

Fig. 2. Example result of the Relevant in Context task (qid=2009005)

For the Best in Context task, we choose the most relevant XML element path of each article as our result.

- **Relevance Score and Full Article Run:** In addition to the relevance score strategy, we combine our overlap-free result with the result of full article runs.

We remove XML element paths whose article does not appear in the result of the full article runs (see Figure 3). We adopt the same strategy for the Reference task as well.

docID	relevance score	element path
7575091	5.33165	/article[1]/physicist[1]/scientist[1]/chemist[1]
14353374	5.00247	/article[1]/physicist[1]/scientist[1]/chemist[1]

Fig. 3. Example result of the Reference task (qid=2009005)

5 Results

For each of the four sub-tasks, we submitted two XML element results and one extra result for the reference task. On the whole, we had 12 submissions to the ad-hoc track.

5.1 Full Document Retrieval

One of our goals for the ad-hoc track is to compare the performance of Indri search engine used in full document retrieval and XML element retrieval. The observation will be used to analyze our element language models and improve our overlapping removal strategies. For official runs, we submitted full document retrieval using both CO and CAS queries for four sub-tasks. Except the Thorough task, our runs were disqualified because of overlapped results. The qualified run for the Thorough task is an automatic run for CAS query (see Table 1).

Table 1. Results of full document retrieval

tasks	performance metrics				
	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
thorough	0.5461	0.5343	0.4929	0.4415	0.2350

5.2 XML Element Retrieval

For official submission, we presented our results using the strategy of **Relevance Score and Full Article Run**. All qualified runs use CAS query. The result of the Thorough task is in Table 2.

The full document retrieval outperforms the element retrieval in locating all relevant information. Our results again agree with the observation in previous INEX results. System wise, the given reference result has better performance.

Table 2. Results of XML element retrieval

tasks	performance metrics				
	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
thorough (utCASeleT09)	0.4364	0.4127	0.3574	0.2972	0.1599
thorough (utCASrefF09)	0.4834	0.4525	0.4150	0.3550	0.1982

Focused Task The official results of the Focused task are in Table 3. Our run (utCASbaseF09) successfully preserves the retrieval result of the Thorough task (utCASeleT09) and brings slightly improvement.

Table 3. Results of XML element retrieval

tasks	performance metrics				
	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
focus (utCASbaseF09)	0.4451	0.4239	0.3824	0.3278	0.1695
focus (utCASrefF09)	0.4801	0.4508	0.4139	0.3547	0.1981

Relevant in Context Task As explained earlier, we rank documents by the highest element score in the collection and rank element paths by their relevance score within the document. Overlapping elements are removed as required. The retrieval result is in Table 4.

Table 4. Results of XML element retrieval

tasks	performance metrics				
	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
relevant-in-context (utCASbaseR09)	0.1966	0.1695	0.1391	0.1054	0.1064
relevant-in-context (utCASrefR09)	0.2216	0.1904	0.1457	0.1095	0.1188

Best in Context Task This task is to identify the best entry point for accessing the relevant information in a document. Our strategy is to return the element path with highest relevance score in a document. The retrieval result is in Table 5.

6 Conclusion

We have limited results for the official runs. As a result, it is hard for us to draw a conclusion with caution. The Indri search engine can provide reasonable result of XML element retrieval. When the result of reference run is used, the result of

Table 5. Results of XML element retrieval

tasks	performance metrics				
	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
best-in-context (utCASeleB09)	0.1795	0.1449	0.1143	0.0875	0.0852
best-in-context (utCASrefB09)	0.1993	0.1737	0.1248	0.0941	0.1056

the Thorough task is improved. This may imply that the search engine is able to locate relevant elements within documents effectively. Its performance depends not only on the element language model but also on the effective formulation of Indri structured query. The step of overlapping removal is the key factor that may harm the retrieval performance. In our case, our result (utCASbaseF09) of the Focused task can even boost the performance.

Except using the relevance score for removing the overlapping element paths, we may try other criteria such as the location of the element within a document. This is especially important for the Best in Context task as users tend to read a document from top-down.

We will present more fruitful discussion when more unofficial results are collected for the formal version of this paper.

Acknowledgments

This work is sponsored by the Netherlands Organization for Scientific Research (NWO), under project number 612-066-513.

References

1. Zhai, C.X., Lafferty, J.: A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Trans. on Information Systems*. 22(2), 179–214 (2004)
2. Schenkel, R., Suchanek, F.M., Kasneci, G.: YAWN: A Semantically Annotated Wikipedia XML Corpus. In *12. GI-Fachtagung fr Datenbanksysteme in Business, Technologie und Web*, Aachen, Germany. (March 2007)
3. Strohman, T., Metzler, D., Turtle, H., Croft, W.B.: Indri: A Language-model Based Search Engine for Complex Queries, In *Proceedings of ICIA* (2005)

OUC's participation in the 2009 INEX Book Track

Michael Preminger¹, Ragnar Nordlie¹, and Nils Pharo¹

Oslo University College

Abstract. In this article we describe the Oslo University College's participation in the INEX 2009 Book track. This year's tasks have been featuring complex topics, containing aspects. These lend themselves to use in both the book retrieval and the focused retrieval tasks. The OUC has submitted retrieval results for both tasks, focusing on using the Wikipedia texts for query expansion, as well as utilizing chapter division information in (a number of) the books.

1 Introduction

This year's task was to try and compare book specific retrieval to generic retrieval for both (whole) book retrieval and focused retrieval. The potential of Wikipedia texts to improve retrieval performance was to be explored.

The most interesting part of this year's topics, which also constitutes the essence of this year's task, is no doubt the Wikipedia text that is supplied with each aspect. The first thing coming to mind is, of course, using the Wikipedia texts for query expansion, which could intuitively provide a precision enhancing device. Those texts are quite long, and the chances of zero hits using the entire text as a query are quite significant. Query expansion needs thus be done with caution.

Whereas a query text (even a test query) is said to originally be formulated by the user, a Wikipedia article does not originate with the user, so that there may be elements in the article that the user would not have endorsed, and thus are unintentional. Used uncritically in a query, those parts may reduce experienced retrieval performance.

A measure to counter this effect would be either using only parts of the Wikipedia text that (chances are that) the user knowingly endorses, or to use the topic title to process the Wikipedia text, creating a version of the latter that is closer to the user's original intention, while still benefitting from the useful expansion potential the text entails.

In investigations involving book retrieval, [1] have experimented with different strategies of retrieval based on query length and document length. Their conclusion has been that basing one's book retrieval on collating results obtained from searching in *book pages* as basic retrieval units (shorter documents), performed better than using *the book as a whole* as a basic retrieval unit. Moreover,

manually adding terms to the query improved page level (shorter document) retrieval, but did not seem to improve retrieval of longer documents (whole books).

We at the OUC wished to pursue this observation, and pose the following questions:

- Can the Wikipedia page partly play the role of manually added terms would play in a batch retrieval situation (laboratory setting)?
- In case it does, would it also benefit users in real life situations?
- What kind of treatment of the Wikipedia text would, in average, give better retrieval?

2 A brief Analysis of a Wikipedia topic text

A Wikipedia text may be quite long. Even if we assume the text is very central to – and a good representative of – the user’s information need, we hypothesize that using the entire text uncritically as a query text or expansion device, would be hazardous.

A glance at the Wikipedia texts supplied with [2] (both on topic and aspect level) leaves the impression that the beginning of the article is quite important. But using even the initial sentences or sections uncritically may result in poor retrieval, or no retrieval at all.

In the beginning of a Wikipedia article, there often occurs a ”to be” (is, are, were a.s.o) or a ”to have” (”has”, ”had” a.s.o.) inflection. The occurrence is not necessarily in the first sentence, but relatively early in the document. The idea is to use this occurrence as an entry point to the important part of the text¹.

Our hypothesis is that on both sides of such an occurrence, one generally finds useful words. There are also grounds to assume that the user, who hunts for book references to this Wikipedia article the way he or she conceives its contents, has read this part of the text and approves of it as representative before proceeding to the rest of the article. This part may arguably be a good representative of the text as seen by the user.

3 Extracting important terms from a Wikipedia article

A way of testing the idea mentioned above is, for each applicable Wikipedia text, to locate the first salient occurrence of a ”to be” or ”to have” inflection (see also footnote above), and define a window of both preceding and succeeding text (while omitting stop-words). The length of the window may vary (See Figure 1. the content of this window is used either as the query or is added to another query as an expansion text.

¹ On a too early occurrence, the second or third occurrence might be considered as an alternative entry.

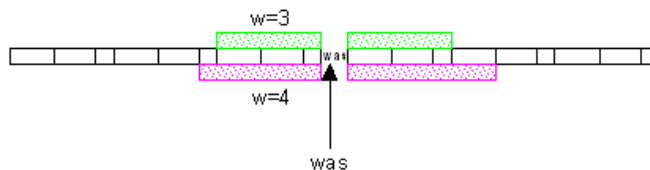


Fig. 1. Using windows of varying widths around an entry word in a Wikipedia text

4 Book-specific retrieval

One of the research objectives of this year’s book track is to compare the performance of generic retrieval methods with more book-specific retrieval methods.

There is probably a large number of possibilities of utilizing book structure in the collection. We have chosen to identify chapter titles with the help of the TOC entries of the book². In addition to the indication of being TOC sections (or lines or words), the marker elements also have references to the page where the referred chapter begins. The chapter names are available and can be used to boost the chances of certain pages (title pages or inner pages of the chapter) to be retrieved as response to queries that include these words.

We create an index for which we identify words in chapter titles, section titles and the like, so we can enhance their significance at query time, and then try to run the expanded queries against this index as well. In practice we identify words constituting chapter or section titles in the TOC section of the book, find the physical location of their respective chapter and add them, specially tagged, to the chapter. this tagging facilitates different weighting of these words related to the rest of the text. Different weighting strategies can then be tested.

For book retrieval it is not important where in the book the title words increase in weight, as they will increase the chances of retrieving this book as a response to a query featuring these words. For focused retrieval we have the limitation that we do not have an explicit chapter partition of the book, only a page partition. One choice of handling this is to identify all pages belonging to a partition, and adding the title words (with enhanced weights) to the text of each page. Within the title page (first page of the chapter) the same title words can be given a different relative weight than for the pages inside the chapter.

² Results may suffer due to the fact that only some 36000 of the 50000 books indeed feature these markup attributes "refid" and "link" of the page and marker element respectively

5 Runs and results

We have been running comparable experiments for book and focused retrieval, using the same index. As the Indri system supports retrieval based on extents, we could define an enclosing extent that would retrieve the entire book, but where the chapter texts were weighted. In focused retrieval we could retrieve page extents, Indri still supporting the weight boosting of chapter titles. We placed chapter titles (enclosed in elements we named *titleinfront* and *titleinside* respectively) in all the pages that constituted a chapter title page or a chapter content page.

In both book retrieval and focused retrieval we have experimented with generic as well as book specific retrieval as described above, using the code pattern as described in table 1.

Table 1. Code pattern of run submissions to the book track

	book retrieval		focused retrieval	
	generic	book specific	generic	book specific
title only	book_to_g	book_to_b	focused_to_g	focused_to_b
title and wiki	book_tw_g#	book_tw_b#	focused_tw_g#	focused_tw_b#
wiki only	book_wo_g#	book_wo_b#	focused_wo_g#	focused_wo_b#

The main partition follows the line of book vs. focused retrieval, so that parallel book and specific runs can be compared. The first row features runs with queries involving topic titles only. The second row has runs where the topic is composed of terms from the topic title and the Wikipedia text, and the third row represents queries composed of the Wikipedia texts only. The hash character is a place holder for the size of the window as described in Section 3, where applicable. In this submission we have experimented with window sizes of 3 and 5 for each run type (as represented by a table cell). This gives a total of 20 runs. The choice of 3 and 5 is somewhat arbitrary, and more extensive experimentation with different combinations will be required.

6 Conclusion

Due to technical problems, results were not available prior to the submission deadline of the pre-proceedings. There is little doubt though, that more experiments in laboratory conditions along this line of research, as well as conditions resembling real life usage of Wikipedia combined with digitized books, will be necessary to approach combinations that will be beneficial in real life situation.

References

1. Wu, M., Scholer, F., Thom, J.A.: The impact of query length and document length on book search effectiveness. In: INEX 2008. (2009) 0–0
2. Kazai, G., Koolen, M.: Inex book search topics for 2009 (2009)

Overview of the INEX 2009 Efficiency Track

Ralf Schenkel^{1,2} and Martin Theobald¹

¹ Max Planck Institute for Informatics, Saarbrücken, Germany

² Saarland University, Saarbrücken, Germany

Abstract. This paper presents an overview of the Efficiency Track that was run for the second time in 2009. This track is intended to provide a common forum for the evaluation of both the effectiveness and efficiency of XML ranked retrieval approaches on *real data* and *real queries*. The Efficiency Track significantly extends the Ad-Hoc Track by systematically investigating different types of queries and retrieval scenarios, such as classic ad-hoc search, high-dimensional query expansion settings, and queries with a deeply nested structure (with all topics being available in both the NEXI-style CO and CAS formulations, as well as in their XPath 2.0 Full-Text counterparts).

1 Introduction

The Efficiency Track was run for the second time in 2009, with its first incarnation at INEX 2008 [2]. It is intended to provide a common forum for the evaluation of both the effectiveness and efficiency of XML ranked retrieval approaches on *real data* and *real queries*. The Efficiency Track significantly extends the Ad-Hoc Track by systematically investigating different types of queries and retrieval scenarios, such as classic ad-hoc search, high-dimensional query expansion settings, and queries with a deeply nested structure (with all topics being available in both the NEXI-style CO and CAS formulations, as well as in their XPath 2.0 Full-Text counterparts).

2 General Setting

2.1 Test Collection

The Efficiency Track uses the INEX-Wikipedia collection³ that has been introduced in 2009, an XML version of English Wikipedia articles with semantic annotations. With almost 2.7 million articles, more than a billion elements and an uncompressed size of approximately 50 GB, this collection is a lot larger than the old Wikipedia collection used in previous years (and for last year's Efficiency track). The collection has an irregular structure with many deeply nested paths, which turned out to be challenging for most systems.

³ available from <http://www.mpi-inf.mpg.de/departments/d5/software/inex/>

2.2 Topic Types

One of the main goals to distinguish the Efficiency Track from traditional Ad-Hoc retrieval is to cover a broader range of query types than the typical NEXI-style CO or CAS queries, which are mostly using either none or only very little structural information and only a few keywords over the target element of the query. Thus, two natural extensions are to extend Ad-Hoc queries with high-dimensional query expansions and/or to increase the amount of structural query conditions without sacrificing the IR aspects in processing these queries (with `topicDescription` and `narrative` fields providing hints for the human assessors or allowing for more semi-automatic query expansion settings, see Figure 1). The Efficiency Track focuses on the following types of queries (also coined “topics” in good IR tradition), each representing different retrieval challenges:

- **Type (A) Topics:** 115 topics (ids 2009-Eff-001—2009-Eff-115) were taken over from the Ad-hoc Track. These topics represent classic, Ad-Hoc-style, focused passage or element retrieval, with a combination of NEXI CO and CAS queries.
- **Type (B) Topics:** Another 115 topics (ids 2009-Eff-116—2009-Eff-230) were generated by running Rocchio-based blind feedback on the results of the article-only AdHoc reference run. These CO topics are intended to simulate high-dimensional query expansion settings with up to 101 keywords, which cannot be evaluated in a conjunctive manner and are expected to pose a major challenge to any kind of search engine. Relevance assessments for these topics can be taken over from the corresponding adhoc topics; a reference run (using TopX2) with the expanded topics was submitted to the adhoc track to make sure that results with the expanded topics were also present in the result pools.
- **Type (C) Topics:** These topics were planned to represent high-dimensional, structure-oriented retrieval settings over a DB-style set of CAS queries, with deeply nested structure but only a few keyword conditions. The focus of the evaluation should have been on execution times. Unfortunately, we did not get any proposals for type (C) topics by the track participants.

2.3 Topic Format

The Efficiency Track used an extension of the topic format of the Adhoc Track. All adhoc fields were identical to the corresponding adhoc topics for type (A) topics. For the type (B) topics, the expanded keyword queries were put into the `title` field and corresponding NEXI queries were generated for the `castitle` field. All topics contained an additional `xpath.title` field with an XPath FullText expression that should be equivalent to the NEXI expression in the `castitle` field. This Xpath FT expression was automatically created from the NEXI expression by replacing `about()` predicates with corresponding `ftcontains()` expressions and connecting multiple keywords within such an expression by `ftand`. After releasing the topics it turned out that `ftand` enforces a conjunctive evaluation of the predicate, which was not the original

intension and especially for the type (B) topics would lead to many topics with empty resultset. The topics were therefore updated and all occurrences of `ftand` replaced by the less strict `ftor`.

```
<topic ct_no="242" id="2009-Eff-057" type="A">
  <title>movie Slumdog Millionaire directed by Danny Boyle</title>
  <castitle>
    //movie[about(., "Slumdog Millionaire")]//director[about(., "Danny Boyle")]
  </castitle>
  <xpath_title>
    //movie[. ftcontains ("Slumdog Millionaire")]
    //director[. ftcontains ("Danny Boyle")]
  </xpath_title>
  <phrasetitle> "Slumdog Millionaire" "Danny Boyle" </phrasetitle>
  <description>
    Retrieve information about the movie Slumdog Millionaire
    directed by Danny Boyle.
  </description>
  <narrative>
    The relevant texts must contain information on: the movie, the awards
    it has got or about the casts and crew of the movie. The criticisms of
    the movie are relevant as well. The other movies directed by Danny Boyle
    are not relevant here. Information about the making of the movie and
    about the original novel and its author is also relevant. Passages or
    elements about other movies with the name "millionaire" as a part of
    them are irrelevant. Information about the game show "Who wants to be a
    millionaire" or any other related game shows is irrelevant.
  </narrative>
</topic>
```

Fig. 1. Example topic (2009-Eff-057)

2.4 Tasks

Adhoc Task. The Efficiency Track particularly encourages the use of top- k style query engines. In the AdHoc task, participants were asked to create top-15, top-150, and top-1500 results with their systems and to report runtimes, using the different title field, including the NEXI CO, CAS, or XPATH titles, or additional keywords from the narrative or description fields. Following the INEX AdHoc Track, runs could be submitted in either Focused (i.e., non-overlapping), Thorough (incl. overlap), or Article retrieval mode:

- **Article:** Here, results are always at the article level, so systems may consider the XML or the plain text version of documents. Results are always free of overlap by definition.
- **Thorough:** The Thorough mode represents the original element-level retrieval mode used in INEX 2003-2005. Here, any element identified as relevant should be returned. Since removing overlap may mean a substantial burden for a system, this setting intentionally allows overlapping results, so query processing times can be clearly distinguished from the time needed to remove overlapping results.

- **Focused:** Focused (i.e., overlap-free) element and/or passage retrieval typically is favorable from a user point-of-view and therefore replaced the Thorough retrieval as primary retrieval mode in the Ad-hoc Track in 2006. Here, the reported run-times include the time needed to remove overlap, which may give rise to interesting comparisons between systems following both Thorough and Focused retrieval strategies.

Budget-Constrained Task. This novel task in 2009 asked participants to retrieve results within a fixed budget of runtime (one of 10ms, 100ms, 1000ms and 10000ms), simulating interactive retrieval situations. Standard top-k algorithms cannot easily be used with such a constraint or may return arbitrarily bad results. However, we did not get any submissions for this task, probably because it required specific modifications to the systems which was too much effort for the participants.

3 Run Submissions

The submission format for all Efficiency Track submissions is defined by the DTD depicted in Figure 7, where the different fields have the following meanings:

- Each *run* submission *must* contain the following information:
 - `participant-id` - the INEX participant id
 - `run-id` - your run id
 - `task` - either `adhoc` or one of the budget-constrained tasks
 - `type` - either `focused`, `thorough`, or `article`
 - `query` - either `automatic` or `manual` mode
 - `sequential` - queries being processed sequentially or in parallel (independent of whether distribution is used)
- Furthermore, each *run* submission *should* contain some basic system and retrieval statistics:
 - `no_cpu` - the number of CPUs (cores) in the system (sum over all nodes for a distributed system)
 - `ram` - the amount of RAM in the system in GB (sum over all nodes for a distributed system)
 - `no_nodes` - the number of nodes in a cluster (only for a distributed system)
 - `hardware_cost` - estimated hardware cost
 - `hardware_year` - date of purchase of the hardware
 - `topk` - top-*k* run or not (if it is a top-*k* run, there may be at most *k* elements per topic returned)
 - `index_size_bytes` - the overall index size in bytes
 - `indexing_time_sec` - the indexing time in seconds to create the indexes used for this run
- Each *run* submission *should* also contain the following brief system descriptions (keywords), if available:

- `general_description` - a general system and run description
 - `ranking_description` - the ranking strategies used
 - `indexing_description` - the indexing structures used
 - `caching_description` - the caching hierarchies used
- Each *topic* element in a run submission *must* contain the following elements:
- `topic_id` - the id of the topic
 - `total_time_ms` - the total processing time in milliseconds: this should include the time for parsing and processing the query but does not have to consider the extraction of resulting file names or element paths (needed to create the above format for the run submission)
- Furthermore, each *topic* element of a run submission *should* contain the following elements:
- `cpu_time_ms` - the CPU time spent on processing the query in milliseconds
 - `io_time_ms` - the total I/O time spent on physical disk accesses in milliseconds
 - `io_bytes` - the number of I/O bytes needed for processing the query. For a distributed system, this should contain the entire amount of bytes spent on network communication.

Particularly interesting for the Efficiency Track submissions is the `runtime` field, of course. This can optionally be split into `cpu_time` and `io_time`, which has been done only by a single participant. We therefore focus on actual wallclock running times as efficiency measure.

4 Metrics

To assess the quality of the retrieved results, the Efficiency Track applied the same metrics and tools used in the Ad-Hoc track. Runs were evaluated with the interpolated precision metric [1]. Like the Adhock Track, we are mainly interested in early precision, so we focus on `iP[0.01]`, but also report mean average `iP` values (MAiP). All runs were first converted to the submission format of the AdHoc Track and then evaluated with the standard tools from that track.

5 Participants

An overall amount of 68 runs was submitted by 4 participating groups using 5 different systems. Here are short system descriptions submitted by the participants.

Max-Planck-Institut Informatik – TopX 2.0 [Part.ID 10] The TopX 2.0 system provides a compressed object-oriented storage for text-centric XML data with direct access to customized inverted files and C++-based implementation of a structure-aware top-k algorithm.

Max-Planck-Institut Informatik – Proximity-Enhanced Top-K [Part.ID 10] Following our work on proximity-based XML retrieval at INEX 2008, we developed a proximity-aware indexing framework at the article level that uses inverted lists for both terms and term pairs and includes an automated pruning step that cuts the size of these lists to a very short prefix with the highest-scoring entries. We evaluate queries through a merge join of the corresponding term and term pair lists, yielding low CPU overhead and fast answer times.

University of Frankfurt – Spirix [Part.ID 16] Spirix is a Peer-to-Peer (P2P) search engine for Information Retrieval of XML documents. The underlying P2P protocol is based on a Distributed Hash Table (DHT). Due to the distributed architecture of the system, efficiency aspects have to be considered in order to minimize bandwidth consumption and communication overhead. Spirix is a top- k search engine aiming at efficient selection of posting lists and postings by considering structural information, e.g. taking advantage of CAS queries. As collections in P2P systems are usually quite heterogeneous, no underlying schema is assumed but schema-mapping methods are of interest to detect structural similarity.

University of Otago [Part.ID 4] We submitted whole document runs. We examined two parameters. One was dynamic pruning of tf-ordered postings lists - this effects the time taken to compute the BM25 score for each document. Computed rsvs were held in an array of accumulators, and the second parameter affected the sorting of this array. An efficient select-top- k -from- n algorithm was used, the second parameter was k .

University of Konstanz – BaseX [Part.ID 304] BaseX is a native XML database and XPath/XQuery processor, including support for the latest XQuery Full Text recommendation. As we put our main focus on efficiency and generic evaluation of all types of XQuery requests and input documents, our scoring model is based on a classical TF/IDF implementation. Additional scoring calculations are performed by XQFT (ftand, ftor, ftnot) and XQuery operators (union, location steps, ...). A higher ranking is given to those text nodes which are closer to the location steps of the input query than others. We decided to stick with conjunctive query evaluation (using 'ftand' instead of 'ftor' in the proposed topic queries), as a change to the disjunctive mode would have led to too many changes, which could not have been reasonably implemented in the remaining time frame. Next, we decided to not extend the proposed queries with stemming, stop words or thesaurus options. As a consequence, many queries might return less results than the TopX reference engine (and sometimes no results at all). To give a realistic picture, we have included both the total time for accessing indexes as well as traversing the inverted specified location paths in our final performance results.

6 Results

Table 1 summarizes important system parameters as they were delivered in the runs' headers, grouping runs with similar id prefixes and settings. The majority of all runs

was submitted for the Focused or Article subtask, only four runs were submitted for the Thorough subtask. Tables 2 and 3 summarize the effectiveness (iP, MAiP) and efficiency (average of wallclock runtimes, cpu and io times in milliseconds) results of all submitted runs for type (A) and type (B) topics. Figure 2 shows the precision/recall diagram for the four Thorough runs and, as a reference, the plot for the (focused) run Eff-20 which has the highest Thorough MAiP among all Efficiency (and in fact also AdHoc) runs. Figures 3 and 4 depict plots that detail the efficiency (runtime) vs. effectiveness (iP[0.01] and MAiP) tradeoff for all runs for the type (A) topics, Figures 5 and 6 show the same for the type (B) topics.

Part.ID	Run prefix	#CPU	RAM	#Nodes	Hardw.Cost	Year
4	Eff	8	8	1	3000 NZD	2008
10	MPI-eff	8	32	1	3000 EUR	2008
10	TopX2-09	4	16	1	5000 EUR	2005
16	Spirix	no information				
304	BaseX	2	32	no information		

Table 1. Run parameters as taken from the submission headers

Regarding efficiency, average running times per topic varied from 8.8 ms to 50 seconds for the type (A) topics and from 367.4 ms to 250 seconds for the type (B) topics. It is important to notice that absolute runtimes across systems are hardly comparable due to differences in the hardware setup and caching. Both the Otago runs and the MPI-prox runs clearly show that the dominant part of retrieval time is spent in IO activity, so improving or reducing IO access could be a promising way to improve efficiency. Most runs (with the exception of a few TopX2 runs) were article-only runs, and like last year these runs generally yielded very good efficiency results. Only TopX2 generated element-level results using the CAS NEXI queries, and it is evident that the additional structure in the queries increases processing time.

Overall effectiveness results were generally comparable to the Ad-hoc Track on the type (A) topics (which are identical to the AdHoc topics), with the best runs achieving a MAiP value of 0.301 and interpolated (early) precision values of 0.589 at 1% recall (iP[0.01]) and 0.517 at 10% recall (iP[0.10]), respectively. Result quality on the type (B) topics was generally slightly worse compared to the type (A) topics, which was probably caused by the extreme query expansion used to generate them, leading to typical problems such as topic drift. This effect is not simply caused by missing assessments: Article-level results for type (B) topics from TopX2 were submitted to the AdHoc track (using their original query ids), and iP[0.01] dropped from 0.6090 (rank 5) to 0.4593 (rank 42) with otherwise unchanged settings.

7 Conclusions

This paper gave an overview of the INEX 2009 Efficiency Track that provided a platform for comparing retrieval efficiency of different systems.

Part.ID	Run ID	T	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP	avg total [ms]	avg CPU [ms]	avg IO [ms]	k	query
Focused/Article												
4	Eff-1	A	0.220	0.215	0.177	0.141	0.060	77.8	20.4	57.3	15	CO
4	Eff-2	A	0.350	0.350	0.315	0.285	0.126	77.1	20.4	56.7	15	CO
4	Eff-3	A	0.480	0.478	0.433	0.359	0.151	77.2	20.0	57.2	15	CO
4	Eff-4	A	0.549	0.540	0.500	0.443	0.184	78.1	21.0	57.1	15	CO
4	Eff-5	A	0.597	0.583	0.532	0.468	0.207	84.5	26.9	57.6	15	CO
4	Eff-6	A	0.598	0.585	0.532	0.475	0.205	101.3	43.0	58.2	15	CO
4	Eff-7	A	0.598	0.585	0.532	0.475	0.204	122.2	64.9	57.3	15	CO
4	Eff-8	A	0.220	0.215	0.177	0.142	0.060	77.7	20.4	57.3	150	CO
4	Eff-9	A	0.355	0.355	0.328	0.305	0.135	76.9	19.9	57.0	150	CO
4	Eff-10	A	0.484	0.482	0.438	0.377	0.187	77.4	20.3	57.2	150	CO
4	Eff-11	A	0.552	0.543	0.513	0.474	0.227	78.3	21.3	57.0	150	CO
4	Eff-12	A	0.600	0.588	0.553	0.510	0.273	83.8	26.9	56.9	150	CO
4	Eff-13	A	0.602	0.589	0.553	0.517	0.278	100.0	42.7	57.3	150	CO
4	Eff-14	A	0.602	0.589	0.553	0.517	0.278	122.2	64.9	57.3	150	CO
4	Eff-15	A	0.220	0.215	0.177	0.142	0.060	76.9	20.3	56.6	1500	CO
4	Eff-16	A	0.355	0.355	0.328	0.305	0.135	77.1	20.2	56.9	1500	CO
4	Eff-17	A	0.484	0.482	0.438	0.377	0.191	77.4	20.1	57.3	1500	CO
4	Eff-18	A	0.552	0.543	0.513	0.474	0.239	78.5	20.9	57.6	1500	CO
4	Eff-19	A	0.600	0.588	0.553	0.510	0.293	83.6	26.8	56.9	1500	CO
4	Eff-20	A	0.602	0.589	0.553	0.517	0.301	100.3	42.7	57.6	1500	CO
4	Eff-21	A	0.602	0.589	0.553	0.517	0.301	121.7	64.3	57.4	1500	CO
10	MPI-eff-1500-1810	A	0.566	0.553	0.532	0.464	0.248	27.1			1500	CO
10	MPI-eff-1500-1810-cold	A	0.566	0.553	0.532	0.464	0.248	287.0			1500	CO
10	MPI-eff-150-610	A	0.574	0.560	0.531	0.466	0.233	13.2			150	CO
10	MPI-eff-150-610-cold	A	0.574	0.560	0.531	0.466	0.233	242.5			150	CO
10	MPI-eff-15-210	A	0.575	0.559	0.511	0.400	0.177	8.8			15	CO
10	MPI-eff-15-210-cold	A	0.575	0.559	0.511	0.400	0.177	216.5			15	CO
10	TopX2-09-Ar-Fo-15-Hot	A	0.598	0.583	0.494	0.397	0.178	84.0			15	CO
10	TopX2-09-ArHeu-Fo-1500-Hot	A	0.597	0.586	0.530	0.475	0.275	301.2			1500	CO
10	TopX2-09-ArHeu-Fo-150-Hot	A	0.598	0.588	0.531	0.474	0.252	87.2			150	CO
10	TopX2-09-ArHeu-Fo-15-Hot	A	0.589	0.577	0.482	0.398	0.175	69.8			15	CO
10	TopX2-09-CAS-Fo-15-Cold	F	0.546	0.480	0.423	0.355	0.138	467.7			15	CAS
10	TopX2-09-CAS-Fo-15-Hot	F	0.545	0.493	0.418	0.350	0.137	379.2			15	CAS
10	TopX2-09-CASHeu-Fo-15-Hot	F	0.525	0.468	0.358	0.304	0.124	234.9			15	CAS
10	TopX2-09-COS-Fo-15-Hot	F	0.645	0.567	0.406	0.285	0.135	125.6			15	CAS
10	TopX2-09-CO-Fo-15-Hot	F	0.641	0.564	0.405	0.291	0.130	338.2			15	CAS
10	TopX2-09-COHeu-Fo-15-Hot	F	0.507	0.429	0.306	0.196	0.079	71.8			15	CAS
16	Spirix09RX01	A	0.621	0.599	0.551	0.499	0.269	50063.5			1500	CAS
16	Spirix09RX02	A	0.435	0.427	0.379	0.319	0.155	46820.5			150	CAS
16	Spirix09RX03	A	0.398	0.372	0.335	0.293	0.130	34663.6			150	CAS
16	Spirix09RX03	A	0.398	0.372	0.335	0.293	0.130	34663.6			150	CAS
16	Spirix09RX04	A	0.402	0.398	0.370	0.295	0.138	44191.5			150	CAS
16	Spirix09RX05	A	0.368	0.333	0.311	0.268	0.121	44352.2			150	CAS
16	Spirix09RX06	A	0.119	0.119	0.119	0.107	0.037	42878.4			150	CAS
16	Spirix09RX07	A	0.604	0.590	0.535	0.497	0.249	959.8			1500	CAS
16	Spirix09RX08	A	0.405	0.403	0.369	0.309	0.140	563.4			150	CAS
16	Spirix09RX09	A	0.354	0.352	0.334	0.283	0.124	496.2			150	CAS
16	Spirix09RX10	A	0.405	0.403	0.386	0.313	0.138	502.6			150	CAS
16	Spirix09RX11	A	0.354	0.344	0.330	0.278	0.127	483.9			150	CAS
16	Spirix09RX12	A	0.119	0.119	0.118	0.099	0.037	474.9			150	CAS
16	Spirix09RX13	A	0.604	0.590	0.535	0.497	0.249	2986.3			1500	CAS
16	Spirix09RX14	A	0.405	0.403	0.369	0.309	0.140	470.5			150	CAS
16	Spirix09RX15	A	0.354	0.352	0.334	0.283	0.124	746.5			150	CAS
16	Spirix09RX16	A	0.405	0.403	0.386	0.313	0.138	1156.6			150	CAS
16	Spirix09RX17	A	0.354	0.344	0.330	0.278	0.127	1863.0			150	CAS
16	Spirix09RX18	A	0.119	0.119	0.118	0.099	0.037	1675.5			150	CAS
16	Spirix09RX19	A	0.621	0.599	0.551	0.499	0.269	47857.1			1500	CAS
16	Spirix09RX20	A	0.435	0.427	0.379	0.319	0.155	46712.3			150	CAS
16	Spirix09RX21	A	0.398	0.372	0.335	0.293	0.130	35746.8			150	CAS
16	Spirix09RX22	A	0.402	0.398	0.370	0.295	0.138	45072.0			150	CAS
16	Spirix09RX23	A	0.368	0.333	0.311	0.268	0.121	44285.8			150	CAS
16	Spirix09RX24	A	0.119	0.119	0.119	0.107	0.037	44256.9			150	CAS
Thorough												
304	2	T	0.133	0.101	0.061	0.045	0.032	11504.4			1500	XPath
304	3	T	0.133	0.101	0.061	0.045	0.032	2553.3			1500	XPath
304	4	T	0.197	0.144	0.109	0.097	0.049	2510.0			1500	XPath
304	5	T	0.209	0.160	0.123	0.107	0.054	2726.7			1500	XPath

Table 2. Effectiveness/efficiency summary of all runs for type(A) topics

Part.ID	Run ID	T	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP	avg total [ms]	avg CPU [ms]	avg IO [ms]	k	query
4	Eff-1	A	0.240	0.238	0.167	0.139	0.048	380.2	32.0	348.3	15	CO
4	Eff-2	A	0.336	0.334	0.322	0.285	0.104	367.5	32.1	335.5	15	CO
4	Eff-3	A	0.350	0.342	0.326	0.293	0.119	367.4	33.4	334.0	15	CO
4	Eff-4	A	0.379	0.379	0.339	0.316	0.119	374.0	41.7	332.2	15	CO
4	Eff-5	A	0.377	0.370	0.334	0.311	0.112	418.0	89.7	328.3	15	CO
4	Eff-6	A	0.392	0.385	0.335	0.298	0.114	511.6	184.9	326.7	15	CO
4	Eff-7	A	0.390	0.383	0.334	0.299	0.112	543.0	217.0	326.0	15	CO
4	Eff-8	A	0.240	0.239	0.179	0.149	0.051	367.2	32.1	335.1	150	CO
4	Eff-9	A	0.340	0.340	0.330	0.295	0.133	367.5	32.1	335.4	150	CO
4	Eff-10	A	0.356	0.352	0.336	0.321	0.162	370.1	33.4	336.7	150	CO
4	Eff-11	A	0.385	0.385	0.350	0.340	0.161	387.6	42.0	345.6	150	CO
4	Eff-12	A	0.386	0.380	0.354	0.335	0.161	419.4	90.0	329.4	150	CO
4	Eff-13	A	0.403	0.397	0.357	0.331	0.165	512.5	185.1	327.5	150	CO
4	Eff-14	A	0.401	0.395	0.356	0.330	0.164	543.5	216.6	326.9	150	CO
4	Eff-15	A	0.240	0.239	0.179	0.149	0.051	368.3	31.8	336.5	1500	CO
4	Eff-16	A	0.340	0.340	0.330	0.296	0.135	369.5	32.3	337.1	1500	CO
4	Eff-17	A	0.356	0.352	0.336	0.321	0.167	378.7	33.3	345.5	1500	CO
4	Eff-18	A	0.385	0.385	0.350	0.341	0.168	378.2	41.8	336.4	1500	CO
4	Eff-19	A	0.386	0.381	0.354	0.335	0.169	421.8	90.1	331.7	1500	CO
4	Eff-20	A	0.403	0.397	0.357	0.331	0.175	533.3	184.9	348.4	1500	CO
4	Eff-21	A	0.401	0.395	0.356	0.330	0.174	551.8	217.5	334.3	1500	CO
10	MPI-eff-1500-1810	A	0.391	0.379	0.337	0.316	0.162	1492.3			1500	CO
10	MPI-eff-1500-1810-cold	A	0.391	0.379	0.337	0.316	0.162	12979.9			1500	CO
10	MPI-eff-150-610	A	0.391	0.379	0.338	0.315	0.157	922.7			150	CO
10	MPI-eff-150-610-cold	A	0.391	0.379	0.338	0.315	0.157	10235.3			150	CO
10	MPI-eff-15-210	A	0.374	0.356	0.304	0.272	0.099	604.3			15	CO
10	MPI-eff-15-210-cold	A	0.374	0.356	0.304	0.272	0.099	7630.4			15	CO
10	TopX2-09-Ar-TOP15-Hot	A	0.440	0.427	0.362	0.315	0.119	4163.2			15	CO
10	TopX2-09-ArHeu-TOP1500-Hot	A	0.443	0.434	0.381	0.358	0.206	2412.0			1500	CO
10	TopX2-09-ArHeu-TOP150-Hot	A	0.443	0.433	0.381	0.358	0.193	2260.2			150	CO
10	TopX2-09-ArHeu-TOP15-Hot	F	0.431	0.418	0.344	0.303	0.118	2205.4			15	CO
10	TopX2-09-CAS-TOP15-Cold	F	0.442	0.428	0.363	0.316	0.119	4352.3			15	CAS
10	TopX2-09-CAS-TOP15-Hot	F	0.440	0.427	0.357	0.315	0.118	4685.1			15	CAS
10	TopX2-09-CASHeu-TOP15-Hot	F	0.431	0.418	0.344	0.303	0.118	2293.1			15	CAS
10	TopX2-09-COHeu-TOP15-Hot	F	0.364	0.329	0.221	0.175	0.066	497.8			15	CO
16	Spirix09RX13	A	0.450	0.440	0.412	0.382	0.194	4199.2			1500	CAS
16	Spirix09RX14	A	0.456	0.439	0.412	0.365	0.172	1712.1			150	CAS
16	Spirix09RX15	A	0.456	0.439	0.412	0.365	0.172	1859.6			150	CAS
16	Spirix09RX16	A	0.456	0.439	0.412	0.365	0.172	2301.6			150	CAS
16	Spirix09RX17	A	0.456	0.439	0.412	0.365	0.172	2953.0			150	CAS
16	Spirix09RX18	A	0.456	0.439	0.412	0.365	0.172	2823.9			150	CAS
16	Spirix09RX19	A	0.444	0.433	0.406	0.390	0.202	250640.3			1500	CAS
16	Spirix09RX20	A	0.484	0.464	0.421	0.383	0.181	249851.5			150	CAS
16	Spirix09RX21	A	0.456	0.439	0.412	0.365	0.172	215376.4			150	CAS
16	Spirix09RX22	A	0.456	0.439	0.412	0.365	0.172	214315.3			150	CAS
16	Spirix09RX23	A	0.456	0.439	0.412	0.365	0.172	214068.9			150	CAS
16	Spirix09RX24	A	0.456	0.439	0.412	0.365	0.172	215025.8			150	CAS

Table 3. Effectiveness/efficiency summary of all runs for type(B) topics

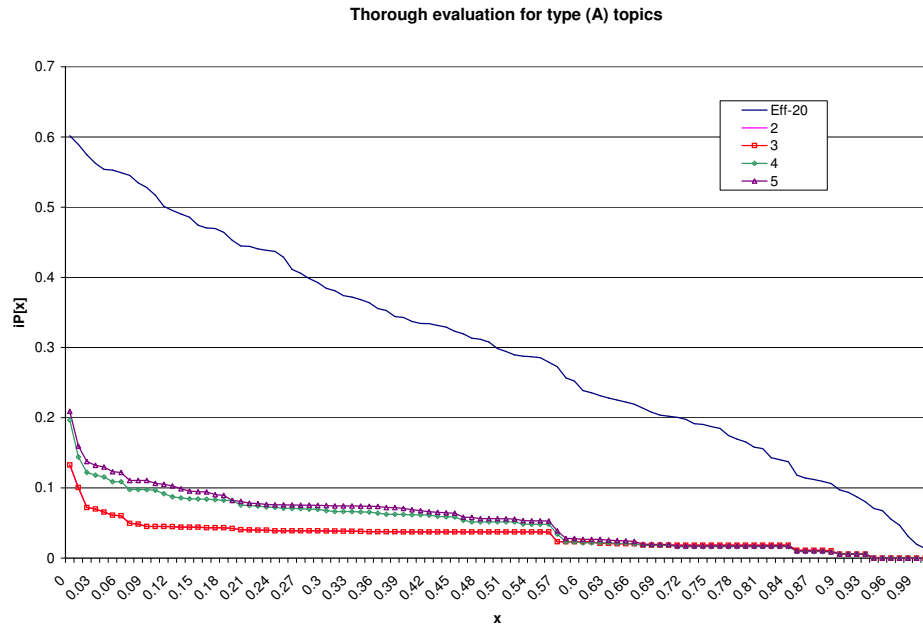


Fig. 2. Precision/recall plots for thorough runs, type (A) topics

References

1. J. Kamps, J. Pehcevski, G. Kazai, M. Lalmas, and S. Robertson. INEX 2007 Evaluation Measures. In N. Fuhr, J. Kamps, M. Lalmas, and A. Trotman, editors, *INEX*, volume 4862 of *Lecture Notes in Computer Science*, pages 24–33. Springer, 2007.
2. M. Theobald and R. Schenkel. Overview of the inex 2008 efficiency track. In S. Geva, J. Kamps, and A. Trotman, editors, *INEX*, volume 5631 of *Lecture Notes in Computer Science*, pages 179–191. Springer, 2008.

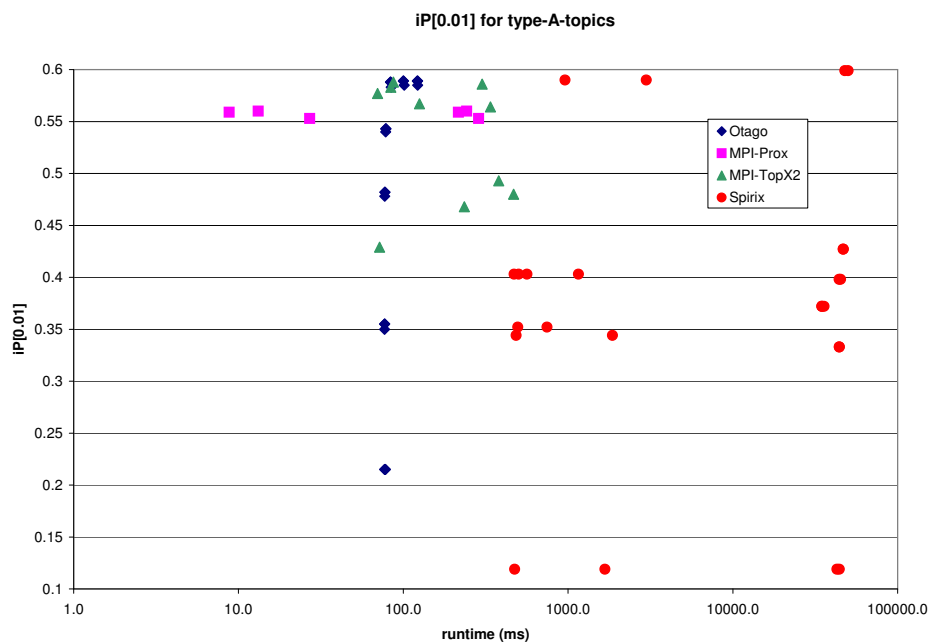


Fig. 3. Runtime vs. early precision plots for Focused and Article runs, type (A) topics

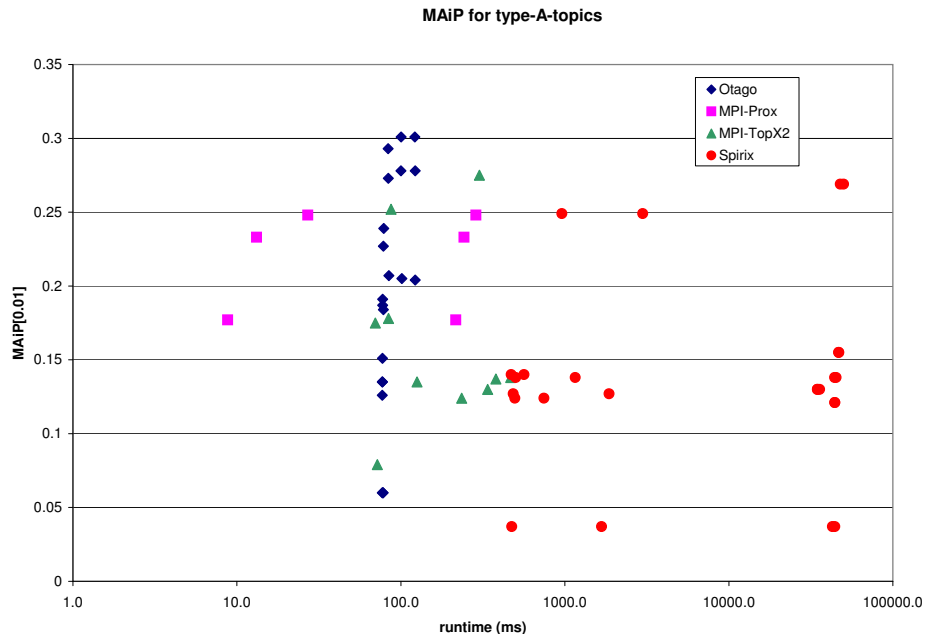


Fig. 4. Runtime vs. MAiP plots for Focused and Article runs, type (A) topics

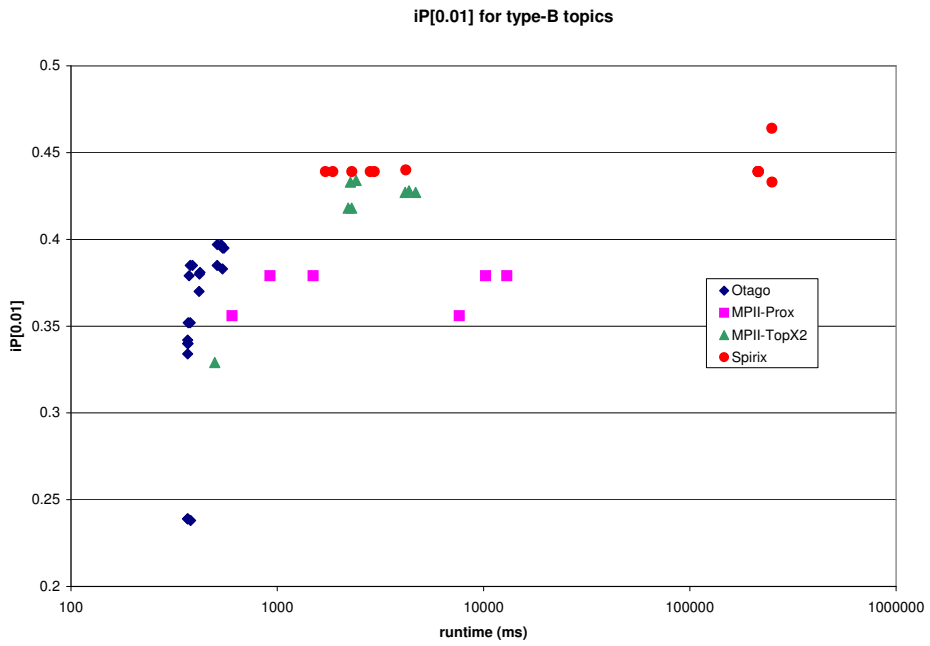


Fig. 5. Runtime vs. early precision plots for Focused and Article runs, type (B) topics

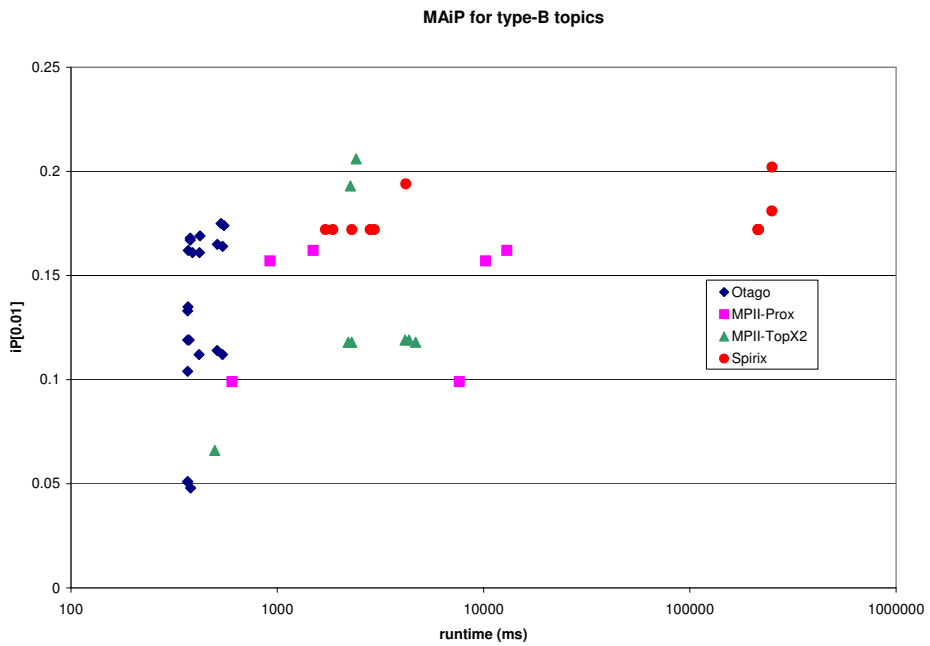


Fig. 6. Runtime vs. MAiP plots for Focused and Article runs, type (B) topics

```

<!ELEMENT efficiency-submission (topic-fields,
                                general_description,
                                ranking_description,
                                indexing_description,
                                caching_description,
                                topic+)>
<!ATTLIST efficiency-submission
  participant-id CDATA #REQUIRED
  run-id        CDATA #REQUIRED
  task          (adhoc | budget10 | | budget100 | budget1000 | budget10000) #REQUIRED
  type          (focused | thorough | article) #REQUIRED
  query         (automatic | manual) #REQUIRED
  sequential    (yes|no) #REQUIRED
  no_cpu        CDATA #IMPLIED
  ram           CDATA #IMPLIED
  no_nodes      CDATA #IMPLIED
  hardware_cost CDATA #IMPLIED
  hardware_year CDATA #IMPLIED
  topk          (15 | 150 | 1500) #IMPLIED
  index_size_bytes CDATA #IMPLIED
  indexing_time_sec CDATA #IMPLIED
>
<!ELEMENT topic-fields EMPTY>
<!ATTLIST topic-fields
  co_title      (yes|no) #REQUIRED
  cas_title     (yes|no) #REQUIRED
  xpath_title   (yes|no) #REQUIRED
  text_predicates (yes|no) #REQUIRED
  description   (yes|no) #REQUIRED
  narrative     (yes|no) #REQUIRED
>
<!ELEMENT general_description (#PCDATA)>
<!ELEMENT ranking_description (#PCDATA)>
<!ELEMENT indexing_description (#PCDATA)>
<!ELEMENT caching_description (#PCDATA)>
<!ELEMENT topic (result*)>
<!ATTLIST topic
  topic-id      CDATA #REQUIRED
  total_time_ms CDATA #REQUIRED
  cpu_time_ms   CDATA #IMPLIED
  io_time_ms    CDATA #IMPLIED
  io_bytes      CDATA #IMPLIED
>
<!ELEMENT result (file, path, rank, rsv?)>
<!ELEMENT file (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT rank (#PCDATA)>
<!ELEMENT rsv (#PCDATA)>

```

Fig. 7. DTD for Efficiency Track run submissions

Index Tuning for Efficient Proximity-Enhanced Query Processing

Andreas Broschart^{1,2} and Ralf Schenkel^{1,2}

¹ Max-Planck-Institut für Informatik, Saarbrücken, Germany

² Saarland University, Saarbrücken, Germany {abrosch,schenkel}@mpi-inf.mpg.de

Abstract. Scoring models that make use of proximity information usually improve result quality in text retrieval. Considering that index structures carrying proximity information can grow huge in size if they are not pruned, it is helpful to tune indexes towards space requirements and retrieval quality. This paper elaborates on our approach used for INEX 2009 to tune index structures for different choices of result size k . To allow for comparison as to retrieval quality with non-pruned index structures, we also depict our results from the Adhoc Track.

1 Introduction

Proximity-enhanced scoring models are known to improve result quality in text retrieval. In the last decade a number of scoring models which integrate content and proximity scores have been proposed, among them the scoring model by Büttcher et al. [2] which we use in a modified version. Schenkel et al. [3] found that index structures for proximity can grow prohibitively large, if they are not pruned. As index pruning is a lossy operation, it risks result quality which translates into lower precision values. Therefore one should also consider precision while cutting index sizes to tolerable levels.

2 Adhoc Track Results

The scoring model we use in INEX 2009 corresponds to the one used in INEX 2008 [1], this time retrieving article elements only. For our contribution we removed all tags from the XML documents in the Official INEX 2009 collection and worked on their textual content only. The last two runs have been submitted to INEX 2009, the first is the non-submitted baseline:

run	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
MPII-COArBM ²	0.5483	0.5398	0.5112	0.4523	0.2392
MPII-COArBP	0.5603	0.5516(26)	0.5361	0.4692	0.2575
MPII-COArBPP	0.5563	0.5477(28)	0.5283	0.4681	0.2566

Table 1. Results for the Adhoc Track: interpolated precision at different recall levels (ranks for iP[0.01] are in parentheses) and mean average interpolated precision

- **MPII-COArBM'**: a CO run that considered the stemmed terms in the title of a topic (including the terms in phrases, but not their sequence) except terms in negations and stop words. We restricted the collection to the top-level article elements and computed the 1,500 articles with the highest $score_{BM25}$ value as described in our last year's INEX contribution [1]. Note that this approach corresponds to standard document-level retrieval. This run is the actual non-submitted baseline to enable a comparison to the submitted runs which all use proximity information.
- **MPII-COArBP**: a CO run which aims to retrieve the 1,500 articles with the highest $score_{BM25} + score_{proximity}$ where $score_{proximity}$ is calculated based on all possible stemmed term pairs in the title of a topic (including the terms in phrases, but not their sequence) except terms in negations and stop words.
- **MPII-COArBPP**: a CO run which is similar to **MPII-COArBP** but calculates the $score_{proximity}$ part based on a selection of stemmed term pairs. Stemmed term pairs are selected as follows: we consider all stemmed tokens in phrases that occur both in the phrasetitle and in the title and are no stop words. The modified phrases in the phrasetitle are considered one at a time to combine term pairs usable to calculate $score_{proximity}$. If the phrasetitle is empty we use approach **MPII-COArBP**.

The results in Table 1 show that computing our proximity score with a subset of term pairs based on information taken from the phrasetitles doesn't improve the iP values compared to using all term pairs. As expected **MPII-COArBP** leads to a slight improvement over **MPII-COArBM'**.

3 Efficiency Track Results

This section describes our effort in INEX 2009 to tune our index structures for efficient query processing, taking into account the expected retrieval quality and index size. As in [3] we use 1) text index lists (TL) which contain, for each term, a set of docid and their BM25 scores ordered by BM25 score and 2) combined index lists (CL) which contain, for each term pair, a set of docid, and both proximity as well as BM25 score information ordered by descending proximity impact. As full, non-pruned indexes will grow huge, we aim at pruning the index structures after a fixed number of entries per list.

The final pruned index is used as input to a merge join which avoids overhead costs of threshold algorithms such as book-keeping of candidates.

To measure retrieval quality one usually compares the retrieval results with a set of relevance assessments. As at the time of tuning we didn't have any relevance assessments, for each number of results k (top-15, top-150, and top-1500), we first built up a groundtruth as a substitute. That groundtruth consists of the first k results obtained through processing the non-pruned BM25 and proximity index structures. Note that this corresponds to the k highest scoring results of **MPII-COArBP**.

The optimization process is supported by Hadoop, an Open Source MapAndReduce framework, which distributes the evaluation and indexing workload across a cluster of 10 server nodes in the same network.

In our recent studies on the TREC .GOV2 collection we found that it was reasonable to use an overlap of $\alpha=0.75$ to achieve the retrieval quality of non-pruned BM25 using pruned TLs and CLs. (Note that the overlap is computed by the amount of overlapping documents and is not based on the number of characters returned.)

For all list lengths l ranging between 10 and 20,000 (step size of 100) we estimate the index size first. We restrict the processing of the query load to those that fit the index size constraint set to 100 GB in our experiments. The shortest list length that fulfills the overlap and the index size constraint is considered the optimal list length l_{opt} . We prefer short list lengths, since we process the pruned lists in a merge join which reads the relevant index structures completely.

Table 2 shows the results of the tuned index structures for type A queries. For performance reasons, tuning was carried out using the type A queries only, type B queries use the same pruned indexes. **MPII-eff-k** depicts the optimal list lengths for different choices of k , the average cold and warm cache running times and interpolated precision values at different recall levels. While measuring the cold cache running times, we have emptied the cache after each query execution, not just after each batch. To collect the warm cache running times, in a first round we fill the cache by processing the complete query load and measure the running times in the second round. The difference between the cold and warm cache running times can be considered as I/O time.

run	l_{opt}	$\varnothing t_{warm}$ [ms]	$\varnothing t_{cold}$ [ms]	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
MPII-eff-15	210	8.8	216.5	0.575	0.559	0.511	0.4	0.177
MPII-eff-150	610	13.2	242.5	0.574	0.56	0.531	0.466	0.233
MPII-eff-1500	1810	27.1	287.0	0.566	0.553	0.532	0.464	0.248

Table 2. Efficiency Track results, type A queries

Queries are processed using the pruned index structures which have been reordered by docid to enable for merge join query processing. As the pruned index is created by Hadoop and stored in a MapFile accessed by Hadoop in a non-optimized way during query execution, we think that there’s still room for performance improvements. It turns out that already very short list prefixes are sufficient to lead to a result quality comparable to **MPII-COArBP** at early recall levels (until iP[0.01]) and to **MPII-COArBM’** at later recall levels.

run	l_{opt}	$\varnothing t_{warm}$ [ms]	$\varnothing t_{cold}$ [ms]	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
MPII-eff-15	210	604.3	7,630.4	0.374	0.356	0.304	0.272	0.099
MPII-eff-150	610	922.7	10,235.3	0.391	0.379	0.338	0.315	0.157
MPII-eff-1500	1810	1,492.3	12,979.9	0.391	0.379	0.337	0.316	0.162

Table 3. Efficiency Track results, type B queries

Table 3 shows the results of the tuned index structures for type B queries. It is clear that in our setting type B queries that consist of partly more than

100 keywords cannot be executed as fast as type A queries. Many thousands of possible single pair lists per query have to be fetched from harddisk first before the evaluation can start.

4 Conclusion

This paper has presented an approach to perform index pruning in a retrieval-quality aware manner to realize performance improvements and smaller indexes at the same time.

References

1. A. Broschart, R. Schenkel, and M. Theobald. Experiments with proximity-aware scoring for xml retrieval at inx 2008. In S. Geva, J. Kamps, and A. Trotman, editors, *INEX*, volume 5631 of *Lecture Notes in Computer Science*, pages 29–32. Springer, 2008.
2. S. Büttcher, C. L. A. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *SIGIR*, pages 621–622, 2006.
3. R. Schenkel, A. Broschart, S. won Hwang, M. Theobald, and G. Weikum. Efficient text proximity search. In N. Ziviani and R. A. Baeza-Yates, editors, *SPIRE*, volume 4726 of *Lecture Notes in Computer Science*, pages 287–299. Springer, 2007.

INEX Efficiency Track meets XQuery Full Text in BaseX

Sebastian Gath, Christian Grün, Alexander Holupirek, and Marc H. Scholl

<firstname>.<lastname>@uni-konstanz.de
Department of Computer & Information Science
Box D 188, 78457 Konstanz, Germany
University of Konstanz

Abstract. BaseX is an early adopter of the upcoming XQuery Full Text Recommendation. This extended abstract describes the challenges of joining the INEX Efficiency Track using BaseX, an XML database system. We will describe some of the problems we encountered during the application of XQuery Full Text to the INEX topic set and discuss the remaining comparability problem.

1 Introduction: BaseX and XQuery Full Text

The existence of more than fifty XQuery processors clearly underlines the large interest in querying XML documents and collections. While many of the database-driven implementations offer their own extensions to support full-text requests, the upcoming XPath and XQuery Full Text 1.0 Recommendation [1] (XQFT) aims to satisfy the need for a unified language extension and might attract more developers and users from the Information Retrieval community. The recommendation offers a wide range of content-based query operations, classical retrieval tools such as Stemming and Thesaurus support, and an implementation-defined scoring model that allows developers to adapt their database to a large variety of use cases and scenarios. BaseX is, to the best of our knowledge, the first implementation to fully support all features of the specification. More implementations are expected to follow in the near future as soon as the recommendation has reached its final state.

In this extended abstract, we present some aspects encountered during our participation in the INEX Efficiency Track 2009 using our XQFT-based database system BaseX [3, 4, 6]. Due to the complexity of the language extension, we will focus on its NEXI related features. The first Ad-Hoc query of the INEX track is depicted as an example:

```
//article[.//text() ftcontains ("Nobel" ftand "prize")]
```

First of all, there is a path condition `//article` in front of the predicate, which demands that each node on the descendant-or-self-axis of the document root

with an element tag `article` has to be taken into further considerations. Additionally, the predicate filters all descendant text nodes of each `article` element for the tokens "Nobel" and "prize". The existence of at least one text node containing both tokens yields a valid result of the query. Changing the full-text operand from conjunction to disjunction would lead to additional results containing at least one of the tokens. In general, any XQFT query has at least one input item and at least one full-text condition, and valid results always have to satisfy both conditions. Keeping this in mind, the following observations try to give an insight into the challenges we encountered during our participation in INEX 09 Efficiency Track using an XQFT based engine.

2 Proceedings

First of all, we have introduced a query rewriting step to take advantage of the internal query optimization in BaseX. In detail, we have modified the atomization of context nodes in the given queries:

```
(1) //article[. ftcontains ("Nobel" ftand "prize")]
(1') //article[.//text() contains ("Nobel" ftand "prize")]
```

The atomization of the the context node in the first query (1) purges element information from the subtree of each `article` node and concatenates the text nodes of its descendants. That means, in a valid result, the tokens "Nobel" and "prize" do not necessarily have to occur in the same text node. Our rewritten version (1') applies the search on the specified terms on each text node, which is contained in the subtree of each `article` node, and the terms are now expected to occur in the same text node. This rewriting allows us to use the full-text index in BaseX, which is optimized to run on single text nodes. This way, all kinds of XQuery location paths can be traversed, independent from the structure of the input document. The results of the rewritten queries represent a subset of the atomized query results, and the usage of `FTOr` instead of `FTAnd`, as proposed for the efficiency track, might lead to results similar to the original query, because the tokens do not have to occur in the same text node.

A second type of query rewritings keeps the original semantics:

```
(1) //article[.//((sec|p) ftcontains ("mean" ftand "average" ftand
"precision" ftand "reciprocal"))]
(2) //article[.//sec ftcontains ("mean" ftand "average" ftand
"precision" ftand "reciprocal") or .//p ftcontains ("mean" ftand
"average" ftand "precision" ftand "reciprocal")]
```

Here, the replacement of `|` in (1) by a union operator (2) is helpful to trigger index-based query processing in BaseX, and results in less compact but equivalent queries. This optimization step is currently not applied automatically by

the query compiler due to some side-effects, which are not relevant for the INEX context.

After this pre-processing step, the BaseX query optimizer performs further internal rewritings and applies the full-text index structure whenever possible and cheap enough. Disjunct or conjunct full-text tokens within an XQFT expression are evaluated by the index as well. A cost-based evaluation strategy is applied on queries with several possible index requests (which is the case, e.g., if queries have two or more full-text predicates). Details can be found in [5].

Next, we have extended our full-text index structure by including a normalized, TF/IDF-based scoring value for each indexed token. The score is pre-computed at database creation time and is accessible at query time without additional computations. Individual scores values for several terms are generally calculated by full-text operands. The XQFT operators `FTAnd`, `FTOr` and `FTNot` are taken into further considerations because of their relevance in the INEX context. We have introduced a minimum-based scoring for `FTAnd`, i.e., the minimum of several score values is adopted in an `FTAnd` expression. For `FTOr`, we are using a maximum-based scoring approach, which is similar to the `FTAnd` based score, and for `FTNot`, the inverted score value is returned ($1 - score$) [7]. To get XQFT and NEXI queries semantics closer, NEXI-like scoring for `FTAnd` and `FTOr` operands would be interesting. Axis steps performed in a query can influence score values as well (score propagation). Due to the high heterogeneity of the INEX documents and a lack of well-known solutions, we have chosen a simple, but efficient approach, multiplying a score value with a constant for each location step.

All BaseX query results conform to the typical structure of XQuery results. To comply with the INEX submission format, the path for each topic result had to be extracted out of the query results, which was realized by a simple XQuery function:

```
declare namespace basex = "http://www.basex.com";
declare function basex:sum-path ( $n as node()? )
as xs:string {
  string-join( for $a in $n/ancestor-or-self::*
    let $ssn := $a/../*[name() = name($a)]
    return concat(name($a), '[' ,
      basex:index-of($ssn, $a), ']' , '/' );
declare function basex:index-of (
  $n as node()* , $ntf as node() ) as xs:integer* {
  for $s in (1 to count($n))
  return $s[$n[$s] is $ntf]};
```

The function `basex:sum-path($n as node() ?) as xs:string` returns the node path from the root node to `$n` by traversing the document tree and caching each element and its index.

3 Similarities and Differences between XQuery Full Text and NEXI

INEX, as an extension of XPath, transfers the semantics defined in the XPath queries to the evaluation system [8]. Generally the semantic of XPath queries is well defined by the query grammar and independent of the evaluation system. In contrast the semantics of INEX queries is more independent and depends mostly on the engine. Since the INEX Efficiency Track offers for each topic a NEXI and an XQFT query, the contest attracts new XML database systems with XQFT implementations, and leads to additional competition with established IR systems. However, to compare performance or measure the quality of results, semantically equivalent queries for database and information retrieval systems are crucial.

Basically, there is a main big difference between XML database systems using XQuery and IR systems using with NEXI. The XQuery 1.0 Recommendation [2] guarantees that the same query returns the same result on any system that is processing XQueries. Even if scoring is applied in the query, which is totally implementation defined in XQFT, combined with top-k conditions, the set and structure of the results will always be the same. But the visible results and their ordering may change because of a scoring based ordering and the top-k condition. In contrast, NEXI engines could have a major influence on the semantics of a query and do not provide a guarantee like that. This observation leads to an incomparability of simple XQFT- and NEXI-based queries.

In detail, there are two major differences between the query languages:

```
//article[about(./p, "information retrieval")]
```

Considering the query above, the strict interpretation demands that article tags are returned as a result, and the path condition within the predicate (`./p`) is handled as a suggestion for the information retrieval search. Even the `//article` path condition can be dealt with as (only) a suggestion [8].

The example above illustrates the hint-like behavior in structural conditions in NEXI-based queries even in the strict interpretation compared to XQuery, where there is always a strict path condition in a query which has to be evaluated. To get some additional freedom, the combination of different path conditions with a logical OR could be useful, but in this case, the user has to know the structure of the document. Alternatively, the usage of a unspecified path condition is possible, but this leads to a completely different query and many irrelevant results.

Another point is the interpretation of AND and OR in the NEXI context:

- (1) `//article[about(., apple) and about(., computer)]`
- (2) `//article[about(., apple) or about(., computer)]`

The first query (1) will return article elements from documents about "apple" and about "computer", the second (2) about "apple" or about "computer". The predicate conditions, however, are only hints. Looking at the loose interpretation, the AND is interpreted as **ANDish**, **OR** as **ORish**. In that case the contained Boolean operators are rather hints on how to resolve the information need [8].

The strict interpretation is close to the **FTAnd** and **FTOr** operands, but is still semantically inequivalent, due to the hint behavior in NEXI. Next, looking at the loose interpretation, there is no similarity to the full-text operands. To get this hint behavior and **ANDish** or **ORish** interpretation, a scoring model is needed which, for example, scores results having more **OR** disjunct search tokens higher than results having less disjunct results. Scoring-oriented query processing is very much dependent on the input data. A general-purpose database systems, such as BaseX, aims to perform well in all kinds of applications. Therefore we decided not to add INEX-specific meta information to the database and indexing engine, such as a scoring priority for certain elements, etc. The scoring model does not have any additional information about the input data, therefore a proper solution could be the usage of individual scoring models depending on the use case. For example, in some use cases it might be reasonable to include structural knowledge of the document in the scoring model, in other use cases this does not create an additional value.

In a nutshell, we believe that the semantic equivalence between the NEXI and XQFT queries in the Efficiency Track is not given. As a consequence, NEXI- and XQFT-based submissions might not be directly comparable at the current stage. The semantic distance between the queries increase with their complexity, and, to get almost semantically similar queries, the fuzziness of the NEXI operands would have to be mapped to XQFT. This might lead to verbose XQFT queries with numerous sub-queries and, most likely, longer evaluations times. Alternatively, additional knowledge on the document structure would have to be utilized to get equivalent results.

References

1. Sihem Amer-Yahia et al. XQuery and XPath Full Text 1.0. W3C Candidate Recommendation. <http://www.w3.org/TR/xpath-full-text-10>, May 2008.
2. Scott Boag et al. XQuery 1.0: An XML Query Language. W3C Recommendation. <http://www.w3.org/TR/xquery>, January 2007.
3. Christian Grün et al. Pushing XPath Accelerator to its Limits. In *Proc. of the 1st ExpDB Workshop*, Chicago, Illinois, USA, 2006.
4. Christian Grün et al. Visually Exploring and Querying XML with BaseX. In *Proc. of the 12th BTW Conference, Demo Tracks*, pages 629–632, Aachen, Germany, 2007.
5. Christian Grün et al. XQuery Full Text Implementation in BaseX. In *Proc. of the 6th International XML Database Symposium (XSym 2009)*, pages 114–131, Lyon, France, 2009.
6. Alexander Holupirek et al. BaseX & DeepFS: Joint Storage for Filesystem and Database. In *Proc. of the 12th EDBT Conference*, pages 1108–1111, 2009.

7. Gerard Salton. Extended Boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.
8. Andrew Trotman. Narrowed Extended XPath I (NEXI). In *Lecture Notes in Computer Science*, pages 16–40, Berlin / Heidelberg, 2005.

TopX 2.0 at the INEX 2009 Ad-Hoc and Efficiency Tracks

–

Distributed Indexing for Top-*k*-Style Content-And-Structure Retrieval

Martin Theobald¹, Ablimit Aji², and Ralf Schenkel³

¹ Max Planck Institute for Informatics, Saarbrücken, Germany

² Emory University, Atlanta, USA

³ Saarland University, Saarbrücken, Germany

Abstract. This paper presents the results of our INEX 2009 Ad-hoc and Efficiency track experiments. While our scoring model remained almost unchanged in comparison to previous years, we focused on a complete redesign of our XML indexing component with respect to the increased need for scalability that came with the new 2009 INEX Wikipedia collection, which is about 10 times larger than the previous INEX collection. TopX now supports a CAS-specific distributed index structure, with a completely *parallel* execution of all indexing steps, including parsing, sampling of term statistics for our element-specific BM25 ranking model, as well as sorting and compressing the index lists for our final inverted block-index. Overall, TopX ranked among the top 3 systems in both the Ad-hoc and Efficiency tracks, with a maximum value of 0.61 for iP[0.01] and 0.29 for MAiP in focused retrieval mode at the Ad-hoc track. Our fastest runs achieved an average runtime of 72 ms per CO query, and 235 ms per CAS query at the Efficiency track, respectively.

1 Introduction

Indexing large XML collections for Content-And-Structure (CAS) retrieval consumes a significant amount of time. In particular inverting (i.e., sorting) index lists produced by the XML parser constitutes a major bottleneck in managing very large XML collections such as the 2009 INEX Wikipedia collection, with 55 GB of XML sources and more than 1 billion XML elements. Thus, for our 2009 INEX participation, we focused on a complete redesign of our XML indexing component with respect to the increased need for scalability that came with the new collection. Through distributing and further splitting the index files into multiple smaller files for sorting, we managed to break our overall indexing time down to less than 20 hours on a single-node system and less than 4 hours on a cluster with 16 nodes for the complete CAS index.

As TopX originally aims at CAS queries, our basic index units are inverted lists for combined tag-term pairs, where the occurrence of each term in an XML element is propagated “upwards” the XML tree structure and the term is bound to the tag name of each element that contains it (see [8]). Content-Only (CO) queries are treated as CAS queries with a virtual \ast tag. Term frequencies (TF) and element frequencies (EF) are

computed for each tag-term pair in the collection individually. In summary, we used the same XML-specific extension to BM25 (generally known as EBM25) as in last years also for the 2009 Ad-hoc track and Efficiency tracks. For the EF component, we precompute an individual element frequency for each distinct tag-term pair, capturing the amount of tags under which the term appears in the entire collection. Because of the large size of the new Wikipedia collection, we approximate these collection-wide statistics by sampling over only a subset of the collection before computing the actual scores. New for 2009 was also the introduction of a static decay factor for the TF component to make the scoring function favor smaller elements rather than entire articles (i.e., the root of the documents), in order to obtain more diverse results in focused element retrieval mode (used in our two best Ad-hoc runs `MPII-COFoBM` and `MPII-COBIBM`).

2 Scoring Model

Our XML-specific extension to the popular Okapi BM25 [5] scoring model, as we first introduced it for XML ranking in 2005 [9], remained largely unchanged also in our 2009 setup. It is very similar to later Okapi extensions in [4, 6]. Notice that regular text retrieval with entire documents as retrieval units is just a special case of the below ranking function, which in principle computes a separate Okapi model for each element type individually.

Thus, for content scores, we make use of collection-wide element statistics that consider the *full-content* of each XML element (i.e., the recursive concatenation of all its descendants' text nodes its XML subtree) as a bag of words:

- 1) the *full-content term frequency*, $ftf(t, n)$, of term t in an element 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 element 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 element nodes with tag A that contain t in their full-contents in the entire corpus.

The score of a tag-term pair of an element node n with tag name A with respect to a content condition of the form `//A[about(., t)]` (in NEXI [10] syntax), where A either matches the tag name A or is the tag wildcard $*$, is then computed by the following BM25-based formula:

$$score(n, //T[about(., t)]) = \frac{(k_1 + 1) ftf(t, n)}{K + ftf(t, n)} \cdot \log \left(\frac{N_A - ef_A(t) + 0.5}{ef_A(t) + 0.5} \right)$$

$$\text{with } K = k_1 \left((1 - b) + b \frac{\sum_{t'} ftf(t', n)}{\text{avg}\{\sum_{t'} ftf(t', n') \mid n' \text{ with tag } A\}} \right)$$

For 2009, we used values of $k_1 = 2.0$ and $b = 0.75$ as Okapi-specific tuning parameters, thus changing k_1 from the default value of 1.25 (as often used in text retrieval) to 2.0 in comparison to 2008 (see also [1] for tuning BM25 on INEX data). Consequently,

for an `about` operator with multiple terms, the score of an element satisfying this tag constraint is computed as the sum over all the element's content scores, i.e.:

$$score(n, //T[about(. , t_1 \dots t_m)]) = \sum_{i=1}^m score(n, //T[about(. , t_i)])$$

Moreover, for queries with multiple support elements (like for example in the query `//A//B[about(. , t)]`), we assign a small and constant score mass c for each supporting tag condition that is matched (like for the tag `A` in this example). This structural score mass is then aggregated with the content scores, again using summation. In our INEX 2009 setup (just like in 2008), we have set $c = 0.01$. Note that our notion of tag-term pairs enforces a strict matching of a query's target element, while content conditions and support elements can be relaxed (i.e., be skipped on-the-fly) by the non-conjunctive query processor [8]. Also, content scores are normalized to $[0, 1]$.

2.1 2009 Extensions

Decay Factor for Term Frequencies. New for 2009 was the introduction of a static decay factor for the *ftf* component to make the scoring function favor smaller elements rather than entire articles (i.e., the root of the documents), in order to obtain more diverse results in focused element retrieval mode. With respect to the fairly deep structure of the new 2009 collection, we chose a relatively high decay factor of 0.925. That is, in addition to summing up the *ftf* values of each tag-term pair among the children of an XML element (recursively upwards to the root), we also multiply the *ftf* value from each of the child nodes by 0.925 before propagating these values upwards.

Sampling for Element Frequencies. With very large, diverse XML collections and very many distinct (but mostly infrequent) tag-term pairs, exact element frequencies as needed for the *ef* component cannot easily be kept in memory anymore. An additional difficulty in a distributed indexing setting is that these statistics need to be shared among peers. Therefore, we introduced a sampling phase for these combined tag-term frequencies, which however generates approximate statistics only. During the sampling phase, we scan (i.e., keep) only a fixed amount of tag-term statistics in memory from the XML parser output at each of the distributed nodes in the network individually. Tag-term pairs for which no statistics are kept in memory after this sampling phase are smoothed by an *ef* value of 1 when materializing the above scoring function.

3 Distributed Indexing

Indexing an XML collection with TopX consists of a 3-pass process: 1) parsing the XML documents (using a standard SAX parser) and hashing individual tag-term pairs and navigational tags into a distributed file storage; 2) sampling these files for the BM25-specific *ef* statistics for all tag-term pairs and materializing the BM25 model; and 3) sorting these BM25-scored files to obtain their final inverted block structure and compressing the blocks into a more compact binary format. While we are keeping all

intermediate index files of steps 1 and 2 in a simple (GZip'ed) ASCII format, our final block-index structure created in step 3 is stored in a customized (compressed) binary format as described in [7], which can be decompressed much faster than a GZip format.

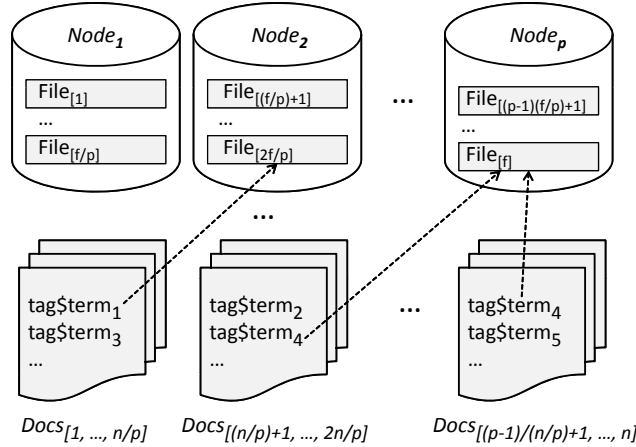


Fig. 1. Two-level hashing of tag-term pairs onto network nodes and index files.

The basic hashing phase is illustrated in Figure 1. We are given a collection of n XML documents yielding m distinct tag-term pairs, f files to hold our inverted index, and p distributed nodes (e.g., a compute cluster, or peers in a network). Before we start the actual indexing phase, the document collection is partitioned into n/p equally sized chunks, and the chunks are distributed over the p nodes. During indexing, every compute node is used for parsing and storing the index files at the same time, i.e., every node has write access to every other node in the network. Let $hash(t_i)$ be the hash code of tag-term pair t_i for all $i = 1, \dots, m$, then $(hash(t_i) \bmod f)$ denotes the file identifier where the inverted list for t_i is stored. Moreover, $(hash(t_i) \bmod f \bmod p)$ then denotes the node identifier at which the file containing t_i is located. This ensures that all tag-term pairs from the entire collection that share the same hash code are stored on the same node and in the same index file. The reason to further partition the index into $f \geq p$ files is that, on each compute node, we can sort multiple such files concurrently in a multi-threaded fashion. Multiple smaller files can of course be sorted more efficiently than a single large file. Thus, every index file contains at least one but possibly more inverted lists. For our INEX 2009 indexing experiments, we used $f = 256$ index files which were distributed over $p = 16$ nodes.

This simple two-level hashing allows for a MapReduce-like [2] but highly specialized form of distributed indexing. After the initial parsing and hashing phase (corresponding to the Map phase in MapReduce), all files needed for materializing the above scoring function are readily available per node, and thus the sampling and scoring can be kept perfectly parallel (corresponding to the Reduce phase in MapReduce). Since all nodes can operate independently in the second phase, this approach allows for a substantially more lightweight Reduce phase than in a classical MapReduce setting. The only data structure that is finally shared across all nodes is a dictionary (see [7]) that

maps a tag-term key from a query back to a file (including the byte-offset within that file) as entry point to the respective inverted list in the distributed storage. Our dictionary maps a 64-bit hash key computed from the combined tag-term string of t_i onto a 64-bit value whose upper 8 bits encode the node id, whose middle 12 bits encode the file id that contains the corresponding inverted list, and whose lower 44 bits encode the byte offset within that file in order to mark the beginning of the inverted list. In this 64-bit setting, we can address up to $2^8 = 256$ nodes with up to $2^{12} = 4,096$ files, each file with a maximum size of $2^{44} = 16$ Terabytes. Of course, the same hash function needs to be used for both indexing and query processing.

This mapping step works particularly well for a CAS-based index, as it is using tag-term pairs as keys to access the inverted lists. In particular, the resulting index files are more uniform in length as compared to a plain (i.e., CO-style) text index because of the much larger amount of distinct tag-term keys in the CAS case. As such, it can scale to almost arbitrary amounts of index files and nodes in a network. Also, a similar hashing step is performed for the inverted tag lists, which results in a distinct set of inverted files for the navigational tag conditions as needed by TopX (see [7]). These files are less uniform in size (due to the much lower amount of distinct tag keys in the collection), but they are anyway much more compact and can benefit from a higher cache locality.

4 Query Processing

TopX is a top- k engine for XML with non-conjunctive XPath evaluations. It supports dynamic top- k -style index pruning for both CO and CAS queries. In dynamic pruning, the traversal of index list scans at query processing time can be pruned early, i.e., when no more candidate elements can make it into the top- k list anymore. Also, since TopX was designed as a native XML engine based on element retrieval, relevant passages are identified based on the XML elements that embrace them.

4.1 Retrieval Modes

Article Mode. In Article mode, all CO queries (including `//*` queries) are rewritten into `/article` CAS conditions. Thus we only return entire `article` elements as target element of the query. This mode conforms to a regular document retrieval mode with our BM25 model collapsing into a document-based scoring model (however including the per element-level decay factor of the TF component). Article mode can thus be seen as a simplest-possible form of focused element retrieval, as entire articles are always guaranteed to be overlap-free.

CO Mode. In CO mode, any target element of the collection is allowed as result. CO queries are processed by TopX equivalently to queries with `//*` as structural condition and exactly one `about` operator with no further structural constraints as filter predicate. The `*` is treated like a virtual tag name and is fully materialized into a distinct set of corresponding `*`-term pairs directly at indexing time. That is, a `*`-term pair is always generated in addition to any other tag-term pair, with individual TF and EF statistics for these `*` tags, which roughly doubles the index size. As a new extension in our 2009

experiments, we cut off very small elements of less than 24 terms (as a form of static index pruning) from these CO-related index lists. At query processing time, a CO query can be processed by directly accessing these precomputed (inverted) CO lists.

CAS Mode. In CAS mode, TopX allows for the formulation of arbitrary path queries in the NEXI [10] or XPath 2.0 Full-Text [10] syntax. As an optimization, leading `*` tags are rewritten as `article` tags, for CAS queries that were otherwise led by a `//*` step.

4.2 Optimizations

Focused, Best-In-Context, and Thorough Modes. We performed experiments with Focused, Best-In-Context and Thorough runs. Element overlap in the *Focused* and *Best-In-Context* retrieval modes is eliminated on-the-fly during query processing by comparing the pre- and post-order labels [9] of elements already contained in the top- k list with the pre- and post-order label of each element that is about to be inserted into the top- k list. During query processing, this top- k list is a constantly updated queue. Thus, if an element is a descendant of another parent element that already is in the top- k queue and the descendant has a higher score than its parent, then the parent is removed from the queue and the descendant is inserted; if otherwise an element is a parent of one or more descendants in the top- k queue, and the parent has a higher score than all the descendants, then all the descendants are removed from the top- k queue and the parent is inserted. In *Thorough* retrieval mode, this overlap check is simply omitted.

Caching. Entire index lists (or their prefixes in case of dynamic top- k pruning) can be cached by the TopX engine, and the decoded and decompressed data structures for each index list can directly be reused by the engine for subsequent queries. Thus, when running over a *hot cache*, only joins and XPath evaluations are carried out over these cached index lists at query processing time, while physical disk accesses can be almost completely avoided for query conditions that were already processed in a previous query.

5 Results

We next present the results for our Ad-hoc and Efficiency track experiments. While indexing was performed on a distributed cluster of 16 nodes, the final runs were performed on a single 3Ghz AMD Opteron Quad-Core machine with 16 GB RAM and a RAID5 storage, with all index files being copied to the local storage.

5.1 Ad-Hoc Track

Figures 2, 3 and 4 depict the $iP[x]$ and $gP[x]$ plots of all runs submitted to the Ad-Hoc track as functions of the recall x , for Focused, Best-In-Context and Thorough modes, respectively (see also [3] for an overview of current INEX metrics). At $iP[0.01]$ (Focused and Thorough) and $gP[0.01]$ (Best-In-Context), the best TopX runs rank at positions 3, 3 and 7, respectively, when grouping the runs by participant id (runs denoted

as p10-MPII-COFoBM and p10-MPII-COBIBM). For all retrieval modes, we generally observe a high early (interpolated or generalized) precision rate (i.e., for iP[0.01] and gP[0.01], each at a recall of 1%), which is an excellent behavior for a top- k engine. Both our top runs used a true focused element retrieval mode, with CO queries being rewritten into `//*` CAS queries, i.e., any result element type was allowed as result. Thus our CO runs achieved higher iP[0.01] (and MAiP) values than our Article runs, as opposed to 2009.

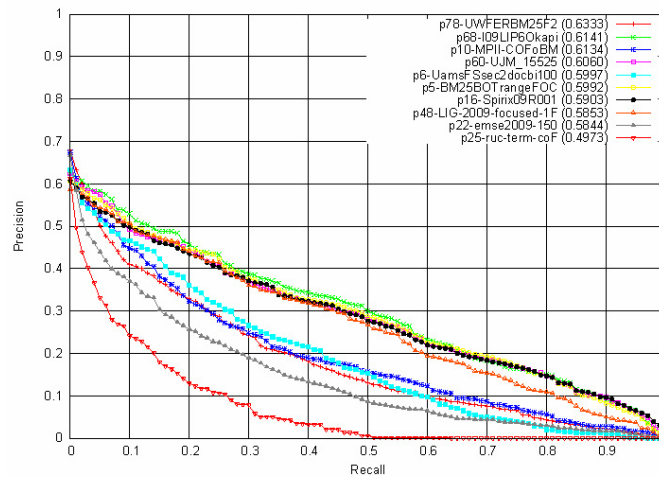


Fig. 2. iP[x] for top runs in Focused mode, Ad-hoc track.

5.2 Efficiency Track

Figure 5 depicts the iP[x] plots for all Focused type A runs submitted to the Efficiency track. The best TopX run (TopX2-09-ArHeu-Fo-150-Hot) is highlighted. Figures 6 and 7 depict the iP[0.01] plots of all Focused runs submitted to the Efficiency track in comparison to their runtime. Our fastest runs achieved an average runtime for type A topics of 72 ms per CO query, and 235 ms per CAS query at the Efficiency track, respectively (denoted as TopX2-09-ArHeu-Fo-15-Hot and TopX2-09-CASHeu-Fo-15-Hot). TopX ranks among the top engines, with a very good retrieval quality vs. runtime trade-off (compare also Tables 1 and 2). Our fastest runs operated over a hot cache and employed a heuristic top- k stopping condition (denoted as *Heu*, thus terminating query evaluations after the first block of elements was read and merged from each of the inverted lists that are related to the query (see [7])). Unsurprisingly, our best Article runs on type A topics (70 ms) were slightly faster than our best CO runs (72 ms), and about a factor of 3 faster than our CAS runs (235 ms), at a comparable result quality as in the Ad-hoc track. For type B topics, containing CO topics with partly more than 90 keywords, average runtimes were more than an order

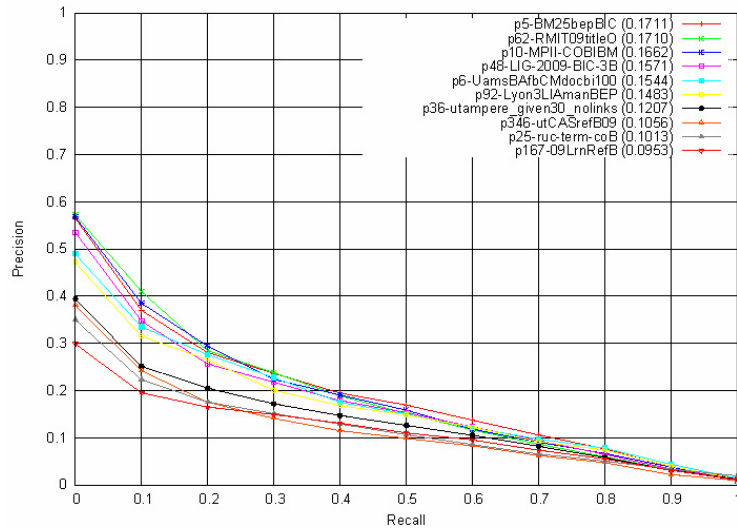


Fig. 3. $gP[x]$ for top runs in Best-In-Context mode, Ad-hoc track.

of magnitude worse than for type A topics. Altogether, TopX was the only engine to submit other than Article runs to the Efficiency track.

Part.ID	Run ID	Type	iP[0.00]	iP[0.01]	iP[0.05]	iP [0.10]	MAiP [ms]	avg total [ms]	k	Mode
10	TopX2-09-Ar-Fo-15-Hot	Article	0.598	0.583	0.494	0.397	0.178	84.0	15	CO
10	TopX2-09-ArHeu-Fo-1500-Hot	Article	0.597	0.586	0.530	0.475	0.275	301.2	1500	CO
10	TopX2-09-ArHeu-Fo-150-Hot	Article	0.598	0.588	0.531	0.474	0.252	87.2	150	CO
10	TopX2-09-ArHeu-Fo-15-Hot	Article	0.589	0.577	0.482	0.398	0.175	69.8	15	CO
10	TopX2-09-CAS-Fo-15-Cold	Focused	0.546	0.480	0.423	0.355	0.138	467.7	15	CAS
10	TopX2-09-CAS-Fo-15-Hot	Focused	0.545	0.493	0.418	0.350	0.137	379.2	15	CAS
10	TopX2-09-CASHeu-Fo-15-Hot	Focused	0.525	0.468	0.358	0.304	0.124	234.9	15	CAS
10	TopX2-09-COS-Fo-15-Hot	Focused	0.645	0.567	0.406	0.285	0.135	125.6	15	CO
10	TopX2-09-CO-Fo-15-Hot	Focused	0.641	0.564	0.405	0.291	0.130	338.2	15	CO
10	TopX2-09-COHeu-Fo-15-Hot	Focused	0.507	0.429	0.306	0.196	0.079	71.8	15	CO

Table 1. Summary of all TopX Efficiency runs, type A queries.

6 Conclusions

TopX was one of the few engines to consider CAS queries in the Ad-hoc track over the new 2009 Wikipedia collection, and even the only engine in the Efficiency track that processed CAS queries at all. In our ongoing work, we already started looking into further XPath Full-Text operations, including phrase matching and proximity-based ranking for both CO and CAS queries, as well as top- k support for more complex XQuery

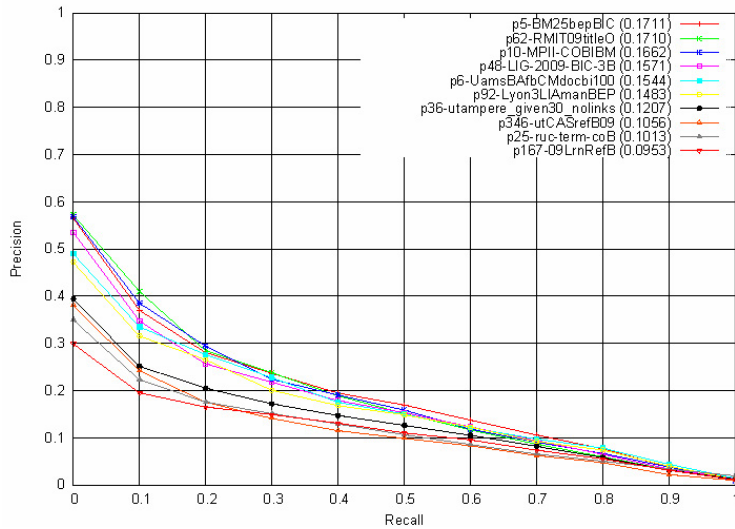


Fig. 4. $iP[x]$ for top runs in Thorough mode, Ad-hoc track.

Part.ID	Run ID	Type	$iP[0.00]$	$iP[0.01]$	$iP[0.05]$	$iP[0.10]$	MAiP [ms]	avg total [ms]	k	Mode
10	TopX2-09-Ar-TOP15-Hot	Article	0.440	0.427	0.362	0.315	0.119	4163.2	15	CO
10	TopX2-09-ArHeu-TOP1500-Hot	Article	0.443	0.434	0.381	0.358	0.206	2412.0	1500	CO
10	TopX2-09-ArHeu-TOP150-Hot	Article	0.443	0.433	0.381	0.358	0.193	2260.2	150	CO
10	TopX2-09-ArHeu-TOP15-Hot	Focused	0.431	0.418	0.344	0.303	0.118	2205.4	15	CO
10	TopX2-09-CAS-TOP15-Cold	Focused	0.442	0.428	0.363	0.316	0.119	4352.3	15	CAS
10	TopX2-09-CAS-TOP15-Hot	Focused	0.440	0.427	0.357	0.315	0.118	4685.1	15	CAS
10	TopX2-09-CASHeu-TOP15-Hot	Focused	0.431	0.418	0.344	0.303	0.118	2293.1	15	CAS
10	TopX2-09-COHeu-TOP15-Hot	Focused	0.364	0.329	0.221	0.175	0.066	497.8	15	CO

Table 2. Summary of all TopX Efficiency runs, type B queries.

constructs. Our long-term goal is to make TopX a full-fledged, open-source indexing and search platform for the W3C XPath 2.0 and XQuery 1.0 Full-Text standards. While we believe that our distributed indexing strategy already scales to future XML indexing needs (with many Terabytes potential index size), making also the search process truly distributed will be a challenging topic for future work.

References

1. C. L. A. Clarke. Controlling overlap in content-oriented XML retrieval. In R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, and J. Tait, editors, *SIGIR*, pages 314–321. ACM, 2005.
2. J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI 2004*, pages 137–150, 2004.

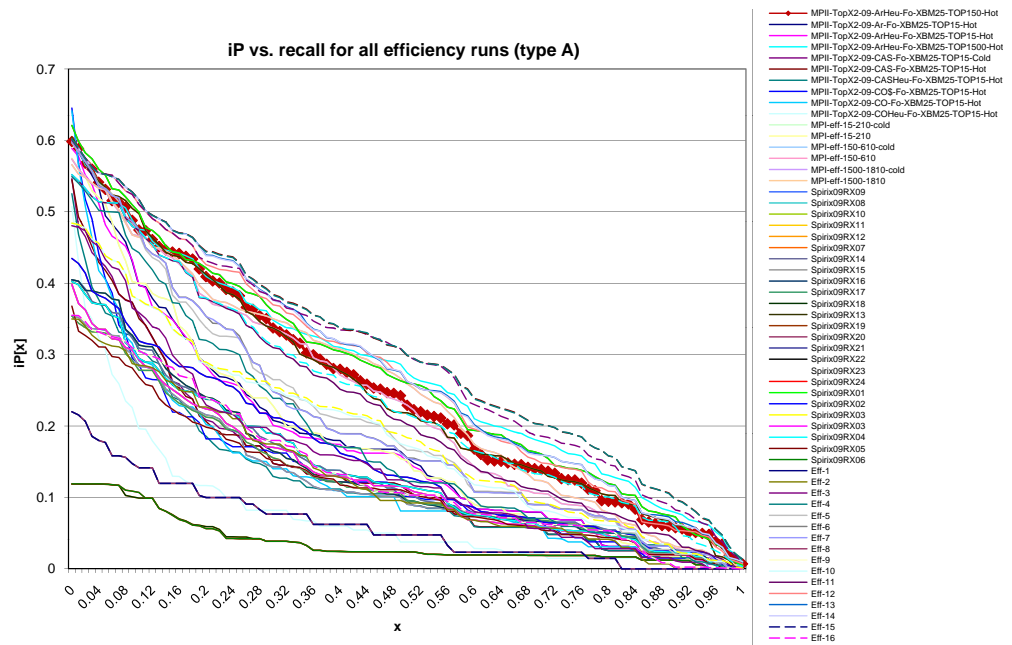


Fig. 5. $iP[x]$ for all Efficiency runs, type A queries.

3. J. Kamps, J. Pehcevski, G. Kazai, M. Lalmas, and S. Robertson. INEX 2007 evaluation measures. In N. Fuhr, J. Kamps, M. Lalmas, and A. Trotman, editors, *INEX*, volume 4862 of *Lecture Notes in Computer Science*, pages 24–33. Springer, 2007.
4. W. Lu, S. E. Robertson, and A. MacFarlane. Field-weighted xml retrieval based on bm25. In N. Fuhr, M. Lalmas, S. Malik, and G. Kazai, editors, *INEX*, volume 3977 of *Lecture Notes in Computer Science*, pages 161–171. Springer, 2005.
5. S. E. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford, and A. Payne. Okapi at TREC-4. In *TREC*, 1995.
6. S. E. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In D. Grossman, L. Gravano, C. Zhai, O. Herzog, and D. A. Evans, editors, *CIKM*, pages 42–49. ACM, 2004.
7. M. Theobald, M. AbuJarour, and R. Schenkel. TopX 2.0 at the INEX 2008 Efficiency Track. In S. Geva, J. Kamps, and A. Trotman, editors, *INEX*, volume 5631 of *Lecture Notes in Computer Science*, pages 224–236. Springer, 2008.
8. M. Theobald, H. Bast, D. Majumdar, R. Schenkel, and G. Weikum. TopX: efficient and versatile top- k query processing for semistructured data. *VLDB J.*, 17(1):81–115, 2008.
9. M. Theobald, R. Schenkel, and G. Weikum. An efficient and versatile query engine for TopX search. In K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, and B. C. Ooi, editors, *VLDB*, pages 625–636. ACM, 2005.
10. A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *INEX*, volume 3493 of *Lecture Notes in Computer Science*, pages 16–40. Springer, 2004.

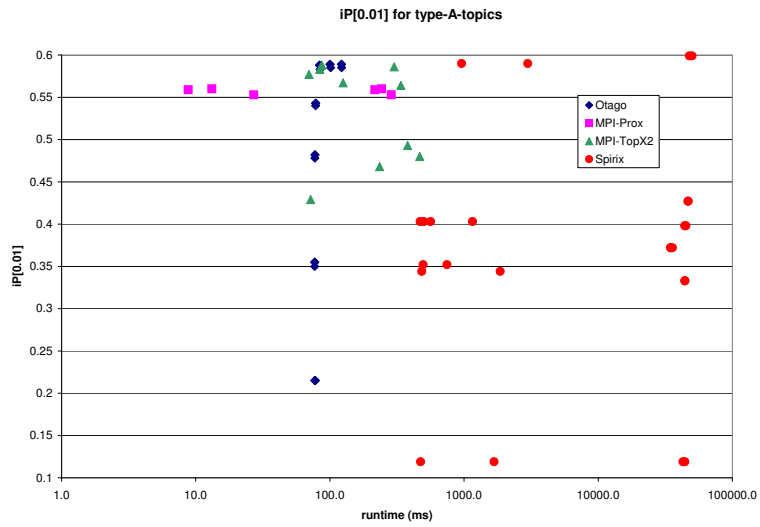


Fig. 6. Runtime vs. iP[0.01] for all Efficiency runs, type A queries.

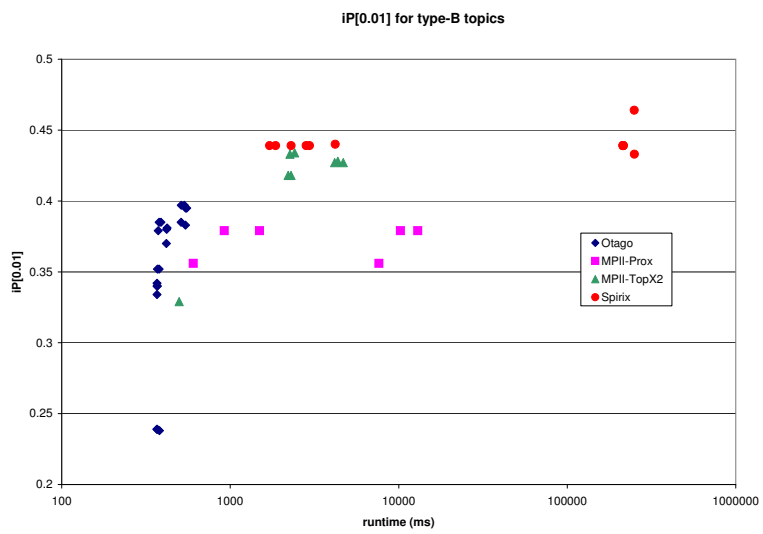


Fig. 7. Runtime vs. iP[0.01] for all Efficiency runs, type B queries.

Fast and Effective Focused Retrieval

Andrew Trotman¹, Xiang-Fei Jia¹ and Shlomo Geva²

¹Computer Science, University of Otago, Dunedin, New Zealand

²Queensland University of Technology, Brisbane, Australia

Abstract. Building an efficient and an effective search engine is a very challenging task. In this paper, we present the efficiency and effectiveness of our search engine at the INEX 2009 Efficiency and Ad Hoc Tracks. We have developed a simple and effective pruning method for fast query evaluation, and used a two-step process for Ad Hoc retrieval. The overall results from both Tracks show that our search engine performs very competitively in terms of both efficiency and effectiveness.

1 Introduction

There are two main performance issues in Information Retrieval (IR); effectiveness and efficiency. In the past, the research was mainly focused on effectiveness. Only until recent years, efficiency is getting more research focus under the trend of larger document collection size. In this paper, we present our approaches towards efficient and effective IR and show our submitted results at the INEX 2009 efficiency and Ad Hoc Tracks. We have developed a simple and effective pruning method for fast query evaluation, and used a two-step process for Ad Hoc retrieval. The overall results from both Tracks show that our search engine performs very competitively in terms of both efficiency and effectiveness.

In section 2, IR efficiency issues are discussed. Section 3 explains how we achieve fast indexing and searching for large document collections. Experiments and results are shown in Section 4 and 5. Section 6 discusses our runs in the Ad Hoc Track. The last section provides the conclusion and future work.

2 Background

Inverted files [1,2] are the most widely used index structures in IR. The index has two parts: a dictionary of unique terms extracted from a document collection and a list of postings (a pair of <document number, term frequency>) for each of the dictionary terms.

When considering efficiency issues, IR search engines are very interesting because search engines are neither purely I/O-intensive nor solely CPU-intensive. To serve a query, I/O is needed in order to read dictionary terms as well as postings lists from disk. Then postings lists are processed using a ranking function and intermediate results are stored in accumulators. At the end, the accumulators are sorted and the top results are returned. There are two obvious questions;

(1) How do we reduce the I/O required for reading dictionary terms and posting lists, and (2) how do we minimise the processing and sorting.

When considering effectiveness of Focused Retrieval, it is necessary to consider whether to index documents, elements or passages. This leads to the question of how effectiveness is effected by these index types — we have experimented using document index and post processing to focus.

2.1 Disk I/O

The dictionary has a small size and can be loaded into memory at startup. Due to their large size, postings must be compressed and stored on disk. Various compression algorithms have been developed, including Variable Byte, Elias gamma, Elias delta, Golomb and Binary Interpolative. Trotman [3] concludes that Variable Byte coding provides the best balance between the compression ratio and the CPU cost for decompression. Anh & Moffat [4,5] construct word-aligned binary codes, which are effective at compression and fast at decompression. We are experimenting with these compression algorithms.

Caching can also be used to reduce disk I/O. There are two levels of caching; system-level and application-level. At the system-level, operating systems provides general purpose I/O caching algorithms. For example, the Linux kernel provides several I/O caching algorithms [6]. At the application-level, caching is more effective since the application can deploy specialised caching algorithms [7]. We are experimenting with caching approaches.

For IR search engines, there are two ways of caching at the application-level. The first solution is to cache query results, which not only reduces disk I/O but also avoids re-evaluation of queries. However, queries tend to have low frequency of repetition [8]. The second is to cache raw postings lists. The challenge is to implement a efficient replacement algorithm in order to keep the postings in memory. We are also experimenting with caching algorithms.

Since the advent of 64-bit machine with vast amount of memory, is has become feasible to load both the dictionary and the compressed postings of a whole-document inverted file into main memory, thus eliminating all disk I/O. For Focused Retrieval a post process of the documents can be a second step. If the documents also fit into memory, then no I/O is needed for Focused Retrieval. This is the approach we are taking, however our experiments in this paper were performed without caching.

2.2 Query Pruning

The processing of postings and subsequent sorting of the accumulators can be computationally expensive, especially when queries contain frequent terms. Frequent terms appear in many documents in a collection and have low similarity scores due to having a low Inverse Document Frequency (IDF). Processing the postings for these terms not only takes time, but also has little impact on the final ranking results.

The purpose of query pruning is to eliminating any unnecessary evaluation while still maintaining good precision. Query pruning requires that (1) every term is assigned a weight [9,10], (2) query terms are sorted in decreasing order of their weights (such as IDF), (3) the postings are sorted in decreasing order of their weights (such as TF). Partial similarity scores are obtained when some stop condition is met. Query terms might be pruned or postings might be pruning.

Harman & Candeka [11] experimented with a static pruning algorithm in which complete similarity scores are calculated by processing all query terms and postings of the terms. But only a limited number of accumulators, those above a given threshold, are sorted and returned. A dynamic pruning algorithm developed by Buckley and Lewit [12] keeps track of the top $k+1$ partial similarity scores in the set of accumulators, and stops the query evaluation when it is impossible to alter the top-k documents.

Moffat & Zobel [13] developed two pruning algorithms; the *quit* algorithm is similar to the top-k algorithm and stops processing query terms when a non-zero number of accumulators exceeds a constant value. While the *continue* algorithm continues to process query terms when the stopping condition is met, but only updates documents already in the set of accumulators.

Persin et al. [14,15] argue that a single stopping condition is not efficient enough to maintain fair partial similarity scores. They introduced both a global and a local threshold. The global threshold determines if a new document should be inserted into the set of accumulators, while the local threshold checks if existing accumulators should be updated. The global threshold is similar to the *quit* algorithm, while the combination of the global and local thresholds is like the *continue* algorithm. However, there are two differences; (1) the quit algorithm keeps adding new documents into the set of accumulators until reaching a stopping condition, while the global threshold algorithm adds a new document into the set of accumulators only if the partial similarity score of the document is above the predefined global threshold. (2) The local threshold algorithm only updates the set of accumulators when a partial similarity score is above the local threshold, while the continue algorithm has no condition to update the accumulators.

Anh et al. [16] introduced impact ordering, in which the postings for a term are ordered according to their overall contribution to the similarity scores. They state that Persin et al. [14,15] defined term-weighting as a form of TF-IDF (the global threshold is the IDF and the local threshold is the TF), while Anh et al. used normalised TF-IDF. The term impact is defined as $w_{d,t}/W_d$ where $w_{d,t}$ is the document term weight and W_d is the length of the document vector.

3 Efficiency

3.1 Indexer

Memory management is a challenge for fast indexing. Efficient management of memory can substantially reduce indexing time. Our search engine has a memory

management layer above the operating system. The layer pre-allocates large chunks of memory. When the search engine requires memory, the requests are served from the pre-allocated pool, instead of calling system memory allocation functions. The sacrifice is that some portion of pre-allocated memory might be wasted. The memory layer is used both in indexing and in query evaluation. As we show in our results, only a very small portion of memory is actually wasted.

The indexer uses hashing with a collision binary tree for maintaining terms. We tried several hashing functions including Hsieh's super fast hashing function. By default, the indexer uses a very simple hashing function, which only hashes the first four characters of a term and its length by referencing a pre-defined look-up table. A simple hashing function has less computational cost, but causes more collisions. Collisions are handled by a simple unbalanced binary tree. We will examine the advantages of various hashing and chaining algorithms in future work.

Postings lists can vary substantially in length. The indexer uses various sizes of memory blocks chained together. The initial block size is 8 bytes and the resize factor is 1.5 for the subsequent blocks.

In order to reduce the size of the inverted file, we always use 1 to store term frequencies. This limits term frequencies to a maximum of 255. Truncating term frequencies could have a impact on long documents. But we assume long documents are rare in a collection and terms with high frequencies in a document are more likely to be common words.

As shown in Figure 1, the index file has four levels of structure. Instead of using the pair of <document number, term frequency> for postings, we group documents with the same term frequency together and store the term frequency at the beginning of each group. We impact order on term frequency. The difference of document ids in each group are then stored in increasing order and each group ends with a zero. The postings are sorted in descending order of term frequency. This format of postings allows pruning during query evaluations. Postings are compressed with Variable Byte coding.

The dictionary of terms is split into two parts. The first-level stores the first four bytes of a term string, the length of the term string and the position to locate the second-level structure. Terms with the same prefix (the first four bytes) are stored in a term block in the second-level. The term block stores the statistics for the terms, including collection frequency, document frequency, offset to locate the postings list, the length of the postings list stored on disk, the uncompressed length of the postings list, and the position to locate the term suffix which is stored at the end of the term block.

At the very end of the index file, the small footer stores the location of the first level dictionary and other values for the management of the index.

3.2 Query Evaluation

At start-up, only the the first-level dictionary is loaded into memory. To process a query term, two disk reads have to be issued; The first reads the second-level dictionary. Then the offset in that structure is used to locate postings, since we

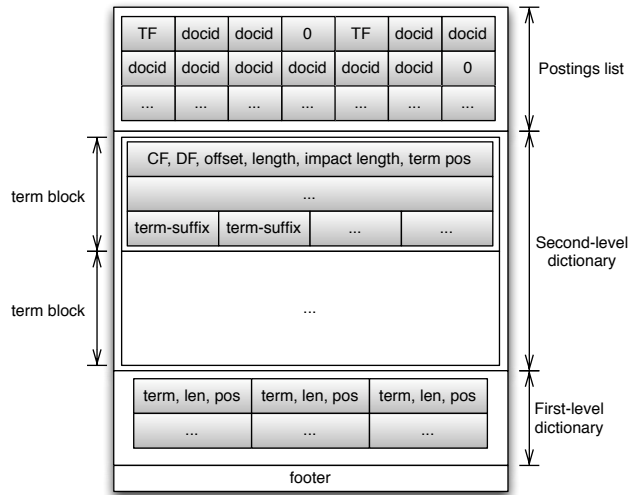


Fig. 1. The index structures

do not use caching in these experiments. The current implementation has no disk I/O caching. We simply deploy the general purpose caching provided by the underlying operating system.

An array is used to store the accumulators. We used fixed point arithmetic on the accumulators because it is faster than the floating point.

We have implemented a special version of quick sort algorithm [17] for fast sorting of the accumulators. One of the features of the algorithm is partial sorting; It will return the top-k documents by partitioning and then only sorting the top partition. Pruning accumulators using partial sorting is similar to that of Harman & Candeka [11]. A command line option (lower-k) to our search engine is used to specify how many top documents to return.

We have also developed a method for static pruning of postings. A command line option (upper-K) is used to specify a value, which is the number of document ids (in the postings list of a term) to be processed. The upper-K value is only a hint. The search engine will always finish processing all postings with the same TF at the K^{th} postings. The combined use of both the lower-k and upper-K methods is similar to the *continue* algorithm.

When upper-K is specified at the command line, the whole postings list of a term is decompressed, even though only partial postings will be processed (this is left for future work). Moffat and Anh [13,18] have developed methods for partial decompression for Variable Byte compressed lists. However, these methods do not come without a cost; Extra housekeeping data must be inserted into the postings lists, thus increasing the size of the index. Further more, there is also a computational cost in keeping track of the housekeeping data.

A modified BM25 is used for ranking. This variant does not result in negative IDF values and is defined thus:

$$RSV_d = \sum_{t \in q} \log \left(\frac{N}{df_t} \right) \cdot \frac{(k_1 + 1) tf_{td}}{k_1 \left((1 - b) + b \times \left(\frac{L_d}{L_{avg}} \right) \right) + tf_{td}}$$

Here, N is the total number of documents, and df_t and tf_{td} are the number of documents containing the term t and the frequency of the term in document d , and L_d and L_{avg} are the length of document d and the average length of all documents. The empirical parameters k_1 and b have been set to 0.9 and 0.4 respectively by training on the previous INEX Wikipedia collection.

4 Experiments

We conducted our experiments on a system with dual quad-core Intel Xeon E5410 2.3 GHz, DDR2 PC5300 7 GB main memory, Seagate 7200 RPM 500 GB hard drive, and running Linux with kernel version 2.6.30.

The collection used in the INEX 2009 Efficiency Track is the INEX 2009 Wikipedia collection [19]. The collection was indexed using the default parameters as discussed in Section 3. No words were stopped and stemming was not used. The indexing took about 1 hour and 16 minute. The memory layer allocated a total memory of 5.3 GB with a utilisation of 97%. Only 160 MB of memory was allocated but never used. Table 1 show s a summary of the document collection.

The INEX 2009 Efficiency Track used two types of topics, with both types having 115 queries. Type A Topics are short queries and the same as the INEX 2009 Ad Hoc topics. Type B Topics are expansions of topics in Type A and intended as long queries. Both topics allow *focused*, *thorough* and *article* query evaluations. Our search engine does not natively support focused retrieval yet, but we instead use a post-process. We only evaluated the topics for *article* Content-Only. We used the BM25 ranking model as discussed in previous section. The k_1 and b values were 0.9 and 0.4 respectively.

We experimented only sorting the top-k documents using the lower-k parameter with $k = 15, 150$ and 1500 as required by the Efficiency Track. Query terms do not have to be sorted in descending order of term frequency since our pruning method does not prune query terms. We also experimented pruning of postings using the upper-K parameter. For each iteration of the lower-k, we specified the upper-K of 1, 15, 150, 1500, 15000, 150000, 1500000. In total we submitted 21 runs.

The disk cache was flushed before each run. No caching mechanism was deployed except that provided by the Linux operating system.

5 Results

Table 2 shows a summary of the runs evaluated on the Type A topics. The first column shows the run-id. The interpolated Precision (iP) reflects the evaluations

Collection Size	50.7 GB
Documents	2666190
Average Document Length	881 words
Unique Words	11393924
Total Worlds	2348343176
Postings Size	1.2 GB
Dictionary Size	369 MB

Table 1. Summary of INEX 2009 Wikipedia Collection

of top-k documents at points of 0%, 1%, 5% and 10%. The overall performance is shown as Mean Average interpolated Precision (MAiP). The average run time, consisting of the CPU and I/O, is the total time taken for the runs. The last two columns show the lower-k and upper-K parameters. In terms of MAiP, the best runs are Eff-21, Eff-20 and Eff-19 with a value of 0.3, 0.3 and 0.29 respectively.

Figure 2(a) shows the Precision-Recall graph of our 21 runs for Type A topics. Except the Eff-1, Eff-8 and Eff-15 runs, all other runs achieved a very good early precision. Bad performance of the three runs was caused by pruning too many postings (a too small value for upper-K) regardless the number of top-k documents retrieved.

The relationship between the MAiP measures and the lower-k and upper-K parameters is plotted in Figure 3(a) using data from Table 2. When upper-K has values of 150 and 1500, MAiP measures are much better than the upper-K 15. In terms of lower-k, MAiP measures approach constant at a value of 15000.

To have a better picture of the total time cost, we plotted the time costs of all runs in Figure 4(a) using data from Table 2. Regardless of the values used for lower-k and upper-K, the same number of postings were retrieved from disk, thus causing all runs to have the same amount of disk I/O. The figure also shows that the CPU usage is high when upper-K has a value greater than 1500.

We used the same measures for Type B topics. Table 3 shows the summary of averaged measures for Type B topics. The best runs are Eff-20, Eff-21, Eff-13 with an MAiP measure of 0.18, 0.17 and 0.17 respectively. An interesting observation is that the best run (Eff-20) does not has the highest upper-k value. Processing fewer postings not only saves time, but also improves precision. The best MAiP in Type A is 0.3 while only 0.18 in Type B. We are investigating why.

Figure 2(b) shows the Precision-Recall graph for Type B topics. The Eff-1, Eff-8 and Eff-15 runs also achieved low precision at the early stage. All other runs received good early precision. Figure 3(b) shows the MAiP measures using various lower-k and upper-K values. It shows a similar pattern to that of Figure 3(a). However, good performance is seen when upper-K has a value of 150, rather than the 15000 for Type A topics.

The time cost for Type B queries is plotted in Figure 4(b). All runs used the same amount of time for I/O, and have different CPU cost due to various values used for lower-k and upper-K parameter. The lower-k again has no effect on the

CPU cost, and values of 1500 or above for upper-K causes more CPU usage. It took a much longer time for I/O, due to more terms, when compared with I/O cost in Type A.

As shown in both Figure 4(a) and 4(b), the runtime is dominated by the I/O. This leads us to consider that storing the whole index in memory is important.

The submitted runs used small values for the lower-k parameter. In order to see the impact of lower-k, we evaluated both topic sets using only lower-k with a value of 1, 15, 150, 1500, 15000, 150000, 1500000 and 2666190. The number 2666190 is the total number of documents in the collection. As shown in Figure 5, The time taken for sorting the accumulators increases when lower-k has a value above 15000. The sorting times increase from 13.72 ms and 25.57 for Type A and B topics (when lower-k is 15000) to 50.81 ms and 159.39 ms (when lower-k is 2666190) respectively.

run-id	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.01]	MAiP	Total time	CPU	I/O	Lower-k	Upper-K
Eff-01	0.22	0.22	0.18	0.14	0.06	77.8	20.4	57.3	15	1
Eff-02	0.35	0.35	0.32	0.29	0.13	77.1	20.4	56.7	15	15
Eff-03	0.48	0.48	0.43	0.36	0.15	77.2	20.0	57.2	15	150
Eff-04	0.55	0.54	0.5	0.44	0.18	78.1	21.0	57.1	15	1500
Eff-05	0.6	0.58	0.53	0.47	0.21	84.5	26.9	57.6	15	15000
Eff-06	0.6	0.59	0.53	0.48	0.21	101.3	43.0	58.2	15	150000
Eff-07	0.6	0.59	0.53	0.48	0.2	122.2	64.9	57.3	15	1500000
Eff-08	0.22	0.22	0.18	0.14	0.06	77.7	20.4	57.3	150	1
Eff-09	0.36	0.36	0.33	0.31	0.14	76.9	19.9	57.0	150	15
Eff-10	0.48	0.48	0.44	0.38	0.19	77.4	20.3	57.2	150	150
Eff-11	0.55	0.54	0.51	0.47	0.23	78.3	21.3	57.0	150	1500
Eff-12	0.6	0.59	0.55	0.51	0.27	83.8	26.9	56.9	150	15000
Eff-13	0.6	0.59	0.55	0.52	0.28	100.0	42.7	57.3	150	150000
Eff-14	0.6	0.59	0.55	0.52	0.28	122.2	64.9	57.3	150	1500000
Eff-15	0.22	0.22	0.18	0.14	0.06	76.9	20.3	56.6	1500	1
Eff-16	0.36	0.36	0.33	0.31	0.14	77.1	20.2	56.9	1500	15
Eff-17	0.48	0.48	0.44	0.38	0.19	77.4	20.1	57.3	1500	150
Eff-18	0.55	0.54	0.51	0.47	0.24	78.5	20.9	57.6	1500	1500
Eff-19	0.6	0.59	0.55	0.51	0.29	83.6	26.8	56.9	1500	15000
Eff-20	0.6	0.59	0.55	0.52	0.3	100.3	42.7	57.6	1500	150000
Eff-21	0.6	0.59	0.55	0.52	0.3	121.7	64.3	57.4	1500	1500000

Table 2. A summary of the runs for Type A topics

6 Ad Hoc

We also used our search engine in the ad hoc track. The whole-document results were extracted and submitted as the REFERENCE run. We then took the reference run and ran a post-process to focus the top 50 results. Our rationale

run-id	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP	Total time	CPU	I/O	Lower-k	Upper-K
Eff-01	0.24	0.24	0.17	0.14	0.05	380.22	31.97	348.25	15	1
Eff-02	0.34	0.33	0.32	0.29	0.1	367.53	32.07	335.46	15	15
Eff-03	0.35	0.34	0.33	0.29	0.12	367.44	33.41	334.03	15	150
Eff-04	0.38	0.38	0.34	0.32	0.12	373.95	41.72	332.23	15	1500
Eff-05	0.38	0.37	0.33	0.31	0.11	418.02	89.73	328.29	15	15000
Eff-06	0.39	0.39	0.34	0.3	0.11	511.56	184.9	326.66	15	150000
Eff-07	0.39	0.38	0.33	0.3	0.11	542.98	216.97	326.02	15	1500000
Eff-08	0.24	0.24	0.18	0.15	0.05	367.21	32.08	335.13	150	1
Eff-09	0.34	0.34	0.33	0.3	0.13	367.51	32.14	335.37	150	15
Eff-10	0.36	0.35	0.34	0.32	0.16	370.12	33.43	336.69	150	150
Eff-11	0.39	0.39	0.35	0.34	0.16	387.61	41.98	345.63	150	1500
Eff-12	0.39	0.38	0.35	0.34	0.16	419.43	90.03	329.39	150	15000
Eff-13	0.4	0.4	0.36	0.33	0.17	512.54	185.07	327.47	150	150000
Eff-14	0.4	0.4	0.36	0.33	0.16	543.53	216.59	326.94	150	1500000
Eff-15	0.24	0.24	0.18	0.15	0.05	368.33	31.84	336.49	1500	1
Eff-16	0.34	0.34	0.33	0.3	0.14	369.46	32.33	337.13	1500	15
Eff-17	0.36	0.35	0.34	0.32	0.17	378.73	33.23	345.5	1500	150
Eff-18	0.39	0.39	0.35	0.34	0.17	378.19	41.77	336.42	1500	1500
Eff-19	0.39	0.38	0.35	0.34	0.17	421.83	90.11	331.72	1500	15000
Eff-20	0.4	0.4	0.36	0.33	0.18	533.32	184.88	348.44	1500	150000
Eff-21	0.4	0.4	0.36	0.33	0.17	551.8	217.52	334.28	1500	1500000

Table 3. A summary of the runs for Type B topics

is that document retrieval should rank document from mostly about a topic is mostly not about a topic. If this is the case then focusing should be a fast and relatively simple post process.

6.1 Query Evaluation

Three sets of runs were submitted: Those starting BM25 were based on the reference run and generated from the topic title field (CO) using the BM25 search engine. Those starting ANTbigram were an experiment into phrase searching (and are not discussed here).

For structure searching (CO+S/CAS) we indexed all those tags in a document as special terms. If the path /A/B/C were present in the document then we indexed the document as containing tags A, B, and C. Searching for these tags did not take into consideration the path, only the presence of the tag; that is, /C/B/A would match /A/B/C. Ranking of tags was done with BM25 because the special terms were treated as ordinary terms during ranking. We call this technique Bag-Of-Tags.

Runs containing BOT in their name were generated from the CAS title using the Bag-Of-Tags approach. All search terms and tag names from the paths were included and the queries were ranked using BM25.

The post processing step was not done by the search engine — we leave that for future work. Several different techniques were used:

1. No Focusing (ARTICLE)
2. Deepest enclosing ancestor of all search terms (ANCESTOR)
3. Enclosing element range between first and last occurrence of a search term (RANGE/BEP)
4. All non-overlapping elements containing a search terms
5. All overlapping elements containing a search term (THOROUGH)

Runs were submitted to the BIC task (1 & 2), RIC (1-4), Focused (1-4) and thorough (1-5) tasks.

6.2 Results

In the BIC task our run BM25bepBIC placed first. It used BM25 to rank documents and then placed the Best Entry Point at the start of the first element that contained the first occurrence of any search term. Our second best run placed third (RMIT placed second) and it used the ancestor approach.

In the RIC task our runs placed first through to ninth. Our best run was BM25RangeRIC which simply trimmed all those elements from the start and end of the document that did not contain any occurrences of the search terms. The next most effective run was BM25AncestorRIC which chose the lowest common ancestor of the range (and consequently more non-relevant material). Of note, the REFERENCE run placed third – that is, whole document retrieval was very effective.

In the Focused task all our Bag-Of-Tags (CAS) runs placed better than our CO runs. Our best run placed ninth and used ranges, the ancestor run placed tenth and the article run placed eleventh.

In the thorough task our best run, BM25thorough, placed sixth with the Bag-of-Tags placing seventh. We have not concentrated on the focused track.

7 Conclusion and Future Work

In this paper, we introduced our search engine, discussed our design and implementation. We also demonstrated the initial evaluation on the INEX 2009 Efficiency and Ad Hoc Tracks. Our best runs for Type A topics have MAiP measure of 0.3 and runtime of 100 milliseconds, MAiP measure of 0.18 and runtime of 533 milliseconds for Type B topics. Compared with the overall results from the Efficiency Track, we believe that our results are very competitive.

Our ad hoc experiments have shown that the approach of finding relevant documents and then post-processing is an effective way of building a Focused Retrieval search engine for the in-Context tasks (where we placed first). They also show that ignoring the structural hints present in a query is reasonable.

Our Focused and Thorough results were not as good as our in-Context runs (we placed respectively fifth and third institutionally). Our experiments here

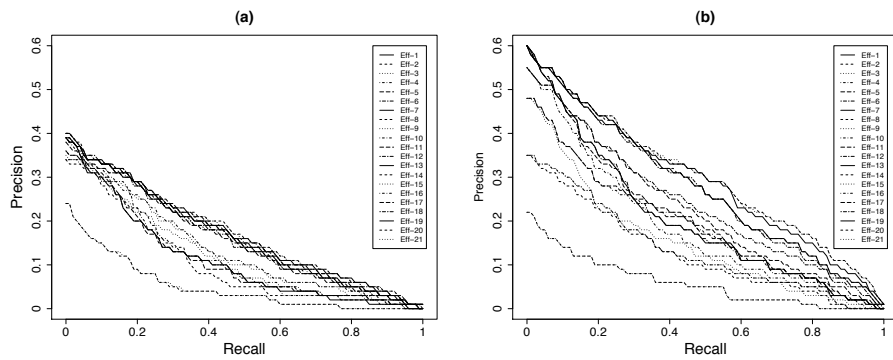


Fig. 2. Precision-Recall plot for (a) Type A and (b) Type B topics

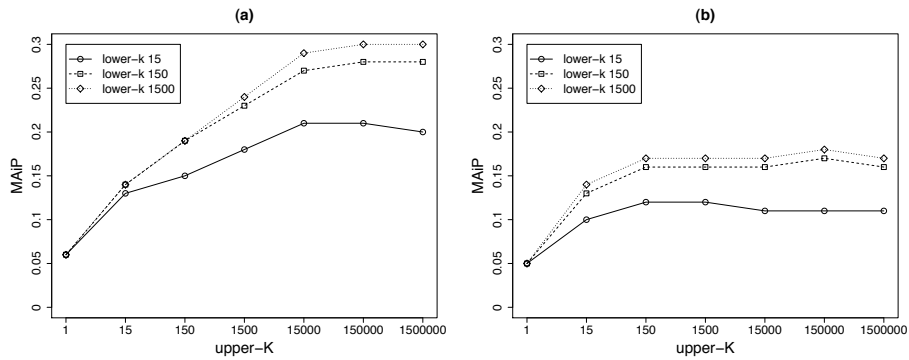


Fig. 3. MAiP measures for (a) Type A and (b) Type B topics

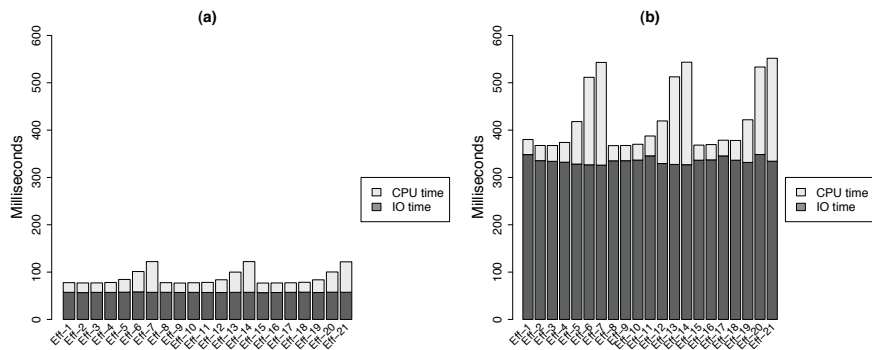


Fig. 4. Total runtime for (a) Type A and Type (b) topics

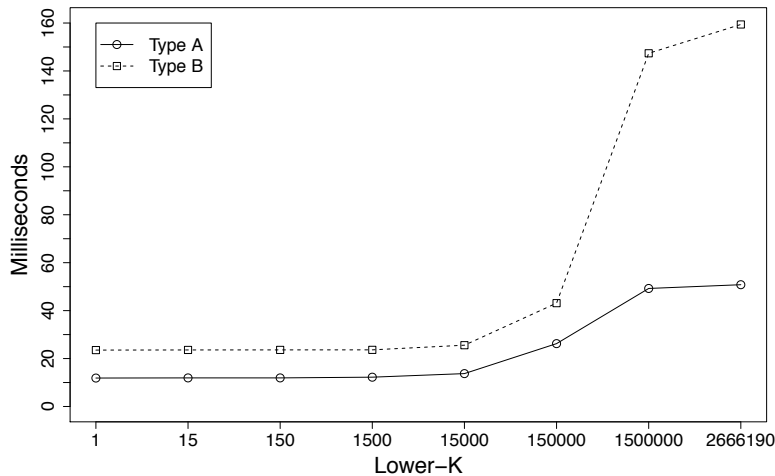


Fig. 5. Times taken for sorting accumulators

suggest that the Bag-of-Tags approach is effective with our BOT runs performing better than ignoring structural hints in the Focused task and comparably to ignoring the hints in the Focused task. In the Focused task we found that ranges are more effective than common ancestor and (because they are better excluders of non-relevant material). In future work we will be concentrating on increasing our performance in these two tasks.

Of particular interest to us, our runs did not perform best when measured as whole-document retrieval. Our focusing were, however, effective. LIG, RMIT University, and University of Amsterdam bettered our REFERENCE run and we are particularly interested in their approaches and how they might be applied to whole document retrieval (so that we may better our own runs).

In the future, we will continue to work on pruning for more efficient query evaluation. We are also interested in other techniques for improving efficiency without loss of effectiveness, including compression, caching and multi-threading on multi-core architectures.

References

1. Zobel, J., Moffat, A., Ramamohanarao, K.: Inverted files versus signature files for text indexing. *ACM Trans. Database Syst.* **23**(4) (1998) 453–490
2. Zobel, J., Moffat, A.: Inverted files for text search engines. *ACM Comput. Surv.* **38**(2) (2006) 6
3. Trotman, A.: Compressing inverted files. *Inf. Retr.* **6**(1) (2003) 5–19
4. Anh, V.N., Moffat, A.: Inverted index compression using word-aligned binary codes. *Inf. Retr.* **8**(1) (2005) 151–166

5. Anh, V.N., Moffat, A.: Improved word-aligned binary compression for text indexing. *IEEE Transactions on Knowledge and Data Engineering* **18**(6) (2006) 857–861
6. Bovet, D.P., Cesati, M.: *Understanding the linux kernel*, 3rd edition. (November 2005)
7. Jia, X., Trotman, A., O’Keefe, R., Huang, Z.: Application-specific disk I/O optimisation for a search engine. In: *PDCAT ’08: Proceedings of the 2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, Washington, DC, USA, IEEE Computer Society (2008) 399–404
8. Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Plachouras, V., Silvestri, F.: The impact of caching on search engines. In: *SIGIR ’07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, ACM (2007) 183–190
9. Salton, G., Buckley, C.: *Term-weighting approaches in automatic text retrieval*. (1988) 513–523
10. Lee, D.L., Chuang, H., Seamons, K.: Document ranking and the vector-space model. *IEEE Softw.* **14**(2) (1997) 67–75
11. Harman, D., Candela, G.: Retrieving records from a gigabyte of text on a minicomputer using statistical ranking. *Journal of the American Society for Information Science* **41** (1990) 581–589
12. Buckley, C., Lewit, A.F.: Optimization of inverted vector searches. (1985) 97–110
13. Moffat, A., Zobel, J.: Self-indexing inverted files for fast text retrieval. *ACM Trans. Inf. Syst.* **14**(4) (1996) 349–379
14. Persin, M.: Document filtering for fast ranking. (1994) 339–348
15. Persin, M., Zobel, J., Sacks-Davis, R.: Filtered document retrieval with frequency-sorted indexes. *J. Am. Soc. Inf. Sci.* **47**(10) (1996) 749–764
16. Anh, V.N., de Kretser, O., Moffat, A.: Vector-space ranking with effective early termination. (2001) 35–42
17. Bentley, J.L., Mcilroy, M.D.: *Engineering a sort function* (1993)
18. Anh, V.N., Moffat, A.: Compressed inverted files with reduced decoding overheads. (1998) 290–297
19. Schenkel, R., Suchanek, F., Kasneci, G.: *YAWN: A semantically annotated wikipedia xml corpus*. (March 2007)

Experiments regarding the Efficiency Track and the Ad Hoc Track at INEX'09: Searching in Large-Scale Collections

Judith Winter and Gerold Kühne

University of Applied Science, Business Information Systems Group, Frankfurt, Germany
winter@fb3.fh-frankfurt.de, gkuehne@web.de

Abstract. This year, the test collection provided by INEX for evaluating results achieved with XML-Retrieval solutions has grown remarkably. The new Wikipedia collection consists of more than 50 GB (not including the images contained in articles), more than 2.660.000 articles and a large supply of semantically rich tags used for markup. On the one hand, searching such a big collection requires the use of much more resources such as processing power, RAM, and index space, in comparison to the small collections that were searched at INEX in previous years. Thus, it is more important than ever to regard efficiency issues when performing XML-Retrieval tasks on such a big collection. On the other hand, the rich markup of the new collection is a big opportunity to exploit this structure, such that a gain of efficiency while searching can be achieved. This paper describes our experiments submitted for the Efficiency Track and the Ad Hoc Track of INEX 2009 and conducts a first interpretation of the results that were recently released. For the Ad Hoc Track, we submitted runs with different ranking functions. For the submitted runs of the Efficiency Track, different functions to compare structural similarity of CAS-queries were applied. The scenario of our experiments is a distributed one: collection, index, and search load are split over a peer-to-peer network in order to gain more efficiency in terms of load balancing. We also analyzed, how the exploitation of the new given rich structure can be used to achieve more efficiency in regard to network traffic while routing in this distributed setting. In spite of the fact that we mainly aimed at efficiency, the official evaluation of our experiments resulted in a high precision in comparison to the other participants. We made it into the top-10 systems at the Ad Hoc Track (Rank 7 at Focused Task). At the Efficiency Track, we gained rank 1 in terms of precision in both categories of topics (type A and type B).

Keywords: XML Retrieval, Large-scale Collection, Distributed Search, INEX

1 Introduction and Motivation

1.1 Motivation

For years, it has been discussed at INEX that we need a test collection that provides tags with more semantic meaning than in the Wikipedia collection used at INEX from 2006 to 2008. This year, we can finally make use of rich markup that includes a wide variety of tags with real semantic meaning, other than the pure structural tags such as paragraph and section that were used before. The new collection has also grown from 4.6 GB to more than 50 GB. Building an index on such a large-scale collection and searching it now requires much more resources such as computing power, memory and disk space for indexing and retrieval. Splitting the resource consumption over a set of computers can distribute the search load and thus gain more efficiency of the search system.

We have developed a distributed search engine for XML-Retrieval that is based on a peer-to-peer (P2P) system. That is, we use a set of autonomous and equal computers (= peers) that are pooled together to share resources in a self-organized manner without using a central control. Due to this self-organization of the system, it provides the potential to realize fault-tolerance and robustness. It may scale to theoretically unlimited numbers of participating nodes, and can bring together otherwise unused resources. Owners of objects such as documents often use P2P networks to share their collections with others without giving up full control over their documents. Many users do not want to store their collections on central servers where the documents might be object to censorship or control by an authority. They prefer to store them locally on their own machines such that it is the owner's sole decision, who can access which documents when. Not only the ownership argument but also privacy and security issues are reasons for people not wanting to store their information in central systems. P2P networks are emerging infrastructures for distributed computing, real-time communication, ad-hoc collaboration, and information sharing in large-scale distributed systems. Applications include E-Business, E-Commerce, E-Science, and Digital Libraries, and are used among large, diverse, and dynamic sets of users. In this paper, we concentrate on the technical advantages of P2P systems: their potential to distribute the search load over the participating peers.

Therefore, we distribute both the Wikipedia collection and the index built on it over a peer-to-peer network. On this basis, we have performed experiments that exploit the newly rich structure of the collection in order to achieve more effectiveness while ranking (Ad Hoc Track) and to achieve more efficiency while routing CAS-queries in the P2P network (Efficiency Track). Our results resulted in a high precision, e.g. we made it onto rank 7 of the Ad Hoc Track (Focused Task) and ranked highest in both Efficiency Track Tasks (type A and type B topics).

1.2 Structural Similarity Functions used for Ranking and Routing

To compare the structural similarity of the hints given by users in CAS-queries and the structure extracted of the indexed articles, we applied different structural

similarity functions. In general, there are four groups of functions that can be used and that differ in the thoroughness they achieve in analyzing the similarity: *perfect match*, *partial match*, *fuzzy match*, and *baseline (flat)*.

We developed the following formula as a representative of the *partial match* type of strategies, i.e. where one of the compared structures has to be a sub-sequence of the other and the overlapping ratio is measured.

$$Sim_1(s_i, s_u) = \begin{cases} \left(\frac{1 + |s_i|}{1 + |s_u|} \right)^\alpha, & \text{if } s_i \text{ is sub-sequence of } s_u \\ \beta \cdot Sim_1(s_u, s_i), & \text{if } s_u \text{ is sub-sequence of } s_i \\ 0, & \text{else} \end{cases} \quad (\text{ArchSim})$$

s_q represents the structural condition in the query and s_m stands for the structure of the search term found in the collection. Both parameters α and β allow for finer tuning of the calculated similarity value. We also implemented a tag dictionary that contains values for the similarity between known tags. For example, *<author>* and *<writer>* are rather similar and a precise similarity value can be assigned to these tags. We are considering the possibility of giving the user the opportunity to define such similarities himself.

The *fuzzy match* type of functions takes into account gaps or wrong sequences in the different tags. As a representative of the class of functions based on cost calculation for transforming the query structure into the target one, we used a method based on the *Levenstein Edit Distance* to compute the similarity between two strings by counting the operations *delete*, *insert* and *replace* needed to transform one string into another. This method allows similar measuring of the difference between XML structures by considering every tag as a single character. We implemented and evaluated this method with a suitable normalization and, as above, an enhancement with a tag dictionary was also applied (**PathSim**).

Another approach for the similarity analysis of XML structures within the scope of *fuzzy* type strategies is the definition of a number of factors describing specific properties of the compared structures and combining them in a single function. Five such factors are proposed in **Fehler! Verweisquelle konnte nicht gefunden werden.**: semantic completeness (*SmCm*), *semantic correctness* (*SmCr*), *structural completeness* (*StCm*), *structural correctness* (*StCr*), and *structural cohesion* (*StCh*). They can be used for a deep and thorough analysis and representation of the different similarity aspects between two XML structures. We used the arithmetic mean to compute the *SmCr* and measured the similarities between tags. We used these factors to construct combined similarity functions. In order to compare these functions, we built a small but highly heterogeneous document collection with search terms occurring in many different XML contexts, resulting in a number of structures to be compared and ranked. Several functions were tested in the process with a number of parameters. We achieved the best ranking results with the following formula:

$$Sim_3 = \alpha \cdot \left(\frac{\omega_1 SmCm + \omega_2 SmCr}{2} \right) + \beta \cdot \left(\frac{\omega_3 StCm + \omega_4 StCr + \omega_5 StCh}{3} \right) \quad (\text{FineSim})$$

All similarity factors were normalized and parameter boundaries were set such that the resulting single similarity value remains within the interval [0,1]. Parameters α and β provide an opportunity to shift the weight of the similarity between the two

classes of factors – *semantic* and *structural*. The five parameters ω_i can be used for further fine-tuning. After a thorough evaluation, we chose the values $\alpha = 0.7$ and $\beta = 0.3$. All other factors were set to 1.

The three developed functions were compared with the *perfect match strategy* where exact match of structures is measured as well as the *baseline (flat)*, where no structure was taken into account (CO-run).

3 System used for Experiments: SPIRIX, a Distributed XML-Retrieval Solution

3.1 SPIRIX

For our experiments, we used *SPIRIX*, a P2P search engine for XML-Retrieval that is based on a structured P2P network (DHT). Figure 1 describes the architecture of each *SPIRIX* peer. When used for INEX experiments, the whole collection is indexed by one single peer. However, the information extracted by the *indexing component* is distributed over all participating peers such that each peer is responsible for storing, maintaining and –at querying time- providing parts of the global index. The indexed information consists of XTerms, which are tuples of XML structure and terms. For each XTerm or combination of XTerms (so called XHDKs), the according postinglist are distributed into the global index, using the P2P protocol *SpirixDHT* which is based on Chord. Term statistics about the indexed documents and their elements are also extracted and distribute in form of vectors, where each component of a vector represents the weight of an XTerm in a potentially relevant retrieval unit. All extracted information can be identified by a unique key and will be stored on the peer assigned to the hash value of this key. Postinglists are stored in the distributed *inverted index*, term statistics are stored in the *statistics index* (split into statistics for documents and statistics for elements).

At querying time, topics are prepared by the local *evaluation component* of the evaluating/querying peers. For this, each topic is converted into an XTerm-format and split into keys (XTerms or XTerm-combinations). For each key, a message is sent (using *SpirixDHT*) to the peer assigned to this key, i.e. holding the key's postinglist. From this postinglist, the best postings are selected, using a combination of ranking algorithms (*weighting calculator*), structural similarity functions (*similarity calculator*), and peer metrics (*source selector* and *peerMetrics calculator*). If the topic consists of several keys, the according postinglists have to be merged with each other by processing them in a pipelined manner: the shortest postinglist is selected, than sent to the peer assigned to the key with the next shortest postinglist etc. until all keys are processed, that is until all postings are selected.

For each selected posting, a ranking request message is send to the peer assigned to the term statistics of the document or element referenced by the posting. This document or element will be ranked by the local *ranking component*, using statistics information of the local part of the global statistics index. The best results will be sent back to the evaluating peer. As for each selected posting, a ranking request message is sent, the number of ranked documents and elements grow with the number of selected

postings. The chance of finding relevant results grows, accordingly. To achieve a high precision, enough postings have to be selected. However, each message produces network traffic. In order to get a scalable, efficient system, the number of selected postings / sent messages has to be reduced without losing too much effectiveness. XML-Retrieval algorithms and structural similarity functions to judge similarity between CAS-queries and postinglist keys are used to select adequate postings, taking advantage of XML-Retrieval methods to gain more efficiency in the P2P system and to guarantee its scalability. That is, on the one hand XML-Retrieval techniques are used to improve the P2P system.

How can, on the other hand, the retrieval performance be supported by the P2P design, especially when searching a large-size collection?

First of all, the global index is split over all participating peers such that each peer has to take care of only a fraction of the total index. This not only reduces the disk space consumed but also allows for a bigger portion of the local index to be hold in memory of each peer. At querying time, the processing of the postinglists is performed in parallel on all the peers that hold adequate postinglists. Those peers share the load of selecting the best postings in respect to computing power necessary for the selection algorithms, the memory used for this task, and the disk I/O for reading the postinglists. Also, the ranking process is performed in parallel: its load is split over all the peers that hold term statistics and thus get a ranking request message with the request to perform a local ranking. Hence, both the process of selecting postings and executing the relevance computations are performed in parallel. This is a big advantage when dealing with large-scale systems such as the new Wikipedia collection of INEX 2009.

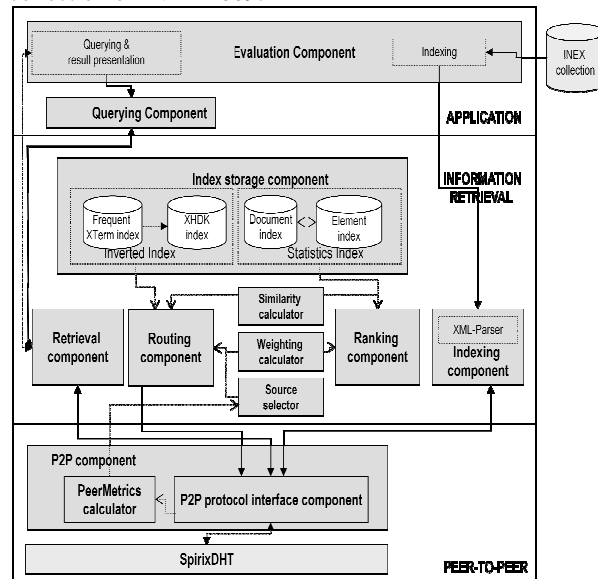


Fig. 1. Architecture of SPIRIX.

3.2 Technical Details

SPIRIX features element retrieval, can answer CO and CAS topics, indexes structure, does not support phrases, performs stemming and stopword removal, and punishes negative terms.

For participation at the Ad Hoc track, we aimed at high precision. Many parameters that influence the balance between effectiveness and efficiency were thus set to values that aim at effectiveness. For instance, this includes global statistics. For the selection of postings, we decided for a compromise of 1500 postings respectively 150 postings.

Global statistics: SPIRIX is based on a DHT which enables collecting and storing of global statistics. Usually, we estimate these statistics from the locally stored part of the distributed document index that contains randomly hashed samples from the collection. This estimation technique saves the messages necessary for distributing and accessing global statistics with the cost of loosing precision depending on the estimations. Thus, for the INEX runs the exact global statistics were used.

Structure size in index: For each term, a separate posting list for each of its structures is stored. This allows efficient selecting of postings according to structural similarity between hints in CAS topics and the stored structure of an XTerm. To reduce the variety of structures, several methods were tried such as:

- ◆ Stemming of tags (Snowball stemmer),
- ◆ Deleting stopword tags (e.g. conversionwarnings),
- ◆ Deleting of tags at specific positions to limit the structure size to 20 Bytes.

In comparison to a full structure index, we saved 40% of the index size (without compression) and reduced the amount of tags per structure to 3,3 tags on average. The total amount of different structures was reduced to 1539. However, evaluation showed that we can achieve a significant improvement on early precision measures when using structural similarity for ranking and routing – but only with the full structure index. With the reduced index, there is almost no improvement (less than 1%). We assume that the reduction to 20 bytes retarded our similarity functions from detecting the correct similarity between structures. Further experiments have to be conducted to analyze, which of the used techniques can be used to reduce the structure variety without losing precision. For the experiments submitted to INEX this year, we used two different indexes: one without structure and one with full structure.

Early termination (Selection of 1500 respectively 150 postings): Due to our system architecture, taking all postings from a posting list is not efficient as this leads to many ranking request messages. However, precision increases with the amount of selected postings (until a collection specific point). Thus, the best 1500 respectively 150 documents were selected from each query term's posting list. Note, that this is done on separate peers and thus without merging – we lose precision for multi term queries when good documents are on positions > 1500 respectively 150.

Element retrieval: Retrieval results can be either whole documents (Wikipedia articles) or XML elements out of these documents. In the indexing, elements are treated as documents, that is they are indexed independently by extracting their statistics and storing these in the retrieval unit index. However, we believe in a strong correlation between the relevance of an XML element and its parent document, as shown in related work. Therefore, the parent document influences the retrieval of elements: when ranking, the score of an element is computed based on the stored

statistics but smoothed by the score of its parent document. In the routing process, only evidence of the parent document and its best retrieval unit are used for the decision, which documents plus their containing elements are to be ranked.

Indexing of structure: XML-structure is considered both in the routing and in the ranking. It is extracted while parsing the XML-document at indexing and stored together with the extracted terms.

Phrases: Phrases are not supported. Terms that appear together in a phrase are regarded independently.

Text processing: Stemming and removal of stopwords are used.

Small elements: All elements with less than 15 words are removed.

Negative terms: Results that contain negative terms (specified by the user by “not” respective “-“) are still ranked. However, their relevance is reduced by the weight of all negative terms multiplied with a negativity factor.

4 Submitted Runs

Table 1. Runs submitted by University of Applied Science, Frankfurt.

<i>Name</i>	<i>Task</i>	<i>CO / CAS</i>	<i>Ranking function</i>	<i>Structural similarity function: Ranking</i>	<i>Routing function</i>	<i>Structural similarity function: Routing</i>	<i>#Postings per peer</i>
Spirix09R001 (#872)	Ad Hoc: Focused	CO	BM25-Adaption	Flat	-	-	1500
Spirix09R002 (#873)	Ad Hoc: Focused	CO	tf*idf-Adaption	Flat	-	-	1500
Spirix09X01	Eff.: typeA	CAS	BM25	Flat	BM25E	Flat	1500
Spirix09X02	Eff.: typeA	CAS	BM25E	Flat	BM25E	Flat	150
Spirix09X03	Eff.: typeA	CAS	BM25E	PathSim	BM25E	PathSim	150
Spirix09X04	Eff.: typeA	CAS	BM25E	FineSim	BM25E	FineSim	150
Spirix09X05	Eff.: typeA	CAS	BM25E	ArchSim	BM25E	ArchSim	150
Spirix09X06	Eff.: typeA	CAS	BM25E	Strict	BM25E	Strict	150
Spirix09X07	Eff.: typeA	CAS	BM25	Flat	Baseline	Flat	1500
Spirix09X08	Eff.: typeA	CAS	BM25E	Flat	Baseline	Flat	150
Spirix09X09	Eff.: typeA	CAS	BM25E	PathSim	Baseline	Flat	150
Spirix09X10	Eff.: typeA	CAS	BM25E	FineSim	Baseline	Flat	150
Spirix09X11	Eff.: typeA	CAS	BM25E	ArchSim	Baseline	Flat	150
Spirix09X12	Eff.: typeA	CAS	BM25E	Strict	Baseline	Flat	150
Spirix09X13	Eff.: typeB	CAS	BM25	Flat	Baseline	Flat	1500
Spirix09X14	Eff.: typeB	CAS	BM25E	Flat	BM25E	Flat	150
Spirix09X15	Eff.: typeB	CAS	BM25E	PathSim	BM25E	Flat	150
Spirix09X16	Eff.: typeB	CAS	BM25E	FineSim	BM25E	Flat	150
Spirix09X17	Eff.: typeB	CAS	BM25E	ArchSim	BM25E	Flat	150
Spirix09X18	Eff.: typeB	CAS	BM25E	Strict	BM25E	Flat	150
Spirix09X19	Eff.: typeB	CAS	BM25	Flat	BM25E	Flat	1500
Spirix09X20	Eff.: typeB	CAS	BM25E	Flat	BM25E	Flat	150
Spirix09X21	Eff.: typeB	CAS	BM25E	PathSim	BM25E	PathSim	150
Spirix09X22	Eff.: typeB	CAS	BM25E	FineSim	BM25E	FineSim	150
Spirix09X23	Eff.: typeB	CAS	BM25E	ArchSim	BM25E	ArchSim	150
Spirix09X24	Eff.: typeB	CAS	BM25E	Strict	BM25E	Strict	150

We submitted two runs for the Ad Hoc Track (Focused Task) to compare two different ranking functions without the use of structural hints, e.g. we used CO-queries. We furthermore submitted several runs for the Efficiency Track where CO- and CAS-queries were applied for ranking or routing. Here, we evaluated four different structural similarity functions and compared the results with a run, where no structural similarity was taken into account (CO-run). Table 1 gives an overview of the submitted runs.

5 Evaluation

5.1 Ad Hoc Track Results

At the Ad Hoc Track (Focused), our runs achieved 59,03% and 58,91% precision at recall level 1% (iP[0.01]). In the official list of the top-10 systems, we thus are on rank 7, with the best performing system being the University of Waterloo with 63,3% precision.

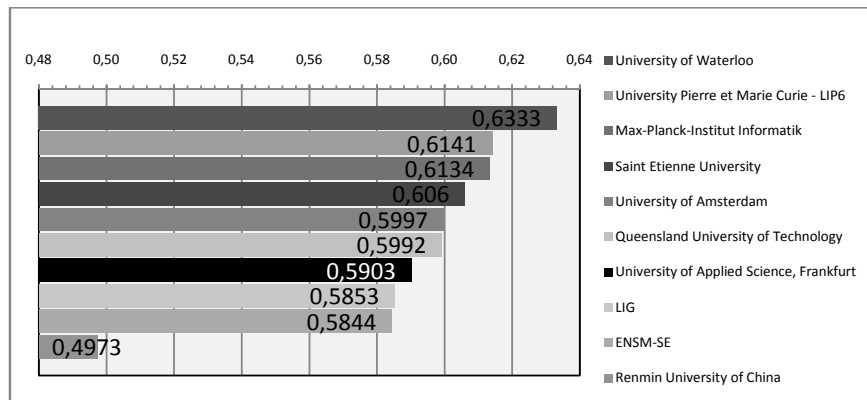


Fig. 2. Rank 7 in the top-10 systems of Ad Hoc Track (Focused).

Our best performing run was Spirix09R001, where an BM25-Adaption was applied. When analyzing the Recall-Precision graph in figure 3, it can be seen that the curve descends slower than that of most other participants. From what we can see at the official INEX webpage results, this is the case for only the best 5 systems. As so far only the iP[0.01] values have been released, we cannot make an exact statement on this but we expect that SPIRIX' MAiP will be significantly higher than that of other search engines. So, SPIRIX seems to be very good in identifying the best 1% results but also is able to find good results on higher recall levels.

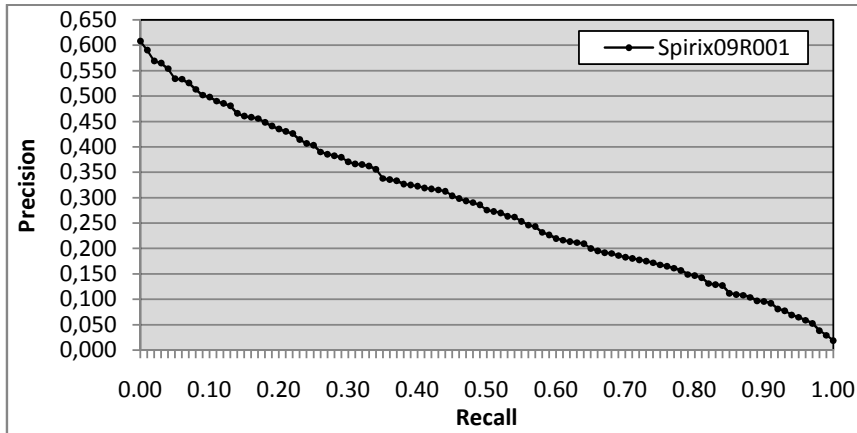


Fig. 3. Relatively flat Recall-Precision graph of the best performing run Spirix09R001.

Figure 4 shows a comparison of both SPIRIX runs for early recall levels. Run Spirix09R001, where a BM25-Adaption was applied, performs only slightly better than Run Spirix09R002, where a Robertson TF*IDF-Adaption to the use of our XTerms was applied. The differences are not significant and can hardly be seen. The same applies for higher recall level and for MAiP.

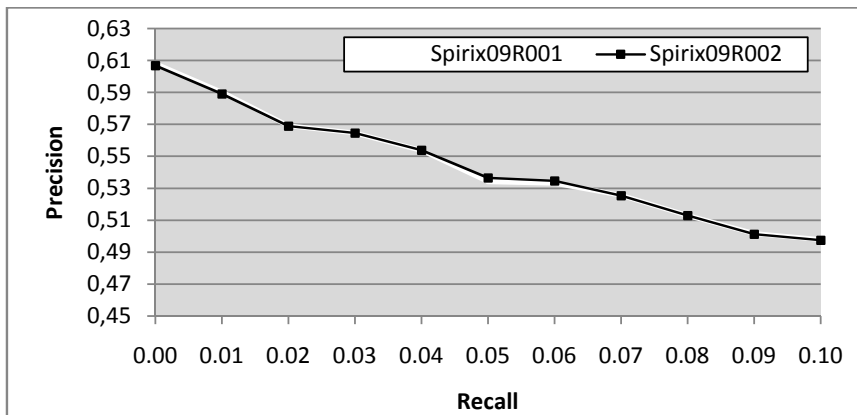


Fig. 4. Comparison of both SPIRIX runs (different ranking functions).

5.2 Efficiency Track Results

At the Efficiency track, we achieved the highest precision in both tasks, which is 59,9% for the type A topics and 46,4% for the type B topics. Our notion of efficiency is to get a scalable system by reducing the bandwidth consumed by messages between

peers. Thus, we aimed at reducing the amount of postings to be used. All other participants aimed at reducing run times. SPIRIX was not developed or tuned to achieve fast response times. Thus, for the type A topics our run times are the slowest. For the type B topics, most of our runs have been in the same range of run times as the other participants except for the system from university of Otago, which performed extraordinary fast in all runs and tasks. Unfortunately, we are still the only distributed system. Therefore, our efforts to be efficient in terms of communication overhead cannot be compared with other participants. A comparison of our runs between themselves will be conducted in the following-up paper of the preproceedings as the results have only been released recently and we have not been able to analyse them yet. For some of the experiments, the use of structure in the routing process showed a clear increase of precision, e.g. when analyzing the use of our BM25E-Adaption in the routing, as shown in figure 5.

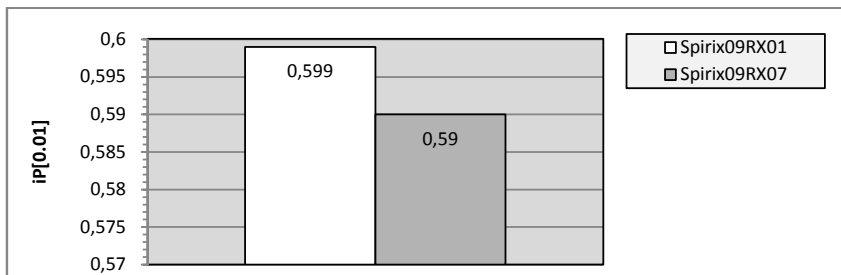


Fig. 5. Run RX01 used BM25E as routing function and thus outperformed Run RX07.

As shown in figure 6 and 7, our runs ranked best in terms of precision. Figure 6 shows the precision $iP[0.01]$ of the top runs in regard of the task where type A topics were queried. Our best runs, e.g. RX01, achieved 59,9% precision at this task.

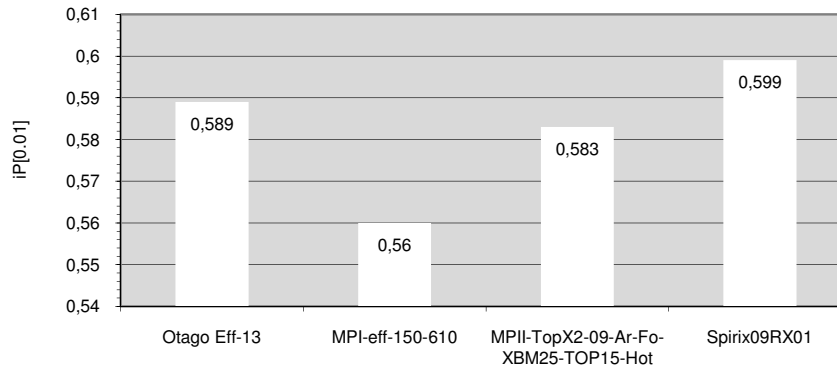


Fig. 6. Top runs for type A topics.

Figure 7 displays $iP[0.01]$ for the top runs, this time for the type B topics. Again, SPIRIX performed best, e.g. with run RX20 and $iP[0.01]$ of 46,6%.

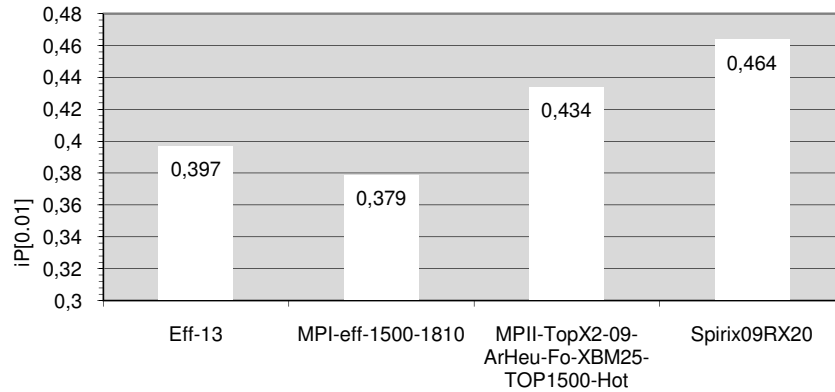


Fig. 7. Top runs for type B topics.

5 Discussion

Our results show, that we gained high precisions in comparison to other INEX participants. We are able to report a precision of 59,3% where the best performing system achieved 63,3%, resulting in rank 7 at the Ad Hoc Track (Focused). At the Efficiency track, we achieved the highest precision in both tasks, which is 59,9% for the type A topics and 46,4% for the type B topics. These results were achieved with a P2P system, where communication overhead has to be considered and thus not all postings of the inverted index can be considered. XML-Retrieval techniques are used to select adequate postings in the routing process. Our INEX results show, that our selection algorithms - based on a BM25E-adaption, on several structural similarity functions, and on a peer metrics computation - work in a way, that on the one hand, high precisions can be achieved while on the other hand, the system scales well. Furthermore, the parallelisation of both the posting selection and the ranking process is a big advantage when dealing with large-scale collections like the new Wikipedia collection, e.g. by being able to process more postings in memory at the same time.

Overview of the INEX 2009 Entity Ranking Track

Gianluca Demartini¹, Tereza Iofciu¹, and Arjen de Vries²

¹ L3S Research Center
Leibniz Universität Hannover
Appelstrasse 9a D-30167 Hannover, Germany
{demartini,iofcui}@L3S.de

² CWI & Delft University of Technology
The Netherlands
arjen@acm.org

Abstract. In some situations search engine users would prefer to retrieve entities instead of just documents. Example queries include Italian Nobel prize winners, Formula 1 drivers that won the Monaco Grand Prix, or German spoken Swiss cantons. The XML Entity Ranking (XER) track at INEX creates a discussion forum aimed at standardizing evaluation procedures for entity retrieval. This paper describes the XER tasks and the evaluation procedure used at the XER track in 2009 as well as summarise the approaches adopted by the participants.

1 Introduction

Many user tasks would be simplified if search engines would support typed search, and return entities instead of 'just' web pages. Since 2007, INEX has organised a yearly XML Entity Ranking track (INEX-XER) to provide a forum where researchers may compare and evaluate techniques for engines that return lists of entities. In entity ranking (ER) and entity list completion (LC), the goal is to evaluate how well systems can rank entities in response to a query; the set of entities to be ranked is assumed to be loosely defined by a generic category, implied in the query itself, or by some example entities. We continue to run both the entity ranking and list completion tasks this year. This year we will adopt the new document collection containing annotations with the general goal to understand how such semantic annotations can be exploited for improving Entity Ranking.

Entity ranking concerns triples of type $\langle \text{query}, \text{category}, \text{entity} \rangle$. The category (that is entity type), specifies the type of 'objects' to be retrieved. The query is a free text description that attempts to capture the information need. Entity specifies example instances of the entity type. The usual information retrieval tasks of document and element retrieval can be viewed as special instances of this more general retrieval problem, where the category membership relates to a syntactic (layout) notion of 'text document', or 'XML element'. Expert finding uses the semantic notion of 'people' as its category, where the query would

specify ‘expertise on T’ for expert finding topic T. Our goal is not to evaluate how well systems identify instances of entities within text (to some extent this is part of the goal of the Link-the-Wiki track).

2 INEX-XER Setup

2.1 Data

The track uses the new Wikipedia 2009 XML data based on a dump of the Wikipedia taken on 8 October 2008 and annotated with techniques described in [3]. Available annotations can be exploited to find relevant entities to return. Category information about the pages loosely defines the entity sets. The entities in such a set are assumed to loosely correspond to those Wikipedia pages that are labeled with this category (or perhaps a sub-category of the given category). Obviously, this is not perfect as many Wikipedia articles are assigned to categories in an inconsistent fashion. Retrieval methods should handle the situation that the category assignments to Wikipedia pages are not always consistent, and also far from complete. The challenge for participants is to exploit the rich information from text, structure, links and annotations to perform the search tasks.

2.2 Tasks

This year’s entity ranking track consists of two tasks, i.e., entity ranking (without examples), and entity list completion (with examples). Entity list completion is a special case of entity ranking where a few examples of relevant entities are provided as relevance feedback information.

Entity Ranking. The motivation for the entity ranking (ER) task is to return entities that satisfy a topic described in natural language text. Given preferred categories, relevant entities are assumed to loosely correspond to those Wikipedia pages that are labeled with these preferred categories (or perhaps sub-categories of these preferred categories). Retrieval methods need to handle the situation where the category assignments to Wikipedia pages are not always consistent, and also far from complete. For example, given a preferred category ‘art museums and galleries’, an article about a particular museum such as the ‘Van Gogh Museum’ (155508) may not be labeled by ‘art museums and galleries’ but labeled by a sub-category of the preferred category instead, such as category ‘art museums and galleries in the Netherlands’. Therefore, when searching for “art museums in Amsterdam”, correct answers may belong to other categories close to this category in the Wikipedia category graph, or may not have been categorized at all by the Wikipedia contributors. The category ‘art museums and galleries’ is only an indication of what is expected, not a strict constraint (like in the CAS title for the ad-hoc track).

List Completion. List completion (LC) is a sub-task of entity ranking which considers relevance feedback information. Instead of knowing the desired category (entity type), the topic specifies a number of correct entities (instances) together with the free-text context description. Results consist again of a list of entities (Wikipedia pages). If we provide the system with the topic text and a number of entity examples, the task of list completion refers to the problem of completing the partial list of answers. As an example, when ranking 'Countries' with topic text 'European countries where I can pay with Euros', and entity examples such as 'France', 'Germany', 'Spain', then the 'Netherlands' would be a correct completion, but the 'United Kingdom' would not.

2.3 Topics

Based on the topics from the previous years, we have set up a collection of 60 Entity Ranking topics, with 25 from 2007 and 35 topics from 2008. The <categories> part is supposed to be used exclusively for the Entity Ranking Task. The <entities> part is supposed to be used exclusively for the List Completion Task.

2.4 The 2009 test collection

The initial set of topics for the 2009 XER Track consisted of 60 topics selected from the 2007 and 2008 editions. As the number of groups participating to the relevance assessments in 2009 is the same as in 2008 (i.e., 6 groups) we decided to keep the assessment effort comparable and, instead of limiting the pool size, we decided to limit the number of topics to be assessed. In Figure 1 it is possible to see the number of relevant entities per topic as from last year's judgements. In 2008, we performed a selection of topics for the final set by removing those with more than 74 and less than 7 relevant entities (see [2]). If we perform the same cut on 2007 topics we would have 52 topics left which is a comparable number with the 50 topics assessed in the each previous year.

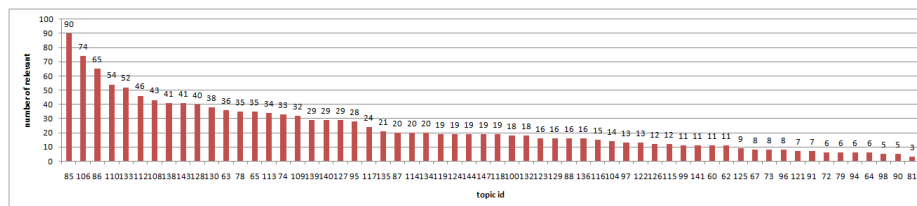


Fig. 1. Number of relevant entities per topic as from last years relevance judgements

3 Participants

At INEX-XER 2009 five groups submitted runs for both the ER and LC tasks. We received a total of 12 ER runs and 11 LC runs. In the following we report a short description of the approaches used as reported by the participants.

Waterloo. Our two runs for each task is based on Clarke et al.’s question answering technique that uses redundancy [1]. Specifically, we obtained top scoring passages from each article in the corpus using topic titles (for ER task) and topic titles+examples (for LC task). For LC task, we estimated the categories of entities to return as the union of categories in the examples. Within each top scoring passages, we located candidate terms that have a Wikipedia page that fall under the desired categories. We ranked the candidate terms by the number of distinct passages that contain the term.

AU-CEG (Anna University, Chennai). In our approach, we have extracted the Entity Determining Terms (EDTs), Qualifiers and prominent n-grams from the query. As a second step, we strategically exploit the relation between the extracted terms and the structure and connectedness of the corpus to retrieve links which are highly probable of being entities and then use a recursive mechanism for retrieving relevant documents through the Lucene Search. Our ranking mechanism combines various approaches that make use of category information, links, titles and WordNet information, initial description and the text of the document.

PITT team (School of Information Sciences, University of Pittsburgh). As recent studies indicate that named entities exist in queries and can be useful for retrieval, we also notice the ubiquitous existence of entities in entity ranking queries. Thus, we try to consider entity ranking as the task of finding entities related to existing entities in a query. We implement two generative models, i.e. MODEL1EDR and MODEL1EDS, both of which try to capture entity relations. These two models are compared with two baseline generative models: MODEL1D, which estimates models for each entity using Wikipedia entity documents; MODEL1E, which interpolates entity models in MODEL1D with entity category models.

UAms (Turfdraagsterpad). We rank entities by combining a document score, based on a language model of the document contents, with a category score, based on the distance of the document categories to the target categories. We extend our approach from last year by using Wordnet categories and by refining the categories we use as target categories.

UAms (ISLA). We propose a novel probabilistic framework for entity retrieval that explicitly models category information in a theoretically transparent manner. Queries and entities are both represented as a tuple: a term-based plus a

category-based model, both characterized by probability distributions. Ranking of entities is then based on similarity to the query, measured in terms of similarities between probability distributions.

Discussion. It is possible to notice that a general behaviour of participants this year was to identify entity mentions in the text of Wikipedia articles, passages, or queries. They then apply different techniques (e.g., detect entity relations, exploit category information) to produce a ranked list of Wikipedia articles that represents the retrieved entities.

4 Conclusions and Further Work

As remaining steps for completing the XER 2009 track the participants will perform relevance assessments for the 52 selected topics and the evaluation of submitted runs will be done using xInfAP [4] as effectiveness metrics. Additionally, we will perform a comparative analysis of the relevance judgements done for INEX-XER 2009 on the new dataset with the ones from past years performed on an older version of Wikipedia.

References

1. Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. Exploiting redundancy in question answering. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365, New York, NY, USA, 2001. ACM.
2. Gianluca Demartini, Arjen P. de Vries, Tereza Iofciu, and Jianhan Zhu. Overview of the inex 2008 entity ranking track. In Shlomo Geva, Jaap Kamps, and Andrew Trotman, editors, *INEX*, volume 5631 of *Lecture Notes in Computer Science*, pages 243–252. Springer, 2008.
3. R. Schenkel, FM Suchanek, and G. Kasneci. YAWN: A semantically annotated Wikipedia XML corpus. In *Symposium on Database Systems for Business, Technology and the Web of the German Society for Computer science (BTW 2007)*, 2007.
4. Emine Yilmaz, Evangelos Kanoulas, and Javed A. Aslam. A simple and efficient sampling method for estimating AP and NDCG. In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *SIGIR*, pages 603–610. ACM, 2008.

The University of Amsterdam (ISLA) at INEX 2009

Krisztian Balog, Jiyin He, Marc Bron, Maarten de Rijke, and Wouter Weerkamp

ISLA, University of Amsterdam, Science Park 107, 1098 XG Amsterdam, The Netherlands
{k.balog, j.he, m.bron, derijke, w.weerkamp}@uva.nl

Abstract. We describe our participation in the INEX 2009 Entity Ranking and Link-the-Wiki tracks. We provide a detailed account of the ideas underlying our approaches to these tasks.

1 Introduction

This year the Intelligent Systems Lab Amsterdam at the University of Amsterdam participated in two INEX tracks: Entity Ranking and Link-the-Wiki.

For the Entity Ranking track our main emphasis was to evaluate a recently proposed probabilistic framework for entity retrieval that explicitly models category information in a theoretically transparent manner [2]. Information needs and entities are represented as a tuple: a term-based model plus a category-based model, both characterized by probability distributions over words. Ranking of entities is then based on similarity to the query, measured in terms of similarity between probability distributions. In our participation, our focus is on two core steps: query modeling and query model expansion. Moreover, we seek to answer how well parameter settings trained on the 2007 and 2008 editions of the Entity Ranking track perform on this year's setup.

In our participation in the Link-the-Wiki track our main aim was to explore the effectiveness of learning methods and learning materials for automatic generation of outgoing links. We experimented with two types of learning approaches: a classification-based approach and a ranking-based approach. We train the classifier as well as the ranker on two different versions of Wikipedia collections.

In this paper, we describe our participation for the tracks mentioned above, in two largely independent sections: Section 2 is devoted to our entity ranking track participation and Section 3 is devoted to our work in the link-the-wiki track. We conclude in Section 4.

2 Entity Ranking

In this section we present a probabilistic retrieval framework for the two tasks that have been formulated within the Entity Ranking track. In the *entity ranking* task we are given a query (q) and a set of target categories (C) and have to return entities. For *list completion* we need to return entities given a query (q), a set of similar entities (E), and (optionally also) a set of target categories (C).

Balog et al. [2] recently proposed a probabilistic retrieval model for entity search, in which term-based and category-based representations of queries and entities are effectively integrated. With the exception of the formula used for weighting terms for query expansion, we present the original approach unchanged.

The remainder of this section is organized as follows. In §2.1 we introduce our retrieval model, followed by the discussion of entity and query models in §2.2 and §2.3, respectively. We discuss our submitted runs in §2.4.

2.1 Modeling Entity Ranking

We rank entities e according to their probability of being relevant given the query q : $P(e|q)$. Instead of estimating this probability directly, we apply Bayes' rule and rewrite it to:

$$P(e|q) \propto P(q|e) \cdot P(e), \quad (1)$$

where $P(q|e)$ expresses the probability that query q is generated by entity e , and $P(e)$ is the *a priori* probability of e being relevant, i.e., the entity prior.

Each entity is represented as a pair: $\theta_e = (\theta_e^T, \theta_e^C)$, where θ_e^T is a distribution over terms and θ_e^C is a distribution over categories. Similarly, the query is also represented as a pair: $\theta_q = (\theta_q^T, \theta_q^C)$, which is then (optionally) refined further, resulting in an expanded query model that is used for ranking entities.

The probability of an entity generating the query is estimated using a mixture model:

$$P(q|e) = \lambda \cdot P(\theta_q^T | \theta_e^T) + (1 - \lambda) \cdot P(\theta_q^C | \theta_e^C), \quad (2)$$

where λ controls the interpolation between the term-based and category-representations. The estimation of $P(\theta_q^T | \theta_e^T)$ and $P(\theta_q^C | \theta_e^C)$ requires a measure of the difference between two probability distributions. Here, we opt for the Kullback-Leibler divergence—also known as the relative entropy. The term-based similarity is estimated as follows:

$$P(\theta_q^T | \theta_e^T) \propto -KL(\theta_q^T || \theta_e^T) = - \sum_t P(t | \theta_q^T) \cdot \frac{P(t | \theta_q^T)}{P(t | \theta_e^T)}, \quad (3)$$

where the probability of a term given an entity model ($P(t | \theta_e^T)$) and the probability of a term given the query model ($P(t | \theta_q^T)$) remain to be defined. Similarly, the category-based component of the mixture in Eq. 2 is calculated as:

$$P(\theta_q^C | \theta_e^C) \propto -KL(\theta_q^C || \theta_e^C) = - \sum_c P(c | \theta_q^C) \cdot \frac{P(c | \theta_q^C)}{P(c | \theta_e^C)}, \quad (4)$$

where the probability of a category according to an entity's model ($P(c | \theta_e^C)$) and the probability of a category according to the query model ($P(c | \theta_q^C)$) remain to be defined.

2.2 Modeling Entities

Term-based representation To estimate $P(t | \theta_e^T)$ we smooth the empirical entity model with the background collection to prevent zero probabilities. We employ Bayesian

smoothing using Dirichlet priors which has been shown to achieve superior performance on a variety of tasks and collections [9, 6] and set:

$$P(t|\theta_e^T) = \frac{n(t, e) + \mu^T \cdot P(t)}{\sum_t n(t, e) + \mu^T}, \quad (5)$$

where $n(t, e)$ denotes the number of times t occurs in the document, $\sum_t n(t, e)$ is the total number of term occurrences, i.e., the document length, and $P(t)$ is the background model (the relative frequency of t in the collection). Since entities correspond to Wikipedia articles, this representation of an entity is identical to constructing a smoothed document model for each Wikipedia page, in a standard language modeling approach [8, 5]. Alternatively, the entity model can be expanded with terms from related entities, i.e., entities sharing the categories or entities linking to or from the Wikipedia page [3]. To remain focused, we do not explore this direction here.

Category-based representation Analogously to the term-based representation, we smooth the maximum-likelihood estimate with a background model. We employ Dirichlet smoothing, and use the parameter μ^C to avoid confusion with μ^T :

$$P(c|\theta_e^C) = \frac{n(c, e) + \mu^C \cdot P(c)}{\sum_c n(c, e) + \mu^C}. \quad (6)$$

In Eq. 6, $n(c, e)$ is 1 if entity e is assigned to category c , and 0 otherwise; $\sum_c n(c, e)$ is the total number of categories to which e is assigned; $P(c)$ is the background category model and is set using a maximum-likelihood estimate:

$$P(c) = \frac{\sum_e n(c, e)}{\sum_c \sum_e n(c, e)}, \quad (7)$$

where $\sum_c \sum_e n(c, e)$ is the number of category-entity assignments in the collection.

Entity priors By default, we use uniform entity priors, i.e., all pages in the collection are equally likely to be returned. Additionally, we experiment with priors that reward pages that are known to belong to entities; we use the 2007 and 2008 topic sets for setting the priors.

2.3 Modeling Queries

In this subsection we introduce methods for estimating and expanding query models. This boils down to estimating the probabilities $P(t|\theta_q^T)$ and $P(c|\theta_q^C)$ as discussed in §2.1.

Term-based representation The term-based component of the baseline query model is defined as follows:

$$P(t|\theta_q^T) = P_{bl}(t|\theta_q^T) = \frac{n(t, q)}{\sum_t n(t, q)}, \quad (8)$$

where $n(t, q)$ stands for the number of times term t occurs in query q .

The general form we use for expansion is a mixture of the baseline (subscripted with bl) defined in Eq. 8 and an expansion (subscripted with ex):

$$P(t|\theta_q^T) = (1 - \lambda^T) \cdot P_{bl}(t|\theta_q^T) + \lambda^T \cdot P_{ex}(t|\theta_q^T). \quad (9)$$

Given a set of feedback entities FB , the expanded query model is constructed as follows:

$$P_{ex}(t|\theta_q^T) = \frac{P_{K_T}(t|FB)}{\sum_{t'} P_{K_T}(t'|FB)}, \quad (10)$$

where $P_{K_T}(t|FB)$ is estimated as follows. First, $P(t|FB)$ is computed according to Eq. 11. Then, the top K_T terms with the highest $P(t|FB)$ value are taken to form $P_{K_T}(t|FB)$, by redistributing the probability mass, in proportion to their corresponding $P(t|FB)$ values.

$$P(t|FB) = \frac{1}{|FB|} \sum_{e \in FB} \frac{s(t, e)}{\sum_t s(t, e)} \quad (11)$$

and

$$s(t, e) = \log \frac{n(t, e)}{P(t) \cdot \sum_t n(t, e)}, \quad (12)$$

where $\sum_t n(t, e)$ is the total number of terms, i.e., the length of the document corresponding to entity e . (This is the same as the *EXP* query model generation method using example documents from [1], with the simplification that all feedback documents are assumed to be equally important.)

The set of feedback entities, FB , is defined in two ways: for the entity ranking task, it is the top N relevant entities according to a ranking obtained using the initial (baseline) query. For the list completion task, the set of example entities provided with the query are used as the feedback set ($FB = E$).

Category-based representation Our baseline model uses the keyword query (q) to infer the category-component of the query model (θ_q^C), by considering the top N_c most relevant categories given the query; relevance of a category is estimated based on matching between the name of the category and the query, i.e., a standard language modeling approach on top of an index of category names.

$$P(c|\theta_q^C) = P_q(c|\theta_q^C) = \begin{cases} P(q|\theta_c) / \sum_{c \in C} P(q|\theta_c), & \text{if } c \in \text{top } N_c \\ 0, & \text{otherwise.} \end{cases} \quad (13)$$

Note that this method does not use the category information provided with the query. To use target category information, we set $n(c, q)$ to 1 if c is a target category, and $\sum_c n(c, q)$ to the total number of target categories provided with the topic statement. Then, we put

$$P_c(c|\theta_q^C) = \frac{n(c, q)}{\sum_c n(c, q)}. \quad (14)$$

To combine the two methods (categories relevant to the query and categories provided as input), we put:

$$P(c|\theta_q^C) = \frac{1}{2} P_q(c|\theta_q^C) + \frac{1}{2} P_c(c|\theta_q^C). \quad (15)$$

(For the sake of simplicity, each model contributes half of the probability mass.)

Expansion of the category-based component is performed similarly to the term-based case; we use a linear combination of the baseline (either Eq. 13 or Eq. 15) and expanded components:

$$P(c|\theta_q^C) = (1 - \lambda^C) \cdot P_{bl}(c|\theta_q^C) + \lambda^C \cdot P_{ex}(c|\theta_q^C). \quad (16)$$

Given a set of feedback entities FB , the expanded query model is constructed as follows:

$$P_{ex}(c|\theta_q^C) = \frac{P_{K_C}(c|FB)}{\sum_{c'} P_{K_C}(c'|FB)}, \quad (17)$$

where $P_{K_C}(c|FB)$ is calculated similarly to the term-based case: first, $P(c|FB)$ is calculated according to Eq. 18 (where, as before, $n(c, e)$ is 1 if e belongs to c). Then, the top K_C categories with the highest $P(c|FB)$ value are selected, and their corresponding probabilities are renormalized, resulting in $P_{K_C}(c|FB)$.

$$P(c|FB) = \frac{1}{|FB|} \sum_{e \in FB} \frac{n(c, e)}{\sum_t n(c, e)}. \quad (18)$$

The set of feedback entities is defined as before (the top N entities obtained using blind relevance feedback for entity ranking, and the example entities E for list completion).

2.4 Runs

Parameter settings Using the 2007 and 2008 editions of the Entity Ranking track as training material, we set the parameters of our models as follows.

- Importance of the term-based vs. the category-based component (Eq. 2): $\lambda = 0.7$
- Number of categories obtained given the query (Eq. 13): $N_c = 15$
- Number of feedback entities: $N = 3$
- Number of feedback terms (Eq. 17): $K_T = 35$
- Weight of feedback terms (Eq. 9): $\lambda^T = 0.7$
- Number of feedback categories (Eq. 17): $K_C = \infty$ (not limited)
- Weight of feedback categories (Eq. 16): $\lambda^C = 0.3$

Entity ranking Table 1 summarizes the 4 runs we submitted for the entity ranking task.

List completion Table 2 summarizes the 6 runs we submitted for the list completion task.

RunID	Baseline		Expanded		Priors	Description
(UAmSISLA_ER....)	$P(t \theta_q^T)$	$P(c \theta_q^C)$	$P(t \theta_q^T)$	$P(c \theta_q^C)$		
TC_ERbaseline	Eq. 8	Eq. 15	-	-	N	Baseline
TC_ERfeedback	Eq. 8	Eq. 15	Eq. 10	Eq. 17	N	Feedback
TC_ERfeedbackS	Eq. 8	Eq. 15	Eq. 10	Eq. 17	N	Feedback (selected topics*)
TC_ERfeedbackSP	Eq. 8	Eq. 15	Eq. 10	Eq. 17	Y	Priors (on top of previous run)

Table 1. Entity ranking runs. (*Topics that were helped by blind feedback on the 2007/2008 topic set.)

RunID	Baseline		Expanded		Priors	Description
(UAmSISLA_LC....)	$P(t \theta_q^T)$	$P(c \theta_q^C)$	$P(t \theta_q^T)$	$P(c \theta_q^C)$		
TE_LCexpT	Eq. 8	Eq. 13	Eq. 10	-	N	Feedback (term-based)
TE_LCexpC	Eq. 8	Eq. 13	-	Eq. 17	N	Feedback (category-based)
TE_LCexpTC	Eq. 8	Eq. 13	Eq. 10	Eq. 17	N	Feedback (term- + category-based)
TE_LCexpTCP	Eq. 8	Eq. 13	Eq. 10	Eq. 17	Y	Priors (on top of previous run)
TEC_LCexpTCS	Eq. 8	Eq. 13/15	Eq. 10	Eq. 17	N	Feedback (selected topics*)
TEC_LCexpTCSP	Eq. 8	Eq. 13/15	Eq. 10	Eq. 17	Y	Priors (on top of previous run)

Table 2. List completion runs. (*Topics that were helped by using example entities on the 2007/2008 topic set do not use input category information (i.e., use Eq. 13 for constructing $P_{bi}(c|\theta_q^C)$); the remainder of the topics use the input category information (i.e., $P_{bi}(c|\theta_q^C)$ is estimated using Eq. 15).)

3 Link-the-Wiki

In this section, we describe our participation in the Link-the-Wiki (LTW) track. The aim of the LTW track is to automatically identify hyperlinks between documents. We only participated in the task of outgoing link generation within Wikipedia (A2B). In our experiments, we focus on exploring machine learning methods and learning material for link detection.

The main purpose of our experiments are two-fold. First, we want to test how our learning methods work on the LTW task, especially how the results learnt from Wikipedia ground truth would be judged by human assessors. On top of that, since the LTW task is defined as a ranking problem for recommendation purposes, we want to see how a learning to rank approach works as it directly optimizes the rankings instead of assigning binary decisions to candidate links as a classification method would do. Second, we trained our models with different versions of Wikipedia. The two versions used, namely Wikipedia 2008 and Wikipedia 2009, differ in the amount of articles as well as the amount of links. Presumably, the 2009 version contains more link information but is also more noisy in terms of missing target pages (as some pages are deleted as time passes by). We experiment with both collections so as to see the impact of the training material used.

Learning Stage	N-gram	N-gram-target	Target	N-gram-topic	Topic-target	1st-stage
Candidate targets ranking		x	x		x	
Candidate links ranking	x	x	x	x	x	x

Table 3. Features and their corresponding application in different learning stages.

3.1 A two-stage learning procedure

Following [7], we consider the linking task as a two stage procedure, namely *candidate target identification* and *link detection*. First, we extract all possible n-grams in a topic page, and train a link-detector to rank the potential target pages for each n-gram, which we refer to as *candidate target pages*.

We experiment with two types of learning methods, namely classification and learning to rank. For classification, we use SVM to classify the instances in both stages, and rank the results by the probability of an instance being positive. For our learning to rank approach, we use RankingSVM [4] to directly optimize the ranking of an instance. In the candidate target identification stage, we train a ranker to rank the target candidates for each n-gram and in the link detection stage a ranker is trained to rank the n-gram target pairs.

3.2 Features

We identify 6 types of feature for learning a preference relation between the candidate links. Table 3 specifies in which stage each type is used and Table 4 lists the features. Here, we discuss the motivations of using these features, as well as detail the formulation of some of the features.

N-gram features The n-gram features suggest how likely a given n-gram would be marked as an anchor text, without any other information such as its context in the topic page, which includes its length, IDF score, number of candidate targets associated with it, and its *ALR* (Anchor Likelihood Ratio) scores. IDF is calculated as

$$\log\left(\frac{|D|}{|\{d_i : ng \in d_i\}_{i=1}^N|}\right),$$

where ng is a n-gram, d_i is a page containing a this n-gram, and D is the total collection of Wikipedia pages. The *ALR* score can be interpreted as a model selection between two models, the anchor model and the collection model, from which a n-gram is generated. To calculate the probability of a n-gram being generated by either model, the maximum likelihood is used. Specifically, it is calculated as

$$ALR(ng) = \frac{|ng \in A|}{|A|} \cdot \frac{|C|}{|ng \in C|} \quad (19)$$

where A is the collection of all anchor texts in the Wikipedia collection and C is the Wikipedia collection. A large *ALR* value indicates that the n-gram is more likely to have been generated from the anchor model, i.e., this n-gram is more likely to be an anchor text than a common word sequence from the background collection.

N-gram - target features The n-gram - target features describe how well an n-gram and its corresponding candidate target page are related. On the assumption that each Wikipedia page is about a specific concept that is usually denoted by its title, the first feature we use is the match between an n-gram and the candidate target page. The second type of feature in this category consists of indicators of how likely a given n-gram ng and a candidate target page $ctar$ are linked, which is expressed by the following two scores: *RatioLink* and *RatioAnchor*. The former is the ratio between the number of times ng and $ctar$ are linked and the number of times $ctar$ is being linked as a target page in the collection. The latter, i.e., *RatioAnchor* is the ratio between the number of times ng and $ctar$ are linked and the number of times ng is used as an anchor text in the collection. Moreover, we adopt retrieval scores between the n-gram and the candidate target pages as features (n-gram as query), which is an obvious description of the relatedness of the two:

$$RatioLink(ng, ctar) = \frac{|link(ng, ctar)|}{|inlink(ctar)|} \quad (20)$$

$$RatioAnchor(ng, ctar) = \frac{|link(ng, ctar)|}{|ng \in A|} \quad (21)$$

Target features The target features are indicators of how likely a candidate target page alone would be linked with some anchor text in the collection. To this end we explore features such as counts of the inlinks and outlinks within the candidate target page, as well as the Wikipedia category information associated with it.

N-gram - topic features This type of feature describes the importance of the n-gram within its context, i.e., topic page. One would assume that an n-gram being selected as an anchor text should be somewhat important to the understanding of the whole topic page as well as being content-wise related. Here, we use the TFIDF score of the n-gram and its location within the topic page as an indication of the importance of a n-gram within a topic page.

Topic - target features The topic-target features describe the relatedness between a topic page and a candidate target page. One obvious feature is the similarity between the two pages. In addition, as a candidate target page itself is about a concept, we could measure how important is this concept, or in other words, how well is this concept being expressed in the topic page. We measure it by using the title of the candidate target page as query and calculate the retrieval score against the topic page.

First stage score Once the target ranking has been completed (during the first stage), we can get the ranking score for each candidate target, as well as their ranks. In the second stage, we select the top X candidate targets to construct the candidate links with their corresponding n-grams, where the scores and ranks from the first stage are used as features.

3.3 The LTW Runs

We submitted 5 runs for the LTW task, as specified in Table 5.

Note that we have a heuristic run UvAdR_LTWA2B_03 which does not use a learning method for link ranking, but only uses RankingSVM for target identification. This run serves as a baseline for other machine learning based approaches. The heuristics used in this run, i.e., the *ALR* and *IDF* scores, however, are the features that are most close to the human intuitions, where *ALR* represents how likely a n-gram is involved in a link based on the observation of existing links and *IDF* represents the uncommonness of a n-gram.

4 Conclusions

We have described our approaches and submissions for this year's INEX participation.

For the Entity Ranking track, we submitted 4 runs for the entity ranking and 6 runs for the list completion tasks. Our main focus is on evaluating the effectiveness of our recently proposed entity retrieval framework on the new Wikipedia collection. In addition, we are interested in investigating whether parameter settings learned on prior editions of the track carry over to this year's setting.

As to the Link-the-Twiki track, we submitted 5 runs to the A2B outgoing links detection task. Our main focus is to explore the effectiveness of applying machine learning approaches for the task. Specifically, we experiment with two types of learning approaches, namely classification and learning to rank. On top of that, we also aim to evaluate the learning material for the task, where we use different sets of training data (based on different versions of Wikipedia).

Acknowledgements

This research was supported by the DAESO and DuOMAn projects carried out within the STEVIN programme which is funded by the Dutch and Flemish Governments (<http://www.stevin-tst.org>) under project numbers STE-05-24 and STE-09-12, and by the Netherlands Organisation for Scientific Research (NWO) under project numbers 640.001.501, 640.002.501, 612.066.512, 612.061.814, 612.061.815, 640.004.802.

Bibliography

- [1] K. Balog, W. Weerkamp, and M. de Rijke. A few examples go a long way: constructing query models from elaborate query formulations. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 371–378, New York, NY, USA, 2008. ACM.
- [2] K. Balog, M. Bron, and M. de Rijke. Category-based query modeling for entity search. In *32th European Conference on Information Retrieval (ECIR 2010)*, To appear, 2010.
- [3] S. Fissaha Adafre, M. de Rijke, and E. Tjong Kim Sang. Entity retrieval. In *Recent Advances in Natural Language Processing (RANLP 2007)*. Borovets, Bulgaria, September 2007.

- [4] R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.
- [5] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, New York, NY, USA, 2001. ACM.
- [6] D. Losada and L. Azzopardi. An analysis on document length retrieval trends in language modeling smoothing. *Information Retrieval*, 11(2):109–138, 2008.
- [7] D. Milne and I. H. Witten. Learning to link with wikipedia. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 509–518, New York, NY, USA, 2008. ACM.
- [8] F. Song and W. B. Croft. A general language model for information retrieval. In *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321, New York, NY, USA, 1999. ACM.
- [9] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004.

N-gram features	
Length(ng)	Number of words contained in the n-gram
IDF(ng)	The IDF score of the n-gram
ALR(ng)	The ALR score of the n-gram, as detailed in Eq. 19
#Cand(ng)	Number of candidate target pages associated with the n-gram
N-gram - target features	
TitleMatch(ng, ctar)	Three values - 2: exact match; 1: partial match(i.e., either the title contains the n-gram, or the n-gram contains the title); 0: not match
RatioLink(ng, ctar)	Link ratio of the n-gram and the candidate target page, see Eq. 20
RatioAnchor(ng, ctar)	Anchor ratio of the n-gram and the candidate target page, see Eq. 21
Ret_uni(ng, ctar)	Retrieval score with unigram model, i.e., BM25 with default parameter settings
Ret_dep(ng, ctar)	Retrieval scores with dependency model, i.e., Markov Random Field model as described in [?]]
Rank_dep(ng, ctar)	The rank of the target page with the dependency retrieval model
Target features	
#Inlinks(ctar)	Number of in-links contained in the candidate target page
#Outlinks(ctar)	Number of out-links contained in the candidate target page
#Categories(ctar)	Number of Wikipedia categories associated with the candidate target page
Gen(ctar)	Generality of the candidate target page as described in [7]
N-gram - topic features	
TFIDF(ng, topic)	The TFIDF score of the n-gram in the topic page
First(ng, topic)	Position of first occurrence of the n-gram in the topic page, normalized by the length of the topic page
Last(ng, topic)	Position of last occurrence of the n-gram in the topic page, normalized by the length of the topic page
Spread(ng, topic)	Distance between first and last occurrence of the n-gram in the topic page normalized by the length of the topic page
Topic-target features	
Sim(ctar, topic)	Cosine similarity between the candidate target page and the topic page
Ret_unigram(ctar, topic)	Retrieval score using the title of the candidate target page as query against the topic page; using BM25 as retrieval model
First stage scores	
score(ng, ctar)	The output of the ranker for the candidate target page given the n-gram
rank(ng, ctar)	The rank of the candidate target page according to the learnt ranker

Table 4. Features used for learning the preference relation, where ng: n-grams; C: collection; ctar: candidate target pages; topic: topic page.

RunID	Description
UvAdR_LTWA2B_01	Binary classification, trained on wiki08 (old and small)
UvAdR_LTWA2B_02	Ranking SVM, trained on wiki08
UvAdR_LTWA2B_03	A heuristic run, combine the ALR and IDF for link ranking, ignore numbers, but using rankingSVM for target ranking
UvAdR_LTWA2B_04	Binary classification, trained on wiki09
UvAdR_LTWA2B_05	ranking SVM, trained on wiki09

Table 5. Submitted runs

University of Waterloo at INEX 2009: Ad Hoc, Book, Entity Ranking, and Link-the-Wiki Tracks

Kelly Y. Itakura and Charles L. A. Clarke

University of Waterloo, Waterloo, ON N2L3G1, Canada,
{yitakura, claclark}@cs.uwaterloo.ca

Abstract. This year, University of Waterloo participated in four tracks; Ad Hoc, Book, Entity Ranking, and Link-the-Wiki tracks. In Ad Hoc and Book tracks, we implemented a variation of Okapi BM25F [5, 15, 18, 20] that gave substantial improvements over the baseline BM25 that ranked first in the previous year [12, 13], during the training and in the preliminary Ad Hoc-focused results. In Entity ranking track, we used redundancy techniques [4] for question answering to retrieve entities. In Link-the-Wiki track, we employed topic-oriented PageRank with KL divergence in addition to the baseline described in [11]. The results of all, but Ad Hoc track is unavailable at this time.

1 Introduction

This year, University of Waterloo participated in four tracks; Ad Hoc, Book, Entity Ranking, and Link-the-Wiki tracks. For all, except the newly-participating Entity Ranking track, our goal this year is to answer the question, “how can we improve our *simple* baseline approaches that *used little XML structure that performed better than any other systems that are more complex?*” Or more bluntly, “What is the meaning of L..., I mean, INEX?”

In INEX 2008, the simple BM25 algorithm scored best for all Ad Hoc tasks, focused, relevant-in-context, and best-in-context [12, 13]. For Book track, even though our BM25 approach performed poorly on book retrieval task, the same approach for page-in-context task performed better than the (only) other participant [12, 14]. Though because of the small number of participants, it is too early to say that our simple BM25 approach is optimal for the Book track tasks, for Ad Hoc track, our approach has been consistently performing well for INEX 2004, 2007, and 2008 [3, 8, 13, 16]. Therefore, this year, while keeping our approaches simple, we incorporated BM25F, BM25 with XML field extension [5, 20]. As it turned out, our approaches, especially Ad Hoc one, were similar to those of Lu et al. [15] and Robertson et al. [18] that use inherited information. However, our experiments are on a much larger scale on a very different corpus, and the tasks are on focused task, rather than on thorough and relevant-in-context tasks, training are at element level as opposed to document level, we used averaged field length as opposed to a document length, we used length normalization at field level

as opposed to at a document level, and we used somewhat different field structure and inheritance. Most importantly, as far as we know, there is no follow-up work on BM25F with inheritance since 2006, but we think there is a room for improvement and especially application in tasks other than Ad Hoc tasks. As a first step of application, we applied BM25F with inheritance in a book page retrieval setting.

In Entity track, our main question is, “How is it different from a water-downed version of Question Answering task?” Therefore, we used redundancy technique for question answering [4], combined with category estimation. We also changed a perspective, and instead of trying to retrieve documents that represent entities as many participants did more or less [6, 22], we extracted entities from passages and then located articles corresponding to the entities.

For Link-the-Wiki track, we saw that the simple statistical approach reproduces the Wikipedia as a ground truth well [9–11] but did not perform well when compared to manual assessments. We think this is due to the fact that the simple statistical approach does not take topicality of anchor phrase into account, and used a modified version of topical PageRank [1, 17].

2 Ad Hoc Track

This year, we only participated in Focused task using CO queries to perform element retrieval. From our previous years’ experience, we expect that the performance of our runs in Focused task would behave very similarly in relevant-in-context and in best-in-context tasks.

We practically submitted two runs (the other two runs had some minor bugs but performed exactly the same), `UWFERBase2` and `UWFERBM25F2`. Our baseline run is `UWFERBase2` that is an element retrieval using BM25 as in the previous years [3, 8, 13, 16]. We used BM25F for another element retrieval run `UWFERBM25F2`.

For both runs, the unit of retrieval is elements. For training set on INEX 2008 collection [7], we used the following elements we used from last year;

```
<p>, <section>, <normallist>, <article>, <body>, <td>, <numberlist>,
<tr>, <table>, <definitionlist>, <th>, <blockquote>, <div>, <li>,
<u>.
```

For test set on INEX 2009 collection [21], we used the following elements that correspond to the elements we used for training.

```
<p>, <sec>, <list>, <article>, <bdy>, <column>, <row>, <table>, <entry>
,<indent>, <ss1>, <ss2>, <ss3>, <ss4>, <ss5>.
```

For both runs, we processed queries by converting each topic title into a disjunction of query terms without negative query terms. We located positions of all query terms and some XML tags using Wumpus [2].

2.1 Baseline BM25 Run

For the baseline, `UWFERBase2`, we scored elements using Okapi BM25 [19]. The score of an element E using Okapi BM25 is defined as follows.

$$s(E) \equiv \sum_{t \in Q} W_t \frac{f_{E,t}(k+1)}{f_{E,t} + k(1-b + b \frac{pl_E}{avgdl})} , \quad (1)$$

where Q is a set of query terms, W_t is an IDF value of the term t in the collection, $f_{E,t}$ is the sum of term frequencies in an element E , pl_E is an element length of E , and $avgdl$ is an average document length in Wikipedia collection to act as a length normalization factor.

We then sorted elements by their scores, and obtained the top elements while removing overlaps.

We tuned our parameters to maximize $iP[0.01]$ on INEX 2008 training set using `inex_eval.jar` provided from INEX. Our tuning method was repeated fixing of one parameter and changing of another until no further change is observed. The parameters $k = 1.8$ and $b = 0.4$ gave $iP[0.6860]$ on INEX 2008 training set, and we used the same set of parameters on INEX 2009 test set.

2.2 Experimental BM25F Run

For each element we listed in Section 2, we constructed two fields, one for “title” and another for “body”. The title field consists of concatenation of an article title and any ancestral and current section titles. The body field contains the rest of the text in the element. Lu et al. [15] and Robertson et al. [18], on the other hand, had separate fields for article title and section title or current title and ancestral titles. Our intuition was that in Wikipedia, section headings do not normally seem to include the article title and therefore, rather vague. We do not know whether our approach is any better or different or not.

For example, in a page titled “Salsa (dance)”¹, one of the section headings is “Basic movements”. In our title field for the section, we fit “Salsa Basic movements”. The body field contains the rest of the section but not the current section title of “Basic movements”.

Unfortunately, because in INEX 2008 collection, we only had one-level section headings, whereas in INEX 2009 collection, we had subsection headings, we only considered the first level of section headings during our test.

Once we collected necessary ingredients, we applied BM25F formula for an element e [5];

$$BM25F(e) = \sum_{t \in q \cap e} \frac{\bar{x}_{e,t}}{K + \bar{x}_{e,t}} W_t ,$$

where q is a query term, $\bar{x}_{e,t}$ is a weighted normalized term frequency that we describe later, and K is a parameter, and W_t is a document-level IDF for a term

¹ [http://en.wikipedia.org/wiki/Salsa_\(dance\)](http://en.wikipedia.org/wiki/Salsa_(dance))

t . The weighted normalized term frequency is obtained by first performing length normalization, on a term frequency $x_{e,f,t}$ of a term t of field f in an element e ,

$$\bar{x}_{e,f,t} = \frac{x_{e,f,t}}{1 + B_f \left(\frac{l_{e,f}}{l_f} - 1 \right)},$$

B_f is a parameter to tune, $l_{e,f}$ is a length of a field f in an element e , l_f is an average field length.

We then multiplied the normalized term frequency $\bar{x}_{e,f,t}$ by field weight W_f ,

$$\bar{x}_{e,t} = \sum_f W_f \cdot \bar{x}_{e,f,t}.$$

Lu et al. [15] and Robertson et al. [18] use a version of BM25F without individual field length normalization [20].

We fixed $K = 1$ and then trained the rest of the four parameters B_{title} , B_{body} , W_{title} , and W_{body} , similarly to our baseline. We created our run with $B_{title} = 0.6$, $B_{body} = 0.6$, $W_{title} = 4$, and $W_{body} = 1.2$ that gave $ip[0.01] = 0.7131$ during training.

2.3 Preliminary Results

Preliminary results showed that **UWFERBM25F2** ranked first with $ip[0.01] = 0.6333$. The score of the second ranked run from University Pierre et Marie Curie is 0.6141, and our other run **UWFERBase2** ranked 12th with $ip[0.01] = 0.5940$, so it seems that our experimental BM25F-based run not only improves the baseline BM25 run substantially, but also seems to outperform other participants. We speculate that if we train on INEX 2009 collection with multi-level section headings and test on the same set, the improvement would be more prominent.

An interesting observation is that even though our BM25F run performs excellent when evaluated at the element level, when one evaluate the same run at the document level, the performance becomes much poorer, and poorer than the baseline run.

The best run evaluated at the document level is LIG's **LIG-2009-focused-1F** run with MAP = 0.3569. Our baseline run ranks at 25th with MAP = 0.3267 and our BM25F run ranks at 34th with MAP = 0.3017. Similar, but not so radical observation applies for our 2008 focused runs. The only reason we could think so far is that a lot of relevant elements we return come from the same documents, therefore, when evaluating at a document level, the number of distinct documents returned is lower, especially for BM25F run, therefore obtaining lower MAP value.

3 Book Track

This year, we only participated in Focused Book Search task. As in the Ad Hoc track, our objective is if BM25F-based element retrieval run has any significance

over the simple BM25-based element retrieval run that performed well last year. Therefore, we submitted one run, `UWBaseline` (`UWBaseline2` fixed minor bugs) as a baseline BM25 element retrieval run, and another run, `UWBM25F` as an experimental BM25F element retrieval run. Additionally, we submitted a manual run, `UWManual` that is the same as `UWBM25F` run, except some of the query phrases that deemed unimportant are manually deleted. We used the same parameters obtained for `UWBM25F` on `UWManual`.

Both `UWBaseline` and `UWBM25F` runs are element retrieval run where the unit of element is a page. This is a difference from the last year, where we considered finer element types such as `<regeon>` and `<section>`. Most tags such as `<marker>` and `<regeon>` did not seem useful, and `<line>` tag is below the minimum unit of retrieval in the task specification. Thus it would have been better to work on both `<page>` and `<section>` elements, but since last year's results were converted to a page-level result, we decided to work on page-level element retrieval this year.

For `UWBaseline` run, each page is scored using Equation 1. We accidentally scored and returned our fictitious `<title>` elements that contained MARC record used for our other runs, but we do not think it would affect the result much.

For `UWBM25F` run, we created two fields for each page; one containing the entire MARC record, and the other containing the actual text in the page. Our original idea was to add the title of the book to each page; however, by examining MARC record, we thought that keywords contained in some of the records may be useful. In the future, we would like to include only the useful information from the records by discarding author name and publishers, and translating LC classification code into phrases. One we obtained fields, we scored pages by Equation 2.2

For both runs, after scoring, we sorted the books by the maximum page scores, and within each book, we sorted pages by the scores. For training, we returned the top 1500 book-page pairs, and used `trec.eval.8.1` with `qrels` from INEX 2008, `averaged-relevance/qrel-alltopics.txt.xpath`, to maximize MAP. For `UWBaseline` run, we obtained $k = 200$ and $b = 1$ with $MAP = 0.0219$. For `UWBM25F` run, we created our run with $B_{title} = 0.2$, $B_{body} = 0.6$, $W_{title} = 2$, and $W_{body} = 0.4$, with a fixed $K = 1$ that gave $MAP = 0.0356$ during training. For testing, we returned the top 1000 books regardless of the number of pages each book contained.

4 Entity Ranking Track

This year marks our first participation in Entity Ranking track. Our objective is to see if a factoid question answering approach can be effectively used as an entity ranking task. Therefore, instead of searching for Wikipedia pages that are entities that fits the topic title, we treated titles as questions, searched for answers in passages within Wikipedia pages, and found the Wikipedia pages that correspond to the answers.

We use redundancy technique of Clarke et al. [4] to extract answers. The technique first scores passages using self-information of probability that an extent contain a query term;

$$\sum_{t \in T} \log \frac{N}{f_t} - |T| \log l ,$$

where N is the corpus length and f_t is a term frequency in the corpus. The technique then retrieves top k passages and expand the passages to size w around the center. The candidates answers are then ranked according to the number of distinct top k passages that contain the term.

4.1 Entity Ranking Task

We submitted one entity ranking run, `1.Waterloo_ER_TC.qap`. The main difference between this run and the question answering technique of Clarke et al. [4] is in the listing of candidate topics from the top k passages.

We first extracted all titles and the categories listed at the beginning of the articles in the Wikipedia corpus, call it *wiki_titles* and *wiki_categories* respectively. We also extracted topic titles and the categories from the topic file, call them *topic_titles* and *topic_categories* respectively.

We collected the highest scoring passages from each article in the corpus using *topic_titles* according to Equation 4. We then expanded the passages around the center to 1000 terms. Call this new set of passages *passage_ER*.

To list the candidate terms from *passage_ER*, we need to extract terms

1. that have corresponding Wikipedia articles
2. whose categories loosely fall under those in *topic_categories*

Therefore, we queried *wikipedia_categories* using *topic_categories* as query terms with BM25. We collected the top 10000 highest scoring Wikipedia pages that correspond to the top 10000 categories including duplicates, retrieved from *wikipedia_categories*. Now the list of the candidate terms are limited to the titles of these Wikipedia pages.

We ranked the candidate terms by the number of passages in *passage_ER* that contain the terms. From the list, we removed the topic titles and duplicates to create the final submission.

4.2 List Completion Task

We submitted one run, `1.Waterloo_LC_TE`. The major difference between `1.Waterloo_ER_TC.qap` and Clarke et al.'s work [4] is how to estimate the categories of entities to retrieve.

We extracted all titles and the categories listed at the beginning of the articles in the Wikipedia corpus, *wiki_titles* and *wiki_categories*. For each topic, we extracted topic titles and examples, and merged each topic title with its examples to make the final *topic_titles*.

Using the merged *topic_titles*, we collected the highest scoring passages from each article according to Equation 4. We then expanded the passages around the center to 1000 terms. Call this new set of passages *passage_LC*.

We estimated categories of retrieval by the union of all categories of examples according to *wiki_categories*. Call this new set of categories *ex_categories*.

We queried *wikipedia_categories* using *ex_categories* as query terms with BM25. We collected the top 10000 highest scoring Wikipedia pages that correspond to the top 10000 categories with duplicates, retrieved from *wikipedia_categories*. The list of the candidate terms are limited to the titles of these Wikipedia pages.

As in the case of Entity Ranking task, we ranked the candidate terms by the number of passages in *passage_LC* that contain the terms. From the list, we removed the topic titles, example titles, and duplicates to create the final submission.

5 Link-the-Wiki Track

This year, our goal is to use link analysis to overcome the problem of the simple statistical approach [11]. We mainly submitted runs for anchor-to-BEP topics, mostly at file level, but also two runs at anchor-to-BEP level. We also submitted one run for Link-Te-Ara-to-the-Wiki task using our baseline.

5.1 Link-the-Wiki Anchor-to-BEP File Level

We submitted five runs, `UWBaseline_F2FonA2B`, `UWKPR_F2FonA2B`, `UWKPRFlip_F2FonA2B`, `UWTPR_F2FonA2B`, and `UWKPR+_F2FonA2B`.

Baseline: `UWBaseline_F2FonA2B` Our implementation of the outgoing link discovery for baseline run is the same as in INEX 2007/2008 [11, 12].

After removing topic files from the Wikipedia corpus, we parsed the corpus to obtain the following ratio, γ .

$$\gamma = \frac{\# \text{ of files that has a link from anchor } a \text{ to a file } d}{\# \text{ of files in which } a \text{ appears at least once}}$$

For each topic file, we located the anchor phrases, and returned the most frequent destination file for each of the top 250 anchor phrases by γ .

For the incoming link discovery, we scored each article in the Wikipedia corpus using topic titles and returned the top 250 articles.

KL-PageRank: `UWKPR_F2FonA2B` Our baseline approach does not take into account the context of anchor phrases. For example, “United States” may be frequently linked, but should it be linked from an unrelated topic such as “olive oil”?

We employed PageRank [1] and topical PageRank [17] to balance the frequency of linkage with topicality of the pages to topic files.

For outgoing links, We created a link graph without topic files and then added the links from the topic files according to the outgoing results from the baseline run, `UWBaseline_F2FonA2B`. We obtained PageRank values for each node in the graph. In order to compute topical PageRank, we set the jump vector to be uniform, except for the entry corresponding to the topic file that is set to an arbitrary large number, $1/2$. Once we obtained both PageRank $g(p)$ and topical PageRank values $f(p)$ for each page p , we computed the contribution of point-wise K-L divergence values for each page for each topic as follows:

$$\tau(p) = f(p) \log \frac{f(p)}{g(p)} .$$

We returned the top 250 destination files according to τ .

For incoming links, we reversed the link graph obtained from outgoing links and using the same topical PageRank values from the outgoing links, we computed the contribution of point-wise K-L divergence. We returned the top 250 files according to τ .

KPR-Flip: `UWKPRFlip_F2FonA2B` This run tries to answer the question: Should there be an outgoing link from a page “Yoga” to “Hatha Yoga”? But should not there also be an incoming links from “Hatha Yoga” to “Yoga”?

Because at file level, it is unclear if there is any difference between the set of incoming links and the set of outgoing links for a topic file, this run returned the set of outgoing links from `UWKPR_F2FonA2B` as incoming links and the set of incoming links from the same run as outgoing links.

Topical PageRank-only: `UWTPR_F2FonA2B` This run tries to answer the question: How much of the links should come from topically related files?

Because in our `UWKPR_F2FonA2B` run, we added point-wise K-L divergence of Topical PageRank with PageRank values to balance topicality with popularity, in this run, we returned the set of outgoing links and incoming links from `UWKPR_F2FonA2B` run, right after obtaining Topical PageRank, but before applying K-L divergence.

KPR Run with Filtering: `UWKPR+_F2FonA2B` It is natural to assume that a file that has a link to a topic file contain the topic title within the file more or less. Therefore, in this run, for incoming links, we filtered the result of `UWKPR_F2FonA2B` to include only those files that contain the topic titles.

We may think that this might also hold true for the outgoing links; an anchor phrase must exist for a file to be considered to be a destination from the topic files. We, however, left the outgoing links the same as in `UWKPR_F2FonA2B` because we were not sure if this holds true for outgoing links at a file level.

5.2 Link-the-Wiki Anchor-to-BEP Passage Level

We submitted two runs, `UWBaseline_A2B` and `UWKPR_A2B`.

The baseline run `UWBaseline_A2B` implements the file-level baseline run `UWBaseline_F2FonA2B` at anchor-to-BEP level. Specifically, for outgoing links, we returned the offset and length of anchor phrases in the topic files and the BEP is set to the titles of destination files. We restricted to return only the most frequent destination per anchor in order to compare our results against `UWKPR_A2B`, which is a KPR-based run that only returns one destination file per anchor phrase. For incoming links, both anchor phrases and destination files are set to the titles of the incoming files and the topic files.

The KPR based run `UWKPR_A2B` implements `UWKPR_F2FonA2B` at anchor-to-BEP level. For outgoing links, we performed disjunctive queries using Equation 4 into the topic files, the titles of top destination files according to the values of K-L divergence. Once we obtained the positions of the destination titles in the topic files, we returned those as anchor phrases with BEP being the titles of the destination files. We only returned one destination file per anchor phrase. For incoming links, we first screened the set of incoming files returned by K-L divergence to only those that contain the topic titles. We then set both the anchor phrase and BEP to be the titles of the incoming and topic files.

5.3 Link-Te-Ara-to-the-Wiki

We only submitted one baseline run at Anchor-to-BEP level, `Teara_Baseline_A2B`. Due to resource constraints, we only computed γ for those anchor phrases that occurred in more than 100 documents in Wikipedia corpus. Once we have a list of anchor phrases-destination pairs, we queried those phrases in TeAra corpus. As is the case with our other Anchor-to-BEP runs, we only returned the most frequent destination per anchor phrase.

6 Conclusions

This year, we continued to participate in Adhoc, Book, and Link-the-Wiki tracks. We also participated for the first time, Entity Ranking track. Our theme for the three continuing tracks is how to improve over our simple baseline that performed well from the previous years. For Ad Hoc track, we implemented BM25F with title and text fields. The preliminary results showed that it gives a noticeable improvements over the baseline as well as over the other participants' runs. For Book track, we took the same BM25F idea, but used MARC record as a title field and a page as a body field. The performance of the run during the training indicates that this approach is also promising over the baseline BM25 approach. In Link-the-Wiki track, we considered topicality of links that were missing from our simple statistical baseline algorithm through Topical PageRank and KL-divergence. For Entity Ranking track, instead of searching directly for Wikipedia pages, we looked for answers in passages and then ranked those entities that appear in the passages according to the frequency.

Future work includes finding an effective way of applying BM25F in XML corpus, as well as using extra-book information to fill in the fields.

References

1. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.
2. S. Büttcher. the Wumpus Search Engine. Accessible at <http://www.wumpus-search.org>, 2007.
3. C. L. Clarke and P. L. Tilker. Multitext experiments for INEX 2004. In *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2004)*, pages 85–87, 2005.
4. C. L. A. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*, pages 358–365, 2001.
5. N. Craswell, H. Zaragoza, and S. Robertson. Microsoft Cambridge at TREC 14: Enterprise track. In *Proceedings of the 14th Text REtrieval Conference*, 2005.
6. G. Demartini, A. P. Vries, T. Iofciu, and J. Zhu. Overview of the INEX 2008 entity ranking track. In *Advances in Focused Retrieval: Seventh International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2008)*, pages 243–252, 2009.
7. L. Denoyer and P. Gallinari. The Wikipedia XML corpus. *SIGIR Forum*, 40(1):64–69, 2006.
8. N. Fuhr, J. Kamps, M. Lalmas, S. Malik, and A. Trotman. Overview of the INEX 2007 Ad Hoc Track. In *Focused Access to XML Documents: Sixth International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2007)*, pages 1–23, 2007.
9. W. C. Huang, S. Geva, and A. Trotman. Overview of the INEX 2008 Link the Wiki Track. In *Advances in Focused Retrieval: Seventh International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX 2008)*, pages 314–325, 2009.
10. K. Y. Itakura and C. L. A. Clarke. From passages into elements in XML retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007) Workshop on Focused Retrieval*, pages 17–27, 2007.
11. K. Y. Itakura and C. L. A. Clarke. University of Waterloo at INEX2007: Adhoc and Link-the-Wiki Tracks. In *Focused Access to XML Documents: Sixth International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2007)*, pages 417–425, 2008.
12. K. Y. Itakura and C. L. A. Clarke. University of Waterloo at INEX 2008: Adhoc, Book, and Link-the-Wiki Tracks. In *Advances in Focused Retrieval: Seventh International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX 2008)*, pages 132–139, 2009.
13. J. Kamps, S. Geva, A. Trotman, A. Woodley, and M. Koolen. Overview of the INEX 2008 Ad Hoc Track. In *Advances in Focused Retrieval: Seventh International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX 2008)*, pages 1–28, 2009.

14. G. Kazai, A. Doucet, and M. Landoni. Overview of the INEX 2008 Book Track. In *Advances in Focused Retrieval: Seventh International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX 2008)*, pages 106–123, 2009.
15. W. Lu, S. Robertson, and A. MacFarlane. Field-weighted XML retrieval based on BM25. In *Advances in XML Information Retrieval and Evaluation: Fourth International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2005)*, pages 161–171, 2006.
16. S. Malik, M. Lalmas, and N. Fuhr. Overview of INEX 2004. In *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2004)*, pages 1–15, 2005.
17. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.
18. S. Robertson, W. Lu, and A. MacFarlane. XML-structured documents: Retrievable units and inheritance. In *Proceedings of the Seventh International Conference on Flexible Query Answering Systems (FQAS 2006)*, pages 121–132, 2006.
19. S. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC-7: Automatic ad hoc, filtering, vlc and interactive track. In *Proceedings of the Seventh Text REtrieval Conference*, pages –, 1998.
20. S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management (CIKM 2007)*, pages 42–49, 2004.
21. R. Schenkel, F. Suchanek, and G. Kasneci. YAWN: A semantically annotated Wikipedia XML corpus. In A. Kemper, H. Schöning, T. Rose, M. Jarke, T. Seidl, C. Quix, and C. Brochhaus, editors, *12. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web*, volume 103 of *Lecture Notes in Informatics*, pages 277–291, Aachen, Germany, 2007. Gesellschaft für Informatik.
22. A. P. Vries, A.-M. Vercoustre, J. A. Thom, N. Craswell, and M. Lalmas. Overview of the INEX 2007 entity ranking track. In *Focused Access to XML Documents: Sixth International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2007)*, pages 245–251. Springer-Verlag, 2008.

University of Amsterdam at INEX 2009: Ad hoc, Book and Entity Ranking Tracks

Marijn Koolen¹, Rianne Kaptein¹, and Jaap Kamps^{1,2}

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

² ISLA, Faculty of Science, University of Amsterdam

Abstract. In this paper we describe our participation in INEX 2009 in the Ad Hoc Track, the Book Track, and the Entity Ranking Track. In the Ad Hoc track we investigate focused link evidence, using only links from retrieved sections. The new collection is not only annotated with Wikipedia categories, but also with YAGO/WordNet categories. We explore how we can use both types of category information, in the Ad Hoc Track as well as in the Entity Ranking Track. Results in the Ad Hoc Track show Wikipedia categories are more effective than WordNet categories, and Wikipedia categories in combination with relevance feedback lead to the best results.

1 Introduction

In this paper, we describe our participation in the INEX 2009 Ad Hoc, Book, and Entity Ranking Tracks. Our aims for this year were to familiarize ourselves with the new Wikipedia collection, to continue the work from previous years, and to explore the opportunities of using category information, which can be in the form of Wikipedia’s categories, or the enriched YAGO/WordNet categories.

The rest of the paper is organized as follows. First, Section 2 describes the collection and the indexes we use. Then, in Section 3, we report our runs and results for the Ad Hoc Track. Section 4 briefly discusses our Book Track experiments. In Section 5, we present our approach to the Entity Ranking Track. Finally, in Section 6, we discuss our findings and draw preliminary conclusions.

2 Indexing the Wikipedia Collection

In this section we describe the index that is used for our runs in the adhoc and the entity ranking track, as well as the category structure of the collection. The collection is based, again, on the Wikipedia but substantially larger and with longer articles. The original Wiki-syntax is transformed into XML, and each article is annotated using “semantic” categories based on YAGO/Wikipedia. We used Indri [14] for indexing and retrieval.

2.1 Indexing

Our indexing approach is based on our earlier work [1, 3, 5, 11, 12, 13].

- *Section index*: We used the <section> element to cut up each article in sections and indexed each section as a retrievable unit. Some articles have a leading paragraph not contained in any <section> element. These leading paragraphs, contained in <p> elements are also indexed as retrievable units. The resulting index contains no overlapping elements.
- *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, and terms were stemmed using the Krovetz stemmer. Queries are processed similar to the documents. In the ad hoc track 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.

2.2 Category Structure

A new feature in the new Wikipedia collection is the assignment of WordNet labels to documents [10]. The WordNet categories are derived from Wikipedia categories, but are designed to be conceptual. Categories for administrative purposes, such as ‘Article with unsourced statements’, categories yielding non-conceptual information, such as ‘1979 births’ and categories that indicate merely thematic vicinity, such as ‘Physics’, are not used for the generation of WordNet labels, but are excluded by hand and some shallow linguistic parsing of the category names. WordNet concepts are matched with category names and the category is linked to the most common concept among the WordNet concepts. It is claimed this simple heuristic yields the correct link in the overwhelming majority of cases.

A second method which is used to generate WordNet labels, is based on the basis of information in lists. For example, If all links but one in a list point to pages belonging to a certain category, this category is also assigned to the page that was not labelled with this category. This is likely to improve the consistency of annotation, since annotation in Wikipedia is largely a manual effort.

3 Ad Hoc Track

For the INEX 2009 Ad Hoc Track we aim to investigate:

- Focused link evidence. Use local link degrees as evidence of topical relevance. Instead of looking at all local links between the top 100 retrieved articles, we consider only the links occurring in the retrieved elements. A link from article *A* to article *B* occurring in a section of article *A* that is not retrieved is ignored. This link evidence is more focused on the search topic and possibly leads to less infiltration.

- Wikipedia and WordNet categories. The new INEX Wikipedia collection has markup in the form of YAGO elements including WordNet categories. Most Wikipedia articles are manually categorised by the Wikipedia contributors. The category structure can be used to generate category models to promote articles that belong to categories that best match the query. We aim to directly compare the effectiveness of category models based on the Wikipedia and WordNet categorisations for improving retrieval effectiveness.

We will first describe our approach and the official runs, and finally per task, we present and discuss our results.

3.1 Approach

We have four baseline runs based on the indexes described in the previous section:

Article : run on the article index with linear length prior and linear smoothing $\lambda = 0.15$.

Section : run on the section index with linear length prior and linear smoothing $\lambda = 0.15$.

Article RF : run on the article index with blind relevance feedback, using 50 terms from the top 10 results.

Section RF : run on the section index with blind relevance feedback, using 50 terms from the top 10 results.

All our official runs for all four tasks are based on these runs. To improve these baselines, we explore the following options.

Category distance : We determine two target categories for a query based on the top 20 results. We select the two most frequent categories to which the top 20 results are assigned and compute a category distance score using parsimonious language models of each category. This technique was successfully employed on the INEX 2007 Ad hoc topics by Kaptein et al. [7]. In the new collection, there are two sets of category labels. One based on the *Wikipedia* category structure and one based on the *WordNet* category labels.

CAS filter : For the CAS queries we extracted from the CAS title all semantic target elements, identified all returned results that contain a target element in the xpath and ranked them before all other results by adding a constant c to the score per matching target element. Other than that, we keep the ranking in tact. A result that matches two target elements gets $2c$ added to its score, while a result matching one target element gets $1c$ added to its score. In this way, results matching n target elements are ranked above results matching $n - 1$ target elements. This is somewhat similar to co-ordination level ranking of content-only queries. Syntactic target elements like `<article>`, `<sec>`, `<p>` and `<category>` are ignored.

Link degrees : Both incoming and outgoing link degrees are useful evidence in identifying topical relevance [4, 9]. We use the combined $indegree(d) + outdegree(d)$ as a document “prior” probability $P_{link}(d)$. This is easy to

incorporate in a standard language model. Of course, local link evidence is not query-independent, so $P_{link}(d)$ is not an actual *prior* probability. We note that for runs where we combine the article or section text score with a category distance score, we get a different score distribution. With these runs we use the link evidence more carefully by taking the log of the link degree as $P_{link}(d)$.

Focused Link degrees : We also constructed a focused local link graph based on the retrieved elements of the top 100 articles. Instead of using all links between the top 100 articles, we only use the outgoing links from sections that are retrieved for a given topic. The main idea behind this is that link anchors appearing closer to the query terms are more closely related to the search topic. Thus, if for an article a_i in the top 100 articles only section s_j is retrieved, we use only the links appearing in section s_j that point to other articles in the top 100. This local link graph is more focused on the search topic, and potentially suffers less from infiltration of important but off-topic articles. Once the focused local link graph is constructed, we count the number of incoming + outgoing links as the focused link prior $P_{foclink}(d)$.

Article ranking : based on [3], we use the article ranking of an article index run and group the elements returned by a section index run as focused results.

Cut-off(n) : When we group returned elements per article for the Relevant in Context task, we can choose to group all returned elements of an article, or only the top ranked elements. Of course, further down the results list we find less relevant elements, so grouping them with higher ranked elements from the same article might actually hurt precision. We set a cut-off at rank n to group only the top returned elements by article.

3.2 Runs

Combining the methods described in the previous section with our baseline runs leads to the following official runs.

For the Thorough Task, we submitted two runs:

UamsTAdbi100 : an article index run with relevance feedback. The top 100 results are re-ranked using the link degree prior $P_{link}(d)$. This run was submitted to the Thorough task.

UamsTSdbi100 : a section index run with relevance feedback. We cut off the results list at rank n and re-rank the focused results of the top 100 articles using the link prior $P_{link}(d)$. This run was submitted to the Thorough task.

For the Focused Task, we submitted two runs:

UamsFSdbi100CAS : a section index run combined with the Wikipedia category distance scores. The results of the top 100 articles are re-ranked using the link degree prior. Finally, the CAS filter is applied to boost results with target elements in the xpath. This run was submitted to the Focused task.

UamsFSs2dbi100CAS : a section index run combined with the Wikipedia category distance scores. The results of the top 100 articles are re-ranked

using the focused link degree prior $P_{foclink}(d)$. Finally, the CAS filter is applied to boost results with target elements in the xpath. This run was submitted to the Focused task.

For the Relevant in Context Task, we submitted two runs:

UamsRSCMACMdbi100 : For the article ranking we used the article text score combined with the manual category distance score as a baseline and re-ranked the top 100 articles with the log of the local link prior $P_{link}(d)$. The returned elements are the top results of a combination of the section text score and the manual category distance score, grouped per article. This run was submitted to the Relevant in Context task.

UamsRSCWACWdbi100 : For the article ranking we used the article text score combined with the WordNet category distance score as a baseline and re-ranked the top 100 with the log of the local link prior $P_{link}(d)$. The returned elements are the top results of a combination of the section text score and the wordnet category distance score, grouped per article. This run was submitted to the Relevant in Context task.

For the Best in Context Task, we submitted two runs:

UamsBAfbCMdbi100 : an article index run with relevance feedback combined with the Wikipedia category distance scores, using the local link prior $P_{link}(d)$ to re-rank the top 100 articles. The Best-Entry-Point is the start of the article. This run was submitted to the Best in Context task.

UamsBAfbCMdbi100 : a section index run with relevance feedback combined with the Wikipedia category distance scores, using the focused local link prior $P_{foclink}(d)$ to re-rank the top 100 articles. Finally, the CAS filter is applied to boost results with target elements in the xpath. The Best-Entry-Point is the start of the article. This run was submitted to the Best in Context task.

3.3 Thorough Task

Results of the Thorough Task can be found in Table 1. We make the following observations:

- Standard relevance feedback improves upon the baseline. The Wikipedia category distances are even more effective. The WordNet category distances are somewhat less effective, but still lead to improvement for MAiP.
- Combining relevance feedback with the WordNet categories hurts performance, whereas combining feedback with the Wikipedia categories improves MAiP. However, for early precision, the Wikipedia categories without feedback perform better.
- The link prior has a negative impact on performance of article level runs. The official run *UamsTAdbi100* is based on the *Article RF* run, but with the top 100 articles re-ranked using the local link prior.

Table 1: Results for the Ad Hoc Track Thorough Task (runs labeled “UAMS” are official submissions)

Run id	MAiP	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]
UamsTAdbi100	0.2676	0.5350	0.5239	0.4968	0.4712
UamsTSdbi100	0.2139	0.5022	0.4915	0.4639	0.4400
Article	0.2814	0.5938	0.5880	0.5385	0.4981
Article RF	0.2967	0.6082	0.5948	0.5552	0.5033
Article + Cat(Wiki)	0.2991	0.6156	0.6150	0.5804	0.5218
Article + Cat(WordNet)	0.2841	0.5600	0.5499	0.5203	0.4950
Article RF + Cat(Wiki)	0.3011	0.6006	0.5932	0.5607	0.5177
Article RF + Cat(WordNet)	0.2777	0.5490	0.5421	0.5167	0.4908
$(Article + CAT(Wiki)) \cdot P_{link}(d)$	0.2637	0.5568	0.5563	0.4934	0.4662
$(Article + CAT(WordNet)) \cdot P_{link}(d)$	0.2573	0.5345	0.5302	0.4924	0.4567
Section	0.1403	0.5525	0.4948	0.4155	0.3594
Section RF	0.1493	0.5761	0.5092	0.4296	0.3623
Section + Cat(Wiki)	0.1760	0.6147	0.5667	0.5012	0.4334
Section + Cat(WordNet)	0.1533	0.5474	0.4982	0.4506	0.3831
Section RF + Cat(Wiki)	0.1813	0.5819	0.5415	0.4752	0.4186
Section RF + Cat(WordNet)	0.1533	0.5356	0.4794	0.4201	0.3737

Table 2: Results for the Ad Hoc Track Focused Task (runs labeled “UAMS” are official submissions)

Run id	MAiP	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]
UamsFSdbi100CAS	0.1726	0.5567	0.5296	0.4703	0.4235
UamsFSs2dbi100CAS	0.1928	0.6328	0.5997	0.5140	0.4647
Section	0.1403	0.5525	0.4948	0.4155	0.3594
Section RF	0.1493	0.5761	0.5092	0.4296	0.3623
Section + Cat(Wiki)	0.1760	0.6147	0.5667	0.5012	0.4334
Section RF + Cat(Wiki)	0.1813	0.5819	0.5415	0.4752	0.4186
Article + Cat(Wiki)	0.2991	0.6156	0.6150	0.5804	0.5218
Article RF + Cat(Wiki)	0.3011	0.6006	0.5932	0.5607	0.5177
UamsRSCMACMdbi100	0.2096	0.6284	0.6250	0.5363	0.4733
UamsRSCWACWdbi100	0.2132	0.6122	0.5980	0.5317	0.4782

- On the section level run it leads to improvement. The official run *UamsTSdbi100* is based on the *Section RF* run, but with the results of the top 100 articles re-ranked using the local link prior. Here, the link prior increases MAiP from 0.1493 to 0.2139.
- Section index runs miss too much relevant information. They perform much worse than the article index runs.

3.4 Focused Task

We have no overlapping elements in our indexes, so no overlap filtering is done. Table 2 shows the results for the Focused Task. We make the following observations:

Table 3: Results for the Ad Hoc Track Relevant in Context Task (runs labeled “Uams” are official submissions)

Run id	MAgP	gP[5]	gP[10]	gP[25]	gP[50]
UamsRSCMACMdbi100	0.1771	0.3192	0.2794	0.2073	0.1658
UamsRSCWACWdbi100	0.1678	0.3010	0.2537	0.2009	0.1591
Article	0.1775	0.3150	0.2773	0.2109	0.1621
Article RF	0.1880	0.3498	0.2956	0.2230	0.1666
Article + Cat(Wiki)	0.1888	0.3393	0.2869	0.2271	0.1724
Article + Cat(WordNet)	0.1799	0.2984	0.2702	0.2199	0.1680
Article RF + Cat(Wiki)	0.1950	0.3528	0.2979	0.2257	0.1730
Article RF + Cat(WordNet)	0.1792	0.3200	0.2702	0.2180	0.1638

- The runs shown are the same as those for the Thorough task. Since the measures used are also the same, the results are also the same. The Wikipedia categories are very effective in improving performance of both the article and section index runs.
- The official Focused runs *UamsFSdbi100CAS* and *UamsFSs2dbi100CAS* are the link prior and focused link prior versions of the *Section + Cat(Wiki)* run. Both runs are also CAS filtered. The document level link degrees hurt performance, while the focused link degrees improve performance.
- The *Article + Cat(Wiki)* run has a slightly lower iP[0.00] than the official *UamsFSs2dbi100CAS*, but a somewhat higher iP[0.01]. The section index is less effective for the Focused task than the article index.
- For comparison, we also show the official Relevant in Context run *UamsRSCMACMdbi100*, which uses the same result elements as the *Section + Cat(Wiki)* run, but groups them per article and uses the $(Article + Cat(Wiki)) \cdot P_{link}(d)$ run for the article ranking. This improves the precision at iP[0.01]. The combination of the section run and the article run gives the best performance.

3.5 Relevant in Context Task

For the Relevant in Context Task, we group result per article. Table 3 shows the results for the Relevant in Context Task. We make the following observations:

- A simple article level run is just as effective for the Relevant in Context task as the much more complex official runs, which uses the $Article + Cat(Wiki) \cdot \log(P_{link}(d))$ run for the article ranking, and the *Section + Cat(Wiki)* run for the top 1500 sections.
- Both relevance feedback and category distance improve upon the baseline article run. Combining relevance feedback with the Wikipedia category distance gives the best results.
- The WordNet categories again hurt performance of the relevance feedback run.

Table 4: Results for the Ad Hoc Track Best in Context Task (runs labeled “Uams” are official submissions)

Run id	MAgP	gP[5]	gP[10]	gP[25]	gP[50]
UamsBAfbCMdbi100	0.1543	0.2604	0.2298	0.1676	0.1478
UamsBSfbCMs2dbi100CASart1	0.1175	0.2193	0.1838	0.1492	0.1278
UamsTAdbi100	0.1601	0.2946	0.2374	0.1817	0.1444
Article	0.1620	0.2853	0.2550	0.1913	0.1515
Article RF	0.1685	0.3203	0.2645	0.2004	0.1506
Article + Cat(Wiki)	0.1740	0.2994	0.2537	0.2069	0.1601
Article + Cat(WordNet)	0.1670	0.2713	0.2438	0.2020	0.1592
Article RF + Cat(Wiki)	0.1753	0.3091	0.2625	0.2001	0.1564
Article RF + Cat(WordNet)	0.1646	0.2857	0.2506	0.1995	0.1542

3.6 Best in Context Task

The aim of the Best in Context task is to return a single result per article, which gives best access to the relevant elements. Table 4 shows the results for the Best in Context Task. We make the following observations:

- Same patterns. Relevance feedback helps, so do Wikipedia and WordNet categories. Wikipedia categories are more effective than relevance feedback, WordNet categories are less effective. Wikipedia categories combined with relevance feedback gives further improvements, WordNet combined with feedback gives worse performance than feedback alone. Links hurt performance. Finally, the section index is much less effective than the article index.

4 Book Track

In the INEX 2009 Book Track we participated in the Book Retrieval and Focused Book Search tasks. Continuing our efforts of last year, we aim to find the appropriate level of granularity for Focused Book Search. The BookML markup has XML elements on the page level. In the assessments of last year, relevant passages often cover multiple pages [8]. With larger relevant passages, query terms might be spread over multiple pages, making it hard for a page level retrieval model to assess the relevance of individual pages.

Can we better locate relevant passages by considering larger book parts as retrievable units? One simple option is to divide the whole book in sequences of n pages. Another approach would be to use the logical structure of a book to determine the retrievable units. The INEX Book corpus has no explicit XML elements for the various logical units of the books, so as a first approach we divide each book in sequences of pages.

Book index : each whole book is indexed as a retrievable unit.

Page index : each individual page is indexed as a retrievable unit.

5-Page index : each sequence of 5 pages is indexed as a retrievable unit. That is, pages 1-5, 6-10, etc., are treated as text units.

We submitted six runs in total: two for the Book Retrieval (BR) task and four for the Focused Book Search (FBS) task. The 2009 topics consist an overall topic statement and one or multiple sub-topics. In total, there are 16 topics and 37 sub-topics. The BR runs are based on the 16 overall topics. The FBS runs are based on the 37 sub-topics.

- Book** : a standard Book index run. Up to 1000 results are returned per topic.
- Book RF** : a Book index run with Relevance Feedback (RF). The initial queries are expanded with 50 terms from the top 10 results.
- Page** : a standard Page index run.
- Page RF** : a Page index run with Relevance Feedback (RF). The initial queries are expanded with 50 terms from the top 10 results.
- 5-page** : a standard 5-Page index run.
- 5-Page RF** : a 5-Page index run with Relevance Feedback (RF). The initial queries are expanded with 50 terms from the top 10 results.

At the time of writing, no relevance assessments have been made. Therefore we cannot yet provide any evaluation results.

5 Entity Ranking

In this section, we describe our approach to the Entity Ranking Track. Our goals for participation in the entity ranking track are to refine last year’s entity ranking method, which proved to be quite effective, and to explore the opportunities of the new Wikipedia collection. The most effective part of our entity ranking approach last year was combining the documents score with a category score, where the category score represents the distance between the document categories and the target categories. We do not use any link information, since last year this only lead to minor improvements [6].

5.1 Category information

For each target category we estimate the distances to the categories assigned to the answer entity. The distance between two categories is estimated according to the category titles. Last year we also experimented with a binary distance, and a distance between category contents, but we found the distance estimated using category titles the most efficient and at the same time effective method.

To estimate title distance, we need to calculate the probability of a term occurring in a category title. To avoid a division by zero, we smooth the probabilities of a term occurring in a category title with the background collection:

$$P(t_1, \dots, t_n|C) = \sum_{i=1}^n \lambda P(t_i|C) + (1 - \lambda)P(t_i|D)$$

where C is the category title and D is the entire wikipedia document collection, which is used to estimate background probabilities. We estimate $P(t|C)$ with a

parsimonious model [2] that uses an iterative EM algorithm as follows:

$$\begin{aligned} \text{E-step:} \quad & e_t = t f_{t,C} \cdot \frac{\alpha P(t|C)}{\alpha P(t|C) + (1-\alpha)P(t|D)} \\ \text{M-step:} \quad & P(t|C) = \frac{e_t}{\sum_t e_t}, \text{ i.e. normalize the model} \end{aligned}$$

The initial probability $P(t|C)$ is estimated using maximum likelihood estimation. We use KL-divergence to calculate distances, and calculate a category score that is high when the distance is small as follows:

$$S_{cat}(C_d|C_t) = -D_{KL}(C_d|C_t) = -\sum_{t \in D} \left(P(t|C_t) * \log \left(\frac{P(t|C_t)}{P(t|C_d)} \right) \right)$$

where d is a document, i.e. an answer entity, C_t is a target category and C_d a category assigned to a document. The score for an answer entity in relation to a target category $S(d|C_t)$ is the highest score, or shortest distance from any of the document categories to the target category.

For each target category we take only the shortest distance from any answer entity category to a target category. So if one of the categories of the document is exactly the target category, the distance and also the category score for that target category is 0, no matter what other categories are assigned to the document. Finally, the score for an answer entity in relation to a query topic $S(d|QT)$ is the sum of the scores of all target categories:

$$S_{cat}(d|QT) = \sum_{C_t \in QT} \operatorname{argmax}_{C_d \in d} S(C_d|C_t)$$

A new feature in the new Wikipedia collection is the assignment of YAGO/-WordNet categories to documents as described in Section 2.2. These WordNet categories have some interesting properties for entity ranking. The WordNet categories are designed to be conceptual, and by exploiting list information, pages should be more consistently annotated. In our official runs we have made several combinations of Wikipedia and WordNet categories.

5.2 Pseudo-Relevant Target Categories

Last year we found a discrepancy between the target categories assigned manually to the topics, and the categories assigned to the answer entities. The target categories are often more general, and can be found higher in the Wikipedia category hierarchy. For example, topic 102 with title ‘Existential films and novels’ has as target categories ‘films’ and ‘novels,’ but none of the example entities belong directly to one of these categories. Instead, they belong to lower level categories such as ‘1938 novels,’ ‘Philosophical novels,’ ‘Novels by Jean-Paul Sartre’ and ‘Existentialist works’ for the example entity ‘Nausea (Book).’ In this case the estimated category distance to the target category ‘novels’ will be small, because the term ‘novels’ occurs in the document category titles, but this is not

Table 5: Target Categories

	olympic dinghy sailing classes	Neil Gaiman novels	chess world champions
Assigned	dinghies	novels	chess grandmasters world chess champions
PR	dinghies sailing	comics by Neil Gaiman fantasy novels	chess grandmasters world chess champions
Wikipedia	dinghies sailing at the olympics boat types	fantasy novels novels by Neil Gaiman	chess grandmasters chess writers living people world chess champion russian writers russian chess players russian chess writers 1975 births soviet chess players people from Saint Petersburg
Wordnet	specification types	writing literary composition novel written communication fiction	entity player champion grandmaster writer chess player person soviet writers

always the case. In addition to the manually assigned target categories, we have therefore created a set of pseudo-relevant target categories. From our baseline run we take the top n results, and assign k pseudo-relevant target categories if they occur at least 2 times as a document category in the top n results. Since we had no training data available we did a manual inspection of the results to determine the parameter settings, which are $n = 20$ and $k = 2$ in our official runs. For the entity ranking task we submitted different combinations of the baseline document score, the category score based on the assigned target categories, and the category score based on the pseudo-relevant target categories. For the list completion task, we follow a similar procedure to assign target categories, but instead of using pseudo-relevant results, we use the categories of the example entities. All categories that occur at least twice in the example entities are assigned as target categories.

5.3 Results

Since the runs are not officially evaluated yet, in this section we will only look at the categories assigned by the different methods. In Table 5 we show a few example topics together with the categories as assigned (“Assigned”) by each method. As expected the pseudo-relevant target categories (“PR”) are more

specific than the manually assigned target categories. The number of common Wikipedia categories in the example entities (“Wikipedia”) can in fact be quite long. More categories is in itself not a problem, but also non relevant categories such as ‘1975 births’ and ‘russian writers’ and very general categories such as ‘living people’ are added as target categories. Finally, the WordNet categories (“WordNet”) contain less detail than the Wikipedia categories. Some general concepts such as ‘entity’ are included. With these kind of categories, a higher recall but smaller precision is expected.

6 Conclusion

In this paper we discussed our participation in the INEX 2009 Ad Hoc, Book, and the Entity Ranking Tracks. For the Ad Hoc Track we can conclude focused link evidence outperforms local link evidence on the article level for the Focused Task. Focused link evidence leads to high early precision. Using category information in the form of Wikipedia categories turns out to be very effective, and more valuable than WordNet category information. Since there are no results yet for the Entity Ranking Track and the Book Track, we cannot draw any conclusions about them here.

Acknowledgments Jaap Kamps was supported by the Netherlands Organization for Scientific Research (NWO, grants # 612.066.513, 639.072.601, and 640.001.-501). Rianne Kaptein was supported by NWO under grant # 612.066.513 Marijn Koolen was supported by NWO under grant # 640.001.501.

Bibliography

- [1] K. N. Fachry, J. Kamps, M. Koolen, and J. Zhang. Using and detecting links in Wikipedia. In *Focused access to XML documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2007)*, volume 4862 of *LNCS*, pages 388–403. Springer Verlag, Heidelberg, 2008.
- [2] D. Hiemstra, S. Robertson, and H. Zaragoza. Parsimonious language models for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185. ACM Press, New York NY, 2004.
- [3] J. Kamps and M. Koolen. The impact of document level ranking on focused retrieval. In *Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2008)*, volume 5631 of *LNCS*. Springer Verlag, Berlin, Heidelberg, 2009.
- [4] J. Kamps and M. Koolen. Is wikipedia link structure different? In *Proceedings of the Second ACM International Conference on Web Search and Data Mining (WSDM 2009)*. ACM Press, New York NY, USA, 2009.

- [5] J. Kamps, M. Koolen, and B. Sigurbjörnsson. Filtering and clustering XML retrieval results. In *Comparative Evaluation of XML Information Retrieval Systems: Fifth Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2006)*, volume 4518 of *LNCS*, pages 121–136. Springer Verlag, Heidelberg, 2007.
- [6] R. Kaptein and J. Kamps. Finding entities in Wikipedia using links and categories. In *Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2008)*, volume 5631 of *LNCS*. Springer Verlag, Berlin, Heidelberg, 2009.
- [7] R. Kaptein, M. Koolen, and J. Kamps. Using Wikipedia categories for ad hoc search. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, New York NY, USA, 2009.
- [8] G. Kazai, N. Milic-Frayling, and J. Costello. Towards methods for the collective gathering and quality control of relevance assessments. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 452–459, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: <http://doi.acm.org/10.1145/1571941.1572019>.
- [9] M. Koolen and J. Kamps. What’s in a link? from document importance to topical relevance. In *Proceedings of the 2nd International Conferences on the Theory of Information Retrieval (ICTIR 2009)*, volume 5766 of *LNCS*, pages 313–321. Springer Verlag, Berlin, Heidelberg, 2009.
- [10] R. Schenkel, F. Suchanek, and G. Kasneci. YAWN: A semantically annotated wikipedia xml corpus. In *12th GI Conference on Databases in Business, Technology and Web (BTW 2007)*, March 2007.
- [11] B. Sigurbjörnsson and J. Kamps. The effect of structured queries and selective indexing on XML retrieval. In *Advances in XML Information Retrieval and Evaluation: INEX 2005*, volume 3977 of *LNCS*, pages 104–118, 2006.
- [12] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-Based Approach to XML Retrieval. In *INEX 2003 Workshop Proceedings*, pages 19–26, 2004.
- [13] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Mixture models, overlap, and structural hints in XML element retrieval. In *Advances in XML Information Retrieval: INEX 2004*, volume 3493 of *LNCS 3493*, pages 196–210, 2005.
- [14] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: a language-model based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, 2005.

A Recursive approach to Entity Ranking and List Completion using Entity Determining Terms , Qualifiers and Prominent n-grams

Madhu RM, Srikant R, Venkatesh Karthik S, Meenakshi Sundaram Murugesan,
Saswati Mukherjee

Department of Computer Science and Engineering,
College of Engineering, Guindy,
Anna University, Chennai-60025, India
{ strmmadhu, r.srikant.k, coolvenks.kar}@gmail.com
msundar_26@yahoo.com
msaswati@yahoo.com

Abstract. This paper presents our approach for INEX 2009 Entity Ranking track which consists of two subtasks *viz.* Entity Ranking and List Completion. Retrieving the correct entities according to the user query is a three-step process *viz.* extracting required information from the query and the provided category information, extracting the relevant documents and ranking the retrieved documents making use of the structure available in the Wikipedia Corpus. We have extracted the Entity Determining Terms (EDTs), Qualifiers and prominent n-grams from the query, strategically exploited the relation between the extracted terms and the structure and connectedness of the corpus to retrieve links which are highly probable of being entities and then used a recursive mechanism for retrieving relevant documents through the Lucene Search. Our ranking mechanism combines various approaches that make use of category information, links, titles and WordNet information, initial description and the text of the document.

Keywords: Entity Ranking, List Completion, Entity Determining Terms (EDTs), Qualifiers, Prominent n-grams, Wikipedia tags.

1 Introduction

Search Engines are widely used to retrieve relevant information from the World Wide Web. However, the task of identifying the necessary information from the relevant documents is left to the user. Research on Question Answering (QA) addresses this problem by analyzing the documents and locating the information according to the user's need. Text Retrieval Conference (TREC) pioneered this research through the Question Answering track [1] and promotes three types of Questions *viz.* factoid, list and complex questions. TREC's Enterprise track [2] has a task similar to Entity

Ranking called Expert Search where a set of expert names (people) are to be identified for a given topic.

Entity Ranking track and List completion task [3] started in 2007 as part of the INEX are aimed at exploring methodologies for retrieving relevant list of documents corresponding to entities (answers) using the Wikipedia XML corpus. In the Entity Ranking task, the query and the category are provided and we are required to retrieve all the entities that match them. In the List Completion task, the query and a few sample entities are given and the remaining entities should be extracted and ranked. The challenge lies in handling different XML tags to filter the exact entities from the set of all related documents and to rank them. The XML corpus consists of 2,666,190 articles collected from Wikipedia. The two subtasks are Entity Ranking (ER) and List Completion (LC). Figure 1 shows an example INEX Entity Ranking topic from INEX 2009.

```
<inex_topic topic_id="139">
<title> Films directed by Akira Kurosawa </title>
<description> find the list of movies directed by Akira Kurosawa
</description>
<narrative>The expected answers are movies directed by the Japanese
director Akira Kurosawa
</narrative>
<categories>
<category>japanese films </category>
<entities>
<entity id="477031">Sanshiro Sugata </entity>
<entity id="187603">Rashomon (film) </entity>
<entity id="75984">Ran (film) </entity></entities>
</inex_topic>
```

Fig. 1. A sample topic from INEX 2009's ER track.

Here, the category "japanese films" and the title "Films directed by Akira Kurosawa" can be used to identify the relevant entities in the Entity Ranking task whereas, the title and the example entities "Sanshiro Sugata", Rashomon (film) and Ran (film) can be used for the List Completion task.

The articles in the Wikipedia corpus are well-organized in such a way that each article starts with an overview which we call Initial Descriptions (IDES) and contains several paragraphs of relevant information labeled with subtitles along with many links to other Wikipedia articles and concludes with references. The documents containing related information are grouped into categories and each document may fall under one or more such categories. In addition, the Wikipedia 2009 corpus contains WordNet tags for titles and links indicating the genre under which the article falls.

Our approach stresses on the importance of extracting the Entity Determining Terms (EDTs), Qualifiers and Prominent n-grams in arriving at a clear distinction between entities and non-entities. We use Lucene to find the initial set of relevant

documents and then use a recursive expansion mechanism so as to arrive at a set that encompasses nearly all the relevant documents. For ranking the retrieved documents we have combined various approaches that rely on the category matching, title keyword matching, WordNet confidence factor and synonyms, expansion of links present at relevant positions and paragraph expansion.

In Section 2 Related Work is explained. Section 3 describes our approach and Section 4 shows the results that we have submitted and Section 5 concludes the paper.

2 Related Work

Using the Part-of-Speech (POS) information for finding the focus of the question given by TREC is explored [4]. Here the authors used the predefined patterns to find the question types. Using Named Entity Recognition (NER) for query processing in List Question Answering is shown [5]. The authors classified the web pages into four categories *Viz.* collection page, topic page, relevant page and irrelevant page.

The information available from Wikipedia can be used for building a Named-Entity recognition system, which is shown in [6]. In this paper, authors have shown how category labels can be extracted from Wikipedia. Ranking sentences by giving importance to the proper names is discussed in [7]. Assigning different weights for WordNet synonyms and stemmed terms to improve the ranking is also explored. Wikipedia article names can be used to form effective queries and the effectiveness of the Initial Descriptions (IDES) is also explored [8]. The role of Wikipedia categories in ranking the entities is shown [9]. The use of Lucene as an efficient information retrieval engine is demonstrated in several Question Answering systems such as in [10].

3 Our Approach

Discovering entities by simply expanding all links of the top ‘n’ documents retrieved by Lucene Search would be a rather rudimentary way to discover entities. The structure of the given document needs to be thoroughly understood to discover heuristics as to what to search in which part of the document. For example, links present in a paragraph named as “Movies” are more probable of being entities than the links in a paragraph named “Early Life” for the query given in Figure 1. Similarly, if an XML page which has the same title as that given in the query is found, almost all links in the page have good candidature of being entities. Close examination of the structure of the pages may reveal that many such nuances could be found. Further, once all possible candidates for entities have been listed, further heuristics like category information and the WordNet confidence information could be used to rank the documents as entities. However, in order to mine for the clues scattered in the corpus and to frame heuristics, the right terms need to be searched at the right locations. This in turn, implies that the required terms need to be correctly segregated into the relevant term groups. This is exactly the approach that this paper proposes.

Further, in order to achieve a good precision, we make use of several ranking heuristics. To increase our recall, the entire procedure is made recursive to extract more entities from the available set of entities.

In our proposed approach we try to accomplish both the tasks of Entity Ranking and List Completion by doing the following steps:

- 1) Query Processing.
- 2) Extracting relevant documents.
- 3) Filtering actual entities.
- 4) Entity Ranking

3.1 Query Processing

The main difference between the classical document retrieval task and Entity Ranking task is that the former returns all documents containing relevant information whereas the latter should return only documents that are entities eliminating all other documents. Hence, in order to distinguish between documents that are entities from those that are not, we extract the Entity Determining terms (EDTs). For the ER track, the categories and the words in the query that are synonymous to the category terms are taken as EDTs. Apart from these, the synonyms of the available terms taken from WordNet are also considered as EDTs. The rationale behind this is the fact that the category terms signify the actual hypernym of entities that are expected which helps in distinguishing the entities from other documents. In the LC task, since the category information is not provided an intersection of the categories of the example entities will yield the probable set of EDTs. However, to verify their correctness we need to check if they are synonymous with the terms in the query.

For the example given in fig. 1 the EDTs would be

- 1) “Japanese films” - category information given
- 2) “films” - synonymous word present in the query.
- 3) “movies” - WordNet synonym

Though the EDTs help us to determine whether a document is a possible entity or not, more information is required to determine whether the entity document meets the description specified in the query. In this example, the EDTs specify that the entity should be a Japanese film or a movie in general, but do not indicate anywhere that it should be directed by Akiro Kurosawa. We call such terms that describe the entities as Qualifiers. From the query, we remove the stop words and EDTs and take all the remaining terms as Qualifiers and we stem the required terms. These qualifiers are used for filtering and ranking of the entities. For the example given in Fig. 1 the Qualifiers would be “Akira Kurosawa” and “direct”.

In natural language, a group of terms together may give a separate meaning when compared with the meaning of each of the terms individually. Such combinations of terms are called n-grams, where ‘n’ represents the number of terms in the meaningful unit. We call these n-grams as Meaningful n-grams. Meaningful n-grams in the topic,

if identified properly, effectively improve the relevance of the initial set of documents retrieved. Each of the articles in Wikipedia discusses about a particular topic and has a name associated with it. The list that we have made use in our approach is the expanded list comprising of Wikipedia article names and WordNet synonyms which we call “Wiki Names List”. The topic given is checked against the list and Meaningful n-grams are identified. However, all the Meaningful n-grams in the topic are not equally important. We used the Part-of-Speech (POS) information¹ and the articles corresponding to them to find the prominent n-grams. The articles on these prominent n-grams have the highest probability of having the links to the possible entities and can be used in entity extraction and ranking. In our example, the prominent n-gram would be “Akira Kurosawa”.

3.2 Extracting Relevant Documents

We used Lucene to index the Wikipedia corpus. Once the query processing is done, we retrieve the top ‘n’ documents using the given query as such, to extract the initial set of relevant documents. Though this initial set may contain a few entities, it may not encompass the set of all entities. In order to overcome this problem, we make use of a combination of heuristics viz. Initial Category Expansion, Prominent n-gram expansion, Title Query match, Document category expansion and Paragraph expansion. All these techniques rely on the document titles, category information, proximity information and the prominent n-grams.

In the Initial Category Expansion technique, we make use of the categories given for the ER task and the intersection of categories of the sample entities for the LC task and perform a category search that extracts all the documents that fall under these categories. The rationale behind this is that the category information provides an exact match of the set of entities required. However, not all entities come under these categories and hence we go in for the other techniques also.

As explained earlier, the articles with the prominent n-grams as their title have a high probability of containing links to the actual entities. Hence, all the links in such documents are added to the list of relevant documents. However, we need to verify if the documents thus obtained are actual entities or not, which is taken care of in step 3.3.

In certain cases, the query itself occurs as the title of a document. For example: Q63 Hugo awarded best novels. In this case there is a document with the title of “Hugo Award for best novels”. Hence, the entries in the possible entities list is checked for such exact title query matches. If such a match exists, then the links in that document are added to the list of possible entities.

¹ <http://web.media.mit.edu/~hugo/montytagger>

The initial category given in the query may not encompass all the entities and we need some means of getting all the related categories. For this we make use of the EDTs and Qualifiers and look for their presence in the category information provided in the list of documents already extracted. The document categories that have one or more EDTs and qualifiers are also expanded as in Initial category expansion and the extracted documents are added to the relevant document list.

In the List Completion task apart from the above methods, we make use of the paragraph expansion technique which is similar to a proximity search where the paragraph size determines the window size. Each of the retrieved documents is checked for paragraphs that contain all the sample entities listed along with the qualifiers. This implies that the paragraph has a high probability of containing the other entities too and so all the links in that paragraph are added to the relevant documents list.

These techniques when applied recursively to the set of relevant documents extracted at each step help in greatly increasing the recall, where the increase percentage is proportional to the number of levels up to which recursion is allowed. The precision does not get affected by increasing the level of recursion because the next step employs techniques using the EDTs and WordNet information to filter the actual entities alone. The only constraint to increasing the level of recursion would be the processing time. An optimum value should be chosen taking into consideration the processing time and the recall required.

3.3 Filtering Actual Entities

Though the previous step populates the list with all the possible entities one has to verify if each of them is an entity or just an article related to entities, because the prominent n-gram expansion and title query match expansion techniques may yield links that are not real entities. The document's categories and its WordNet tags are the two main features that enable the distinction of an entity from a non-entity. We follow two mechanisms to separate out the actual entities namely Category verification and WordNet verification. In category verification, each of the documents obtained as a result of the previous step is taken and the document's categories are explored to check for the presence of one or more EDTs. Similarly, in WordNet verification, each of the documents is checked for the presence of EDTs in the WordNet tags. If any one of the above tests is true then the document is added to the Final Entity List, otherwise it is considered as a non-entity and discarded.

3.4 Entity Ranking

The previous step yields the set of actual entities. However, they have to be ranked according to the actual degree of relevance to the query. This ranking of the retrieved documents is done using the WordNet tags, category terms and the locality of query terms in the paragraphs.

The INEX 2009 Wikipedia corpus contains the WordNet tags added to the titles and links in the document. The WordNet tag contains the WordNet id, WordNet term and the precision percentage which indicated how related the WordNet term is to the word that is tagged. We consider the WordNet terms of the title of the document alone for ranking purposes and count for the number of EDTs that are present in the WordNet tags. The WordNet fitness is assigned a value proportional to this count after scaling.

To enhance the ranking further, we have used the category information available for the articles in Wikipedia. Each Wikipedia article falls under a few categories. Apart from checking for the presence of EDTs in the categories we look for the number of qualifier terms present in the categories. The ratio of the maximum number of qualifiers present in a document category to the total number of qualifiers is taken as the category fitness.

$$\text{Category Fitness} = \frac{\text{Max number of qualifiers present in the document's categories}}{\text{total number of qualifiers}} \quad (1)$$

In addition, for the List Completion task, we used the categories of the given example entities as reference set (E_c). This set is compared against the set of categories the retrieved document belongs (R_c). The ratio of the match is used to find the similarity between the retrieved entity and the example entities. This is added to the category fitness score.

$$\text{Category Match Score} = \frac{\text{cat}(E_c) \cap \text{cat}(R_c)}{\text{cat}(R_c)} \quad (2)$$

Apart from these, we used a third technique similar to a proximity search where each paragraph in a document is checked for the presence of EDTs and qualifiers. Depending upon the number of qualifiers present the paragraph fitness is increased. If the paragraph being considered is the initial description paragraph then a greater weight is assigned since the relevancy of information in the IDES is greater when compared to the other paragraphs. This is same as the proximity search except that the window size is not fixed but depends upon the size of the paragraph.

The final rank is taken as the sum of the WordNet fitness, category fitness and the paragraph fitness and the entities are ranked based on this score.

4 Evaluation

INEX 2009 topics are selected from the previous editions INEX ER track. A sample Entity Ranking and List Completion results retrieved by our approach for the topic in Figure 1 are shown below.

Entity Ranking:

139 0 WP1624660 Dersu Uzala (1975 film)
139 0 WP1942885 Madadayo
139 0 WP235331 Red Beard
139 0 WP2553318 Scandal (1950 film)
139 0 WP477031 Sanshiro Sugata
139 0 WP6716962 After the Rain (film)
139 0 WP180241 Sanjuro
139 0 WP187603 Rashomon (film)

List Completion

139 0 WP1624660 Dersu Uzala (1975 film)
139 0 WP235331 Red Beard
139 0 WP2553318 Scandal (1950 film)
139 0 WP31371 Seven Samurai
139 0 WP477031 Sanshiro Sugata
139 0 WP180241 Sanjuro
139 0 WP187603 Rashomon (film)
139 0 WP75984 Ran (film)

5 Conclusion

The method we have deployed relies heavily on Entity Determining terms, qualifiers and the prominent n-grams. But since the category hierarchy is not specified the scope of expanding the EDTs is reduced. However, the WordNet tagging introduced in the 2009 Wikipedia corpus provides a better distinction between entities and non-entities. Though our method provides better accuracy and recall in majority of the queries for those that have some form of reference to time periods as in “movies shot after 1999” it fails. Better results for such queries can be obtained by introducing NLP techniques to semantically process such queries.

References

1. Ellen M. Voorhees.: Overview of the TREC 2001 Question Answering Track. In: Proceedings of the 10th Text REtrieval Conference, 2001.
2. N. Craswell, A.P. de Vries, I. Soboroff.: Overview of the TREC 2005 Enterprise Track. In: Proceedings of the 14th Text REtrieval Conference, 2005.
3. A.P. de Vries, Vercoustre, A., Thom, J. A., Craswell, N., and Lalmas, M.: Overview of the INEX 2007 Entity Ranking Track. In: Focused Access To XML Documents: 6th

- international Workshop of the initiative For the Evaluation of XML Retrieval, INEX 2007 Dagstuhl Castle, Germany, December 17-19, 2007.
4. Chen, J., Diekema, A., Taffett, M.D., McCracken, N., Ozgencil, N.E., Yilmazel, O., and Liddy, E.D. Question Answering: CNLP at the TREC 10 Question Answering Track. In Proceedings of the 10th Text REtrieval Conference, 2001.
 5. Hui Yang, Tat-Seng Chua. : Web-based list question answering. In: Proceedings of the 20th International Conference on Computational Linguistics, 2004.
 6. J. Kazama and K. Torisawa.: Exploiting Wikipedia as External Knowledge for Named Entity Recognition. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007.
 7. U. Hermjakob, E.H. Hovy and Chin-Yew Lin.: Knowledge-Based Question Answering. In: Proceedings of the 6th World Multiconference on Systems, Cybernetics and Informatics (SCI-2002), Orlando, FL, U.S.A., July 14-18, 2002.
 8. Murugesan, M. S. and Mukherjee, S.: An n-Gram and Initial Description Based Approach for Entity Ranking Track. In Focused Access To XML Documents: 6th international Workshop of the initiative For the Evaluation of XML Retrieval, INEX 2007 Dagstuhl Castle, Germany, December 17-19, 2007.
 9. James A. Thom, Jovan Pehcevski, Anne-Marie Vercoustre. : Use of Wikipedia Categories in Entity Ranking. In: Proceedings of the 12th Australasian Document Computing Symposium (ADCS 07), Melbourne, Australia, December 10, 2007
 10. Mark A. Greenwood, Mark Stevenson and Robert Gaizauskas.: The University of Sheffield's TREC 2006 Q&A Experiments. In: Proceedings of the 15th TREC, 2006.

Overview of the INEX 2009 Interactive Track

Nils Pharo¹, Ragnar Nordlie¹, Norbert Fuhr², Thomas Beckers² and Khairun Nisa Fachry³

¹Faculty of Journalism, Library and Information Science, Oslo University College, Norway
nils.pharo@jbi.hio.no, ragnar.nordlie@jbi.hio.no

²Department of Computer Science and Applied Cognitive Science, University of Duisburg-Essen, Germany
norbert.fuhr@uni-due.de, tbeckers@is.inf.uni-due.de

³ Archives and Information Studies, University of Amsterdam, the Netherlands
k.n.fachry@uva.nl

Abstract. In the paper we present the organization of the INEX 2009 *interactive track*. For this year's experiments the iTrack collects data on user search behavior in a collection consisting of book metadata taken from the bookstore Amazon and the social cataloguing application LibraryThing. Thus the data are more structured than in previous years' experiments, consisting of traditional bibliographic metadata, user-generated tags and reviews and promotional texts and reviews from publishers and professional reviewers. Through monitoring searches based on three different task types the experiment aims at studying how users interact with highly structured data. We describe the methods used for data collection and the tasks performed by the participants.

1 Introduction

The INEX interactive track (iTrack) is a cooperative research effort run as part of the INEX Initiative for the Evaluation of XML retrieval [1]. The overall goal of INEX is to experiment with the potential of using XML to retrieve relevant parts of documents. In recent years, this has been done through the provision of a test collection of XML-marked Wikipedia articles. The main body of work within the INEX community has been the development and testing of retrieval algorithms. Interactive information retrieval (IIR) [2] aims at investigating the relationship between end users of information retrieval systems and the systems they use. This aim is approached partly through the development and testing of interactive features in the IR systems and partly through research on user behavior in IR systems. In the INEX iTrack the focus over the years has been on how end users react to and exploit the potential of IR systems that facilitate the access to *parts* of documents in addition to the full documents.

The INEX interactive track (iTrack) was run for the first time in 2004 [3], repeated in 2005 [4], in 2006/2007 [5] (due to technical problems the tasks scheduled for 2006 were actually run in early 2007), and in 2008 [15]. Although there has been variations in task content and focus, some fundamental premises has been in force throughout:

- a common subject recruiting procedure

- a common set of user tasks and data collection instruments such as interview guides and questionnaires
- a common logging procedure for user/system interaction
- an understanding that collected data should be made available to all participants for analysis

This has ensured that through a manageable effort, participant institutions have had access to a rich and comparable set of data on user background and user behavior, of sufficient size and level of detail to allow both qualitative and quantitative analysis. This has already been the source of a number of papers and conference presentations ([6], [7], [8], [9], [10], [11], [12]).

In 2009, it was felt that although the "common effort" quality of the previous years was valuable and still held potential as an efficient way of collecting user behavior data, the Wikipedia collection had exhausted its potential as a source for studies of user interaction with XML-coded documents. We decided to base the experiments on a new data collection with richer structure and more semantic markup than has previously been available. A crawl of 2 million records from the book database of the online bookseller Amazon.com was consolidated with corresponding bibliographic records from the cooperative book cataloguing tool LibraryThing. The records present book descriptions on a number of levels: formalized author, title and publisher data; subject descriptions and user tags; book cover images; full text reviews and content descriptions. The data base was intended to enable investigation of research questions concerning, for instance

- What is the basis for judgments on relevance in a richly structured and diverse material? What fields / how much descriptive text do users make use of / chose to see to be able to judge relevance?
- How do users understand and make use of structure (e.g. representing different levels of description, from highly formalized bibliographic data to free text with varying degrees of authority) in their search development?
- How do users construct and change their queries during search (sources of terms, use and understanding of tags, query development strategies ..)?

2 Tasks

For the 2009 iTrack the experiment was designed with two categories of tasks constructed by the track organizers, from each of which the searchers were instructed to select one of three alternative search topics. In addition the searchers were invited to perform one semi-self-generated task. The two categories of tasks were intended to reflect the most common purposes a searcher would have for visiting a database of primarily bibliographic data, a broad, explorative task and a narrower, more specific, purpose-driven task. The self-selected task was intended to force the searcher to make a more quality-driven task than the two others. The tasks provided were as follows:

The broad tasks

These tasks are designed to investigate thematic exploration which will give us data on query development, metadata type preference and navigation patterns.

1. You are considering to start studying sociology. In order to prepare for the course you would like to get acquainted with some good and recent introductory texts within the field as well as some of its classics.
2. You are interested in taking a course on environmental friendly energy. In order to prepare for the course you would like to get acquainted with some good introductory texts on the field.
3. You are considering to start studying existentialism. In order to prepare for the course you would like to get acquainted with some good introductory texts within the field as well as some of its classics.

The narrow tasks

These tasks represent relatively narrow topical queries where the purpose is to allow us to study the basis for relevance decisions and compare the searchers' preference of different document representations.

1. Find trustworthy books discussing the conspiracy theories which developed after the 9/11 terrorist attacks in New York.
2. Find books which present documentation of the specific health and/or beauty effects of consuming olive oil.
3. The Kabbalah is an esoteric religious tradition which has inspired works of fiction. Find novels where the plot is inspired by the Kabbalah, and a factual treatment of the origins and development of this tradition.

The semi self-selected task

For one of the courses you are currently attending, you need an additional textbook. You have only money for one book (assuming they all have about the same price). You are free to select the course topic yourself.

3 Participating groups

Due to unfortunate delays in the preparation of the experimental system, the experiments were launched only a week before the deadline of this paper. Five

groups of experimenters have so far expressed intent of participation, with a deadline for completion of experiments on January 31, 2010.

4 Research design

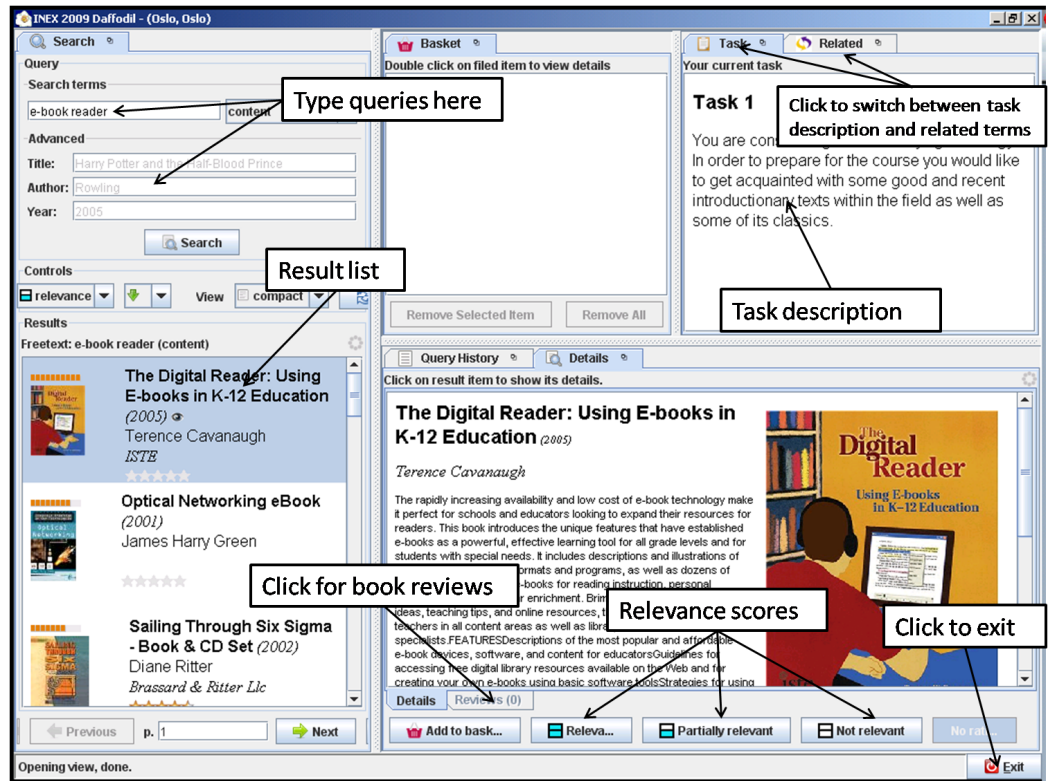
4.1 Search system

The experiments are conducted on a java-based retrieval system built within the Daffodil framework [14], which resides on a server at and is maintained by the University of Duisburg. The basis of the search system is the same as have been used for previous iTracks, but the interface has been modified extensively to accommodate the new data set, and a set of new functionalities have been developed.

Figure 1 shows the interface of the system. The main features available to the user are

- When a search term is entered, the searcher can choose to search on “content”, “reviews”, or both together. “Content” searches all the “formalized” text connected to each book – title, keywords, publisher’s description etc. “Reviews” searches the text of any user reviews of the book. In both cases the search index bases result rankings on term occurrence. In addition, there is field-based search available on author, title or publication year.
- The system can order the search results according to “relevance” (which books the system considers to be most relevant to your search terms), “year” (publication year of the book), or “average rating” (in the cases where people have rated the quality of the books).
- The system will show results twenty titles at a time, with features to assist in moving further forwards or backwards in the result list.
- A double click on an item in the result list will show the book details in the “Details” window. If the book has been reviewed, the reviews can be seen by clicking the “Reviews” tab at the bottom of this window.
- The relevance of any which is examined should be determined, as “Relevant”, “Partially relevant” or “Not relevant”, by clicking markers at the bottom of the screen. Any book decided to constitute part of the answer to the search task should be moved to a result basket by clicking the “Add to basket” button next to the relevance buttons.
- When the first search term has been entered, the system will use the task window to suggest search terms which might be relevant to the task. A double click on a term in this list will move it to the search term window.
- A “Query history” button in the middle of the screen displays the search terms used so far in the search session.
- A line of yellow dots above an item in the result list is used to indicate

the system's estimate of how closely related to the query the item is considered to be.



4.2 Document corpus

The document corpus consists of records from the book database of the online bookseller Amazon.com, consolidated with corresponding bibliographic records from the cooperative book cataloguing tool LibraryThing. The XML-coded records present book descriptions on a number of levels: formalized author, title and other bibliographic data; controlled subject descriptions and user-provided tags; book cover images; full text reviews and publisher-supplied content descriptions.

4.3 Online questionnaires

During the course of the experiment, searchers are issued brief online questionnaires to support the analysis of the log data. Before the search tasks are introduced, the searchers are given a pre-experiment questionnaire, with demographic questions such as searchers' age, education and experience in information searching in general and in searching and buying books online. Each search task is preceded with a pre-task questionnaire, which concerns searchers' perceptions of the difficulty of the search task, their familiarity with the topic etc. After each task, the searcher is asked to fill out a post-task questionnaire. The intention of the post-task questionnaire is to learn about the searchers' use of and their opinion on various features of the search system, in relation to the just completed task. The experiment closes with a post-experiment questionnaire, which elicits the searchers' general opinion of the search system.

4.4 Relevance assessments

The users' task is partly to indicate the relevance of any item in the result list found sufficiently interesting for them to view in detail, partly to collect a result set which they consider to constitute an answer to their task. A three-part relevance scale of "relevant", "partly relevant" and "not relevant" is used.

4.5 Logging

All search sessions are logged and saved to a database. The logs register and time stamp the events in the session and the actions performed by the searcher, as well as the responses from the system. In addition to system logs, participating institutions will to a large extent log additional data through eye-tracking, screen image capture etc.

5 Experimental Procedure

Each experiment is performed following the standard procedure outlined below. Steps 7 to 10 were repeated for each of the three tasks performed by the searcher.

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 experimental system. Tutorial of the system is given with a training task provided by the system. The experimenter hands out and explains the system features document.
3. Any questions answered by the experimenter.
4. The experimenter administers the pre-experiment questionnaire.

5. Topic descriptions for the first task category administered, and a topic selected.
6. Pre-task questionnaire administered.
7. Task begins by clicking the link to the search system. Maximum duration for a search is 15 minutes, at which point the system issues a “timeout” warning. Task ended by clicking the “Finish task” button.
8. Post-task questionnaire administered.
9. Steps 5-8 repeated for the second and third task.
10. Post-experiment questionnaire administered.

6 Data analysis and conclusions

As the experiment phase has recently started, results will be reported in the final version of this track report.

References

- [1] Malik, S., Trotman, A., Lalmas, M. & Fuhr, N. (2007): Overview of INEX 2006. In: Fuhr, N., Lalmas, M. and Trotman, A. 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 2006*. Berlin: Springer, p. 1-11.
- [2] Ruthven, I. (2008): Interactive Information Retrieval. In: *Annual Review of Information Science and Technology*, 42, p. 43-91.
- [3] Tombros, A., Larsen, B. and Malik, S. (2005): The Interactive Track at INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S. and Szlávik, Z. eds. *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004*. Berlin: Springer, p. 410-423
- [4] Larsen, B., Malik, S. and Tombros, A. (2006): The interactive track at INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S. and Kazai, G. eds. *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005*. Berlin: Springer, p. 398-410.
- [5] Larsen, B., Malik, S. & Tombros, A. (2007): The Interactive track at INEX 2006. In: Fuhr, N., Lalmas, M. and Trotman, A. 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 2006*. Berlin: Springer, p. 387-399.
- [6] Pharo, N. & Nordlie, R. (2005): Context Matters: An Analysis of Assessments of XML Documents. In: F. Crestani and I. Ruthven eds. *Information Context: Nature, Impact, and Role: 5th International Conference on Conceptions of Library and*

- Information Sciences, CoLIS 2005, Glasgow, UK, June 4-8, 2005*. Berlin: Springer, p. 238-248.
- [7] Hammer-Aebi, B., Christensen, K. W., Lund, H. and Larsen, B. (2006): Users, structured documents and overlap: interactive searching of elements and the influence of context on search behaviour. In: Ruthven, I. et al. eds. *Information Interaction in Context : International Symposium on Information Interaction in Context : IIIiX 2006 : Copenhagen, Denmark, 18-20 October, 2006 : Proceedings*. Copenhagen: Royal School of Library and Information Science, p. 80-94.
- [8] Malik, S., Klas, C.-P., Fuhr, N., Larsen, B. and Tombros, A. (2006): Designing a user interface for interactive retrieval of structured documents: lessons learned from the INEX interactive track? In: Gonzalo, J. et al. eds. *Research and Advanced Technology for Digital Libraries, 10th European Conference, ECDL 2006*. Alicante, Spain, September 17-22, 2006, Proceedings. Berlin: Springer,
- [9] Kim, H. & Son, H. (2006): Users Interaction with the Hierarchically Structured Presentation in XML Document Retrieval. In: Fuhr, N., Lalmas, M., Malik, S. & Kazai, G. eds. *Advances in XML Information Retrieval and Evaluation: 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005*. Berlin: Springer, p. 422-431.
- [10] Kazai, G. & Trotman, A (2007): Users' perspectives on the Usefulness of Structure for XML Information Retrieval. In: Dominich, S. & Kiss, F. eds. *Proceedings of the 1st International Conference on the Theory of Information Retrieval*. Budapest: Foundation for Information Society, p. 247-260.
- [11] Larsen, B., Malik, S & Tombros, A. (2008): A Comparison of Interactive and Ad-Hoc Relevance Assessments. In: Fuhr, N., Kamps, J., Lalmas, M. & Trotman, A. eds. *Focused Access to XML Documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007 Dagstuhl Castle, Germany, December 17-19, 2007*. Berlin: Springer, p. 348-358.
- [12] Pharo, N. (2008): The effect of granularity and order in XML element retrieval. *Information Processing and Management*. 44(5), 1732-1740.
- [13] Kamps, J., Geva, S., Trotman, A., Woodley, A. & Koolen M. (2009). Overview of the INEX 2008 Ad Hoc Track. In: *Proceedings from the 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2009*. Berlin: Springer
- [14] Fuhr, N., Klas, C.P., Schaefer, A. & Mutschke, P. (2002): Daffodil: An integrated desktop for supporting high-level search activities in federated digital libraries. In *Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, p. 597-612.
- [15] Pharo, N., Nordlie, R. & Fachry, K. N. (2009): Overview of the INEX 2008 Interactive Track. In: Geva, S., Kamps, J. and Trotman, A. eds. *Advances in Focused Retrieval, 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008, Dagstuhl Castle, Germany, December 2008*. Berlin: Springer, p. 300-313.
- [16] Pehcevski, J. (2006): Relevance in XML retrieval: the user perspective. In: Trotman, A. and Geva, S. eds. *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology : Held in Seattle, Washington, USA, 10 August 2006*. Dunedin (New Zealand): Department of Computer Science, University of Otago, p. 35-42.
- [17] Pharo, N. (2002): The SST Method Schema: a tool for analyzing work task-based Web information search processes. Doctoral Thesis. University of Tampere

An Overview of INEX 2009 Link the Wiki Track

Wei Che (Darren) Huang¹, Shlomo Geva² and Andrew Trotman³

Faculty of Science and Technology, Queensland University of Technology, Brisbane,
Australia ^{1,2}

Department of Computer Science, University of Otago, Dunedin, New Zealand ³

w2.huang@student.qut.edu.au ¹

s.geva@qut.edu.au ²

andrew@cs.otago.ac.nz ³

Abstract. In the third year of the Link the Wiki track, the focus has been shifted to anchor-to-BEP link discovery. There are two collections, Wikipedia and Te Ara, used in 2009 for respective tasks. For the link the wiki tasks, 5000 file-to-file topics were randomly selected while 33 anchor-to-BEP topics were nominated by the participants. The Te Ara tasks are adopted for the first time and the participants are encouraged to discover all anchor-to-BEP links for the entire collection. A validation tool for pre-examining the link discovery results was distributed. This helps participants verify that the offset of the anchor text is specified correctly. The assessment tool was revised to improve efficiency. Both file-to-file and anchor-to-BEP runs are evaluated against Wikipedia ground-truth. Focus-based evaluation was undertaken using a new metric. Evaluation results are presented.

Keywords: Wikipedia, Link Discovery, File-to-File, Anchor-to-BEP, Assessment, Evaluation.

1 Introduction

The Link the Wiki task was [1] run for the first time in 2007 [2]. It aims at providing an independent evaluation forum for discussing approaches of anchor-to-BEP link discovery in the Wikipedia. The participants are encouraged to utilize different technologies (e.g. data mining, NLP, information retrieval, etc.) to resolve the issue of anchor-to-BEP link discovery. The goal is to rank relevant anchors and links to a best entry point in the target document.

In 2007, the file-to-file runs were evaluated against the Wikipedia whilst the anchor-to-BEP was announced as a task in 2008 [3]. Highly accurate approaches to file-to-file link discovery were described, in comparisons with the Wikipedia ground truth. In 2009, several improvements have been made to the anchor-to-BEP evaluation, including the tools, assessment methods and metrics. Apart from the Wikipedia collection, the Te Ara encyclopedia is introduced and the tasks, *Link Te Ara* and *Link Te Ara to Wiki*, are set up for the first time.

The platform for the evaluation of anchor-to-BEP link discovery consists of a set of resources, including link discovery rules, document collections, qrels, metrics and assistant tools. The participants are encouraged to explore the content in the

Wikipedia collection by using different technologies. The experiment focuses on approaches to link discovery. The goal of link discovery is to construct a high quality link graph by generating a comprehensive set of anchors and links to/from each document. For instance, in order to improve the Te Ara encyclopedia, links to contents inside the encyclopedia, and to/from the Wikipedia, are added, thereby extending the Te Ara knowledge base with the Wikipedia. This type of application could be used in various scenarios, such as education, library and mobile knowledge discovery.

Six groups from different organizations participated in the 2009 tasks. 16 runs were received for the file-to-file task while 13 runs for the anchor-to-BEP task and 8 runs for the F2F on A2B task were submitted. Two groups were also involved in the Te Ara tasks with 7 runs contributed. All link the wiki runs were evaluated against the Wikipedia ground truth. All anchor-to-BEP runs were additionally evaluated in different ways such as anchor-to-file and anchor-to-BEP. The former qrels can be produced from the Wikipedia ground truth while the later is done through the manual assessment. A set of evaluation results have been produced and a brief discussion is also presented.

2 Document Collection

Two collections, the Wikipedia and the TeAra, were used in the Link the Wiki track in 2009. The Wikipedia corpus is a 2,666,190 article dump of the Wikipedia. This collection is much larger than the one used in 2008. For file-to-file link discovery, 5000 articles were randomly selected, but filtered by certain criteria such as the document size and the number of anchors (i.e. links). This can have the quality control of the documents used in the task. For anchor-to-BEP link discovery, 33 topics were nominated by the participants and topics would be assessed by their nominators.

Apart from the Wikipedia collection, the Te Ara Encyclopedia was introduced and used in the Link the Wiki track for the first time. Its size is around 50 MB without images. Currently there is no link in the collection and some of documents are still small. Two tasks are designated for the Te Ara collection. *Link-Te-Ara* is to discover anchor texts and link them to best entry points within the collection. *Link-Te-Ara-to-Wiki* is designated to link the anchor text from a Te Ara topic to best entry points in the Wikipedia document.

3 Task Specification

3.1 Tasks

The task was specified as twofold: the identification of links from the orphan into the document collection; and the identification of links from the collection into the orphan.

File-to-File link discovery for the Wikipedia collection: This task can be described as a special case of the anchor-to-BEP task, which decrease the complexity and offers an entry level for newcomers. Documents within the Wikipedia collection are used. Up to 250 outgoing links and up to 250 incoming links might be specified per topic. Missing topics would be regarded as having a score of zero for the purpose of computing system performance. Anchor-to-BEP link discovery: This task represents the main goal of the Link the Wiki track. Researchers are encouraged to discover the link discovery approaches, produce a reliable qrel and participate in the forum for discussing solution to link discovery issues. Documents in the Wikipedia and Te Ara collections are adopted. Only 50 topics and up to 5 BEPs per anchor should be specified for each topic. Anchors and links last would not be taken into account for performance evaluation. Alongside, at most, 250 incoming links can be specified in the case of Link-the-Wiki. Each incoming links must be from a different document. Only outgoing links are needed for the Te Ara tasks because the entire documents are used.

3.2 Submission

Each submission run must specify the task (i.e. *LTW_F2F*, *LTW_A2B*, *LTArA_A2B* and *LTArATW_A2B*) performed. The description section in the submission format is important for the classification of different link discovery approaches from the same participant. A sample format in the case of the Wikipedia is presented below.

```

<outgoing>
  <anchor name="Luminiferous aether" offset="1688" length="19">
    <tobep offset="2038">123456</tobep>
    <tobep offset="971">359</tobep>
    ...
  </anchor>
  ...
</outgoing>
<incoming>
  <bep offset="2038">
    <fromanchor offset="799" length="9" file="654321">radiation</fromanchor>
    <fromanchor offset="1019" length="10" file="3162088">medication</fromanchor>
    ...
  </bep>
  ...
</incoming>

```

Fig. 1. Sample Submission Format

An anchor link is specified in four parts; the document file name, the start position of the anchor (i.e. **Offset**), the **Length** of the anchor text and the anchor text itself. The first three define the FOL of the anchor link while the last factor is used to verify the anchor is specified correctly by the OL. The offset specified the anchor starting byte position within the corresponding text-only document, and the anchor length is specified in bytes. The document name could be a unique number in the Wikipedia, or a unique name in the Te Ara collection. A destination link could be specified in two parts: a unique file name and a best entry point (BEP). It is the best starting point of the content that can well describe the context of the corresponding anchor text.

3.3 Restriction of Linking

An anchor, indicated by a combination of *Offset* and *Length*, must appear only once in a topic - although it may have multiple distinct BEPs. An anchor-text in one document can be linked to several destinations (BEPs) in other distinct documents. It means that the same set of *Offset* and *Length* should not appear more than once and hence there is no duplicated anchor for a given topic. For the evaluation purpose, only first 5 links within the instances of the same anchor are taken. Document title can also be an anchor, but like any other anchor it can be linked to at most 5 destinations. This intends to encourage the discovery of the most suitable anchor for the topic. The title text could appear anywhere in the document. However, the assessors might accept only one set of offset and length regarding to that particular anchor text. As a consequence, other instances of this anchor text would not be taken for evaluation; even the relevant links are connected. In the case of anchor-to-BEP evaluation, the same target document can be linked from different anchor texts once it has been indicated as relevant. On the contrary, the duplicated links will be truncated and not used for file-level evaluation.

The number of anchors and its BEP links is highly restricted. Only first 50 anchors and first 5 BEPs per anchor may be taken for the evaluation purpose. Both anchors and BEPs within each anchor must be ranked and listed in decreasing order of relevance. By contrast, there is no anchor and BEP indicated in file level link discovery but 250 outgoing and 250 incoming file-level links may be specified.

3.4 Assistant Program

In order to facilitate the production of the offset for each anchor and BEP, several tools have been developed and distributed to participants. A Java program, *XML2FOL*, was built to generate a list of all the element nodes offsets and lengths for a given XML document. Another Java program, *XML2TXT*, was also prepared for the participants to convert the XML document into the text-only content. Apart from the tools, a text-only version of the collection was also provided so the offset could be computed by counting the characters from the beginning of the document. These two programs could be embedded into the participant's link discovery system as a parser of indicating term position.

The validation tool was developed and delivered to the participants for validating the generated runs. The anchors are highlighted in the left screen while the right screen shows the link content with a best entry point on it and a table recording the hierarchical structure of anchor-links for the given topic. The participants can click on a link in the table to check the particular anchor-link result. This tool intends to bring up what the link discovery application should look like and facilitate to revise the results.

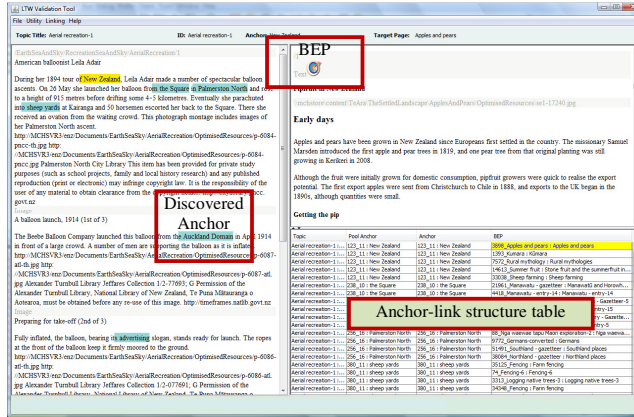


Fig. 2. The Validation Tool

4 Preparation of qrels

There are two types of qrels for evaluation. One is the Wikipedia ground truth, the other one is generated from the participants' runs. The Wikipedia ground truth is produced from the existing anchors and links in the Wikipedia collection. This is the easiest way to automated evaluation. However, according to the experiment last year, the comparison of performance between submission runs by using Wikipedia ground truth is unsound. Some Wikipedia links are topically-obsolete or redundantly assigned. Most of anchors are linked to the documents with the same name. The relevant portions of the document content have not been further discovered. All relevant contents that are not in the Wikipedia are also considered non-relevant for the evaluation. As a consequence, the evaluation result might appear biased. However, evaluation based on the Wikipedia ground-truth does measure performance relative to what is present, and so it is reasonable to believe it is useful. The general idea is to experiment how easy the existing technology can achieve to produce the similar anchors and links in the Wikipedia.

Apart from the file-to-file ground truth, the Wikipedia is also able to generate the anchor-to-file (i.e. offset is zero) ground truth. Although the Wikipedia does contain anchor-to-BEP links, in practice they are rarely used. In order to experiment the anchor-to-BEP technology, a special pooling procedure has been applied to collect all anchors and links from participants' runs. The pool was assessed to completion.

5 Assessment and Evaluation

5.1 Manual Assessment

The pool for each topic was generated by the following three parts: Anchor-to-BEP (A2B) link discovery submission, the file-to-file link discovery on A2B (F2FonA2B) topics and anchor-to-file ground truth. In the case of F2FonA2B, the anchor was set as the topic title and linked to the beginning of the target document. The anchor-to-file set from the Wikipedia presents a one-to-one relation and the BEP was set at the position zero. The duplicated links were removed to prevent from multiple-assess. This procedure performs an exhausted collection of anchors and links.

Since the offset sometimes could hardly be specified precisely and anchor terms might be indicated in different ways (e.g. *Information*, *Technology* and *Information Technology* are all possible anchor terms), overlapped anchor texts were merged as a so-called pool anchor. A pool anchor might contains more than one anchors generated from different submission runs. Therefore, the anchor texts showing on the screen might not definitely be the anchors returned by the participants; instead it might be a combined text term that involved several anchor texts and respective links. In order to prevent from merging unallied anchor texts, the pooling program adjusts the offset to separate pool anchors. Anchors pooling is also to allow the evaluation being tolerant of certain deviation for anchor text location.

5.2 Assessment Tool

As the assessment is laborious and time consuming, several attempts have been made on improving the efficiency.

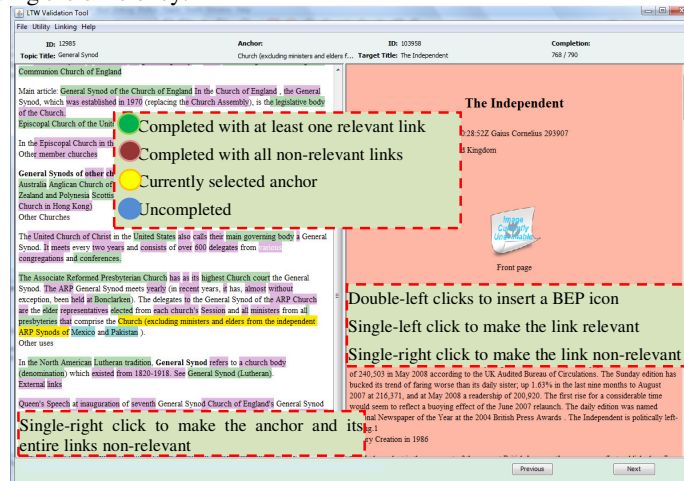


Fig. 3. The Assessment Tool

Firstly, the anchor text could be made non-relevant by mouse right-click, it could save time to assess even hundred links. On the link pane, only a single right or left click is necessary to judge the relevance in the case of incoming links. Moving the mouse cursor is not even necessary. According to the survey carried out after the assessment, a lack of related anchor texts highlighted in the incoming document could be a major obstacle of efficiency. Sometimes it is difficult to identify whether the incoming document is relevant to the topic content or not. Indicating the BEP in the target document is also a difficult task to achieve without any supplemental information (e.g. system's discovered BEP). Highlighting anchor texts or related phrases on the document seems necessary. For instance, a sub-title or a paragraph teeming with the linking anchor text (or related phrases) could be a best start point for reading from.

Each topic contains around 1000 anchor links and 900 incoming links. An average around 4 hours was spent to finish a topic.

5.3 Metrics

As with all metrics, it is important to first define the use-case of the application. The assumption at INEX is that link-discovery is a recommendation tasks. The system produces a ranked list of anchors and for each a set of recommended target/BEP pairs. The list should also be comprehensive because it is not clear that the document author can know a priori which links will be relevant to a reader of the document. That is, link discovery is a recall oriented task. The Mean Average Precision based metrics are very good at taking rank into account and are recall oriented. A good metric for link discovery should, consequently, be based on MAP. The difficulty is computing the relevance of a single result in the results list. For evaluation purposes it is assumed that if the target is relevant and the anchor overlaps a relevant anchor then the anchor is relevant; $f_{anchor}(i) = 1$.

The assessor might have assessed any number of documents as relevant to the given anchor. If the target of the anchor is in the list of relevant document then it is considered relevant; $f_{doc}(i) = 1$. The contribution of the links' BEP is a function of distance from the assessor's BEP [4]:

$$f_{bep}(j) = \begin{cases} \frac{n - 0.9 \times d(x, b)}{n} & \text{if } 0 \leq d(x, b) \leq n \\ 0.1 & \text{if } d(x, b) > n \end{cases}$$

Where $d(x, b)$ is the distance between submission BEP and result BEP in character. Therefore, the score of $f_{bep}(j)$ varies between 0.1 (i.e. d is greater than n) and 1 (i.e. the submission and result BEPs are exactly matched). The score of 0.1 is reserved for the right target document with an indicated BEP not in range of n . n typically is set up as 1000 (characters). The score of a result in the results is then:

$$P = \left[f_{anchor}(i) \times \frac{(\sum_{j=1}^m (f_{doc}^i(j) \times f_{bep}^i(j)))}{m_i} \right]$$

Where m is the number of returned links for the anchor and m_i is the number of relevant links for the anchor in the assessments. As the result list is restricted to 5

targets per anchor m_i is capped at 5 for evaluation. A perfect run can thus score a MAP of 1.

5.4 Evaluation

A portable (Java) evaluation tool, *LtwEval*, was developed for evaluation purposes. It is GUI based and provides numerous evaluation metrics including: precision, recall, MAP, and precision@R. Different runs can be evaluated and compared to each other. Interpolated-Precision/Recall graphs can be generated for sets of run. New functionality has been added to allow the manipulation of the graphs, which increase the usability of the tool.

For the file-to-file evaluation (i.e. F2F and F2FonA2B), the number of outgoing and incoming links have been restricted by 250. Links beyond this number were truncated. The total number of relevant links is based on the ground truth, but at last 250 to make sure the measurement of Recall is meaningful. For the anchor-to-BEP evaluation against ground-truth, the first 50 anchors for each topic were taken and the first link from each anchor was collected. As a result, there were 50 outgoing links per topic, used for evaluation. By contrast, first 250 incoming links were taken to do the evaluation since the discovery of BEP in the topic document is not that obvious. Most incoming links belong to the same BEP. Therefore, in the INEX use case of link discovery it is important to rank the discovered links for presentation to the page author. This use case was modeled in the manual assessment where assessors did exactly this. In a realistic link discovery setting the user is unlikely to trudge through hundreds of recommended anchors, so the best anchors should be presented first. The link discovery system must also balance extensive linking against link quality.

7 Results and Discussion

The Queensland University of Technology (i.e. QUT) submitted 6 runs for the file-to-file (F2F) task, 4 runs for the anchor-to-BEP (A2B) task and 1 run for the F2FonA2B task. University of Waterloo contributed 2 runs on the A2B task and 5 run for the F2FonA2B task. University of Amsterdam had 5 runs for the A2B task. University of Otago submitted 1 runs for the F2F task, 2 runs for the A2B task and 2 runs for the F2FonA2B task. University of Wollongong submitted 4 runs for the F2F task. Technische Universität Darmstadt contributed 4 runs on the F2F task. Apart from the Link the Wiki tasks, QUT also participated in the Link the Te Ara and Link Te Ara to Wiki tasks by submitting 1 run each. Technische Universität Darmstadt also contributed 5 runs on the Link the Te Ara task. These runs were generated by the anchor-to-BEP link discovery technology.

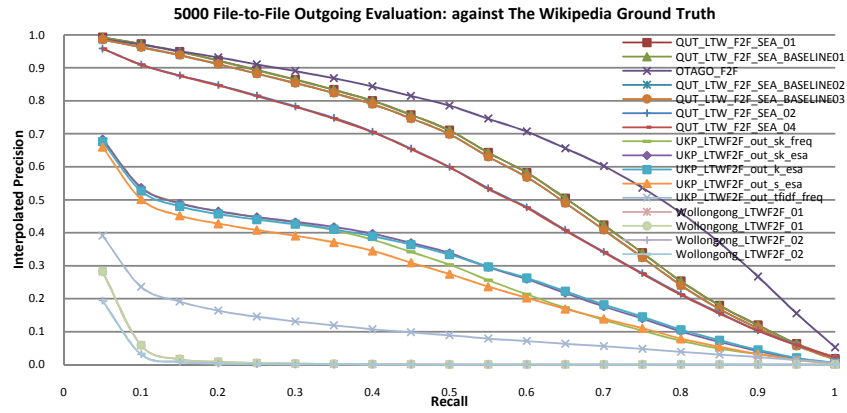


Fig. 4. 5000 F2F Topics Outgoing link discovery evaluated against Wikipedia Ground Truth

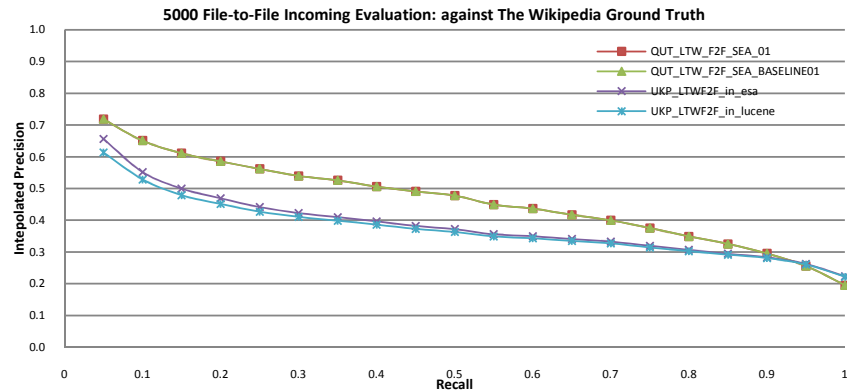


Fig. 5. 5000 F2F Topics Incoming link discovery evaluated against Wikipedia Ground Truth

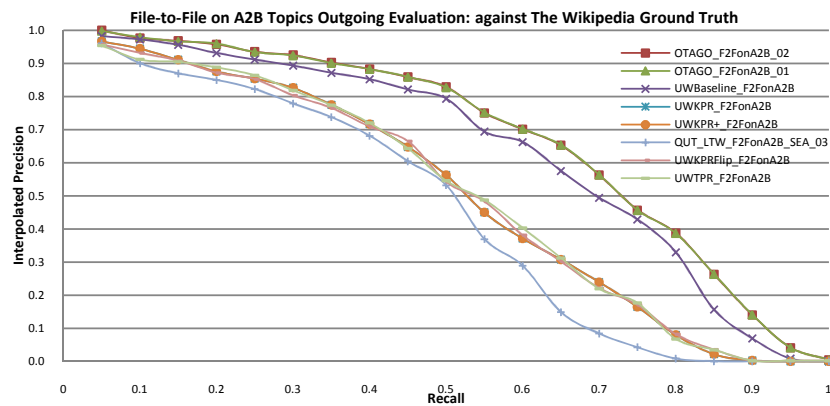


Fig. 6. F2F on A2B Topics Outgoing links evaluated against Wikipedia Ground Truth

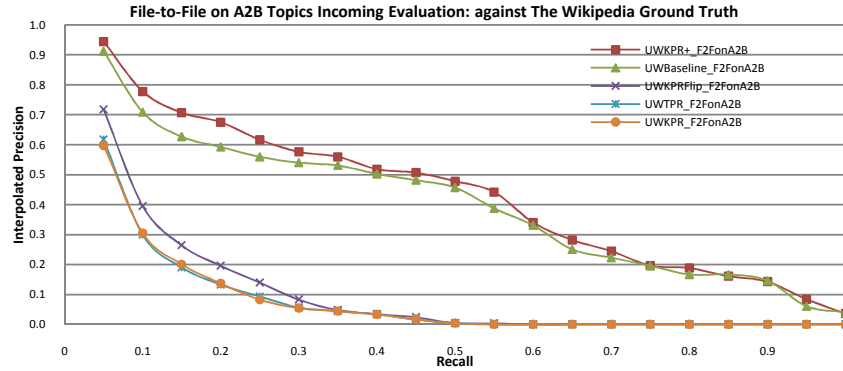


Fig. 7. F2F on A2B Topics Incoming links evaluated against Wikipedia Ground Truth
33 Anchor-to-BEP Outgoing Evaluation: against The Wikipedia Ground Truth

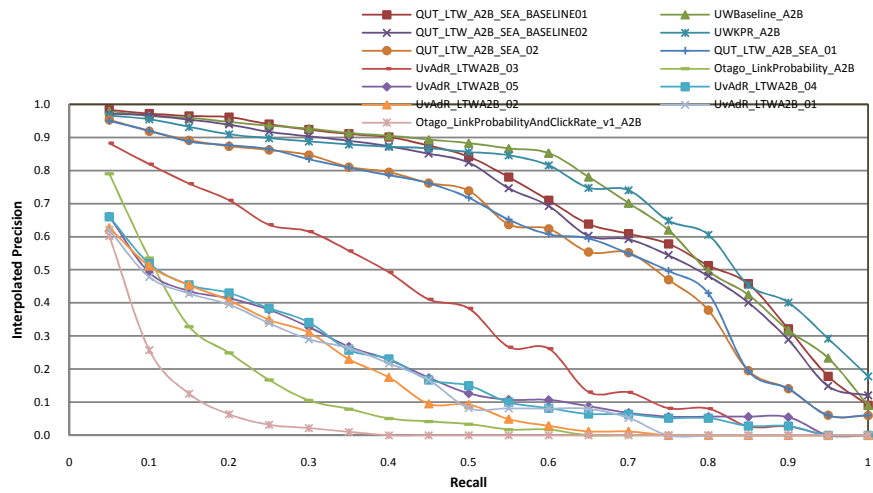


Fig. 8. 33 A2B Topics Outgoing links evaluated against Wikipedia Ground Truth
33 Anchor-to-BEP Incoming Evaluation: against The Wikipedia Ground Truth

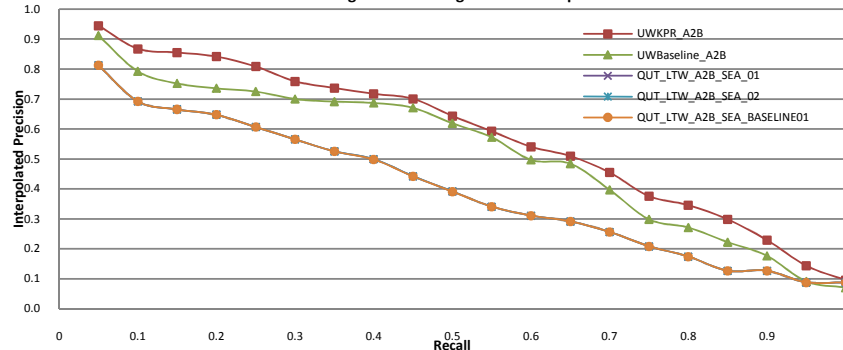


Fig. 9. 33 A2B Topics Incoming links evaluated against Wikipedia Ground Truth

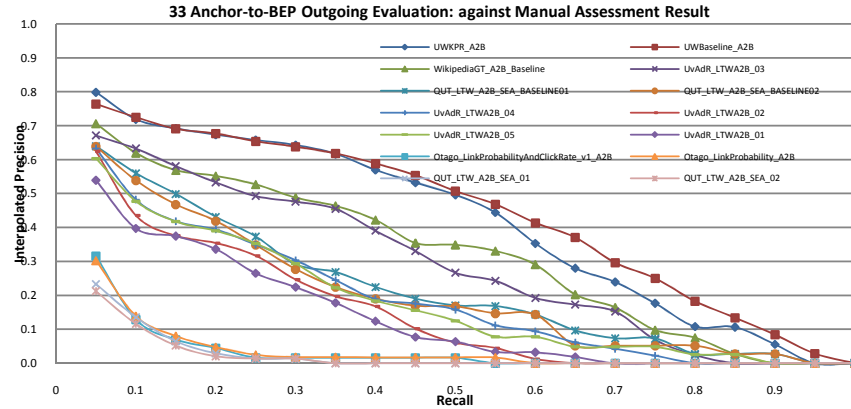


Fig. 10. 33 A2B Topics Outgoing links evaluated against Manual Assessment Set

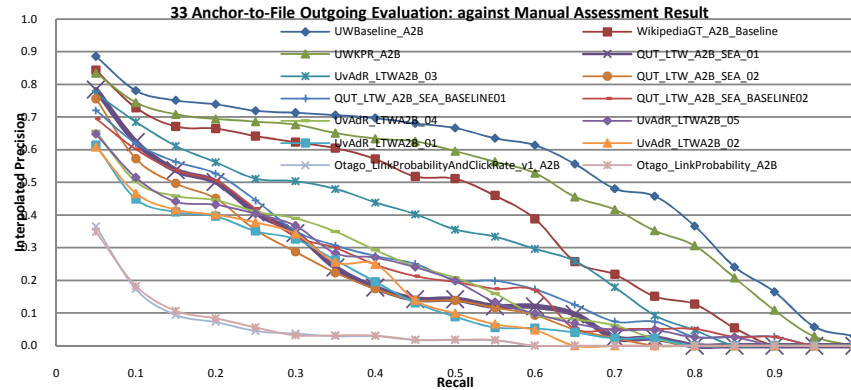


Fig. 10. 33 A2F Topics Outgoing links evaluated against Manual Assessment Set

The University of Waterloo (UW) had two approaches, one baseline and the other link-based, to undertake the experiment. For baseline, UW made statistics on the frequency phrases. These phrases were located in the topic files and the most frequent links were returned. For incoming links, we scored the corpus using topic titles as query terms and returned the top documents. The link-based approach computes *PageRank* and *Topical PageRank* values for each file in the corpus for each topic, and returned the top scoring pages according to the contribution of *K-L divergence*. For incoming links, UW reversed the graph to get new *PageRank* values and returned the top pages according to the contribution of *K-L divergence* with the new *PageRank* values and the old *Topical PageRank* values.

The Queensland University of Technology (QUT) used the statistical link information of Wikipedia corpus to calculate the probability of anchors and their corresponding target documents for a list of sortable outgoing links. A hybrid approach that combines the results of link analysis method and title matching algorithm for the prediction of potential outgoing links was also undertaken. For the incoming links, the top ranking search results with topic title as the query terms

retrieved from a BM25 ranking search engine were chosen as source documents that can be linked to the topics. In finding the BEPs for either outgoing or incoming links, we tried two different methods: one is that BEP is the position of phrase in target document where the terms of anchor, either the entire words or part of which, appear; the other one is that BEP is the beginning of a text block which has similar terms features with that of the passage which is extracted from the surrounding text of anchor in source document.

8 Conclusion and Outlook

This is the third year of the Link the Wiki track at INEX. According to the file-to-file experiment, producing Wikipedia links could be achieved by current approaches. In 2009, the focus has been shifted to the anchor-to-BEP link discovery and several changes have been made to improve the evaluation procedure. Assistant tools were prepared to self-examine the status of submission. An online questionnaire was set up to gather the suggestions and ideas of the assessment. Based on this outcome, an interview will be involved at the workshop to classify the need of the evaluation. Further experiments will be undertaken on the anchor-to-BEP runs. The submission would be evaluated on anchor-to-file, file-to-BEP and anchor-to-BEP level to test the usability of approaches provided. This aims to classify the performance of each approach on the contribution of linking for the given topic. The Te Ara collection is introduced for the first time at INEX to bring up the new concept of cross collection link discovery. Through the focus link discovery, the Wikipedia content could be fully explored. Anchors indicated for the given document could be linked to the most relevant content in the Wikipedia. Every piece of content discovered in the Wikipedia can be used to provide links from anchors in the document from other collections. Going through this process, a well defined knowledge network can be constructed. Based on participants' comments and ideas via survey, customization can be made, and the enhancement of evaluation procedure and efficiency is expected. According to the experiment, the contribution of each approach can be classified and future direction of anchor-to-BEP link discovery can be possibly pointed out.

References

1. Trotman, A. and Geva, S. Passage Retrieval and other XML-Retrieval Tasks, In: the SIGIR 2006 Workshop on XML Element Retrieval Methodology, pp. 48-50.
2. Huang, W. C., Xu, Y., Trotman, A. and Geva, S. (2008) Overview of INEX 2007 Link the Wiki Track, INEX 2007, LNCS 4862, N. Fuhr et al. (Eds.), pp. 373-387.
3. Huang, W. C., Xu, Y., Trotman, A. and Geva, S. (2009) Overview of INEX 2008 Link the Wiki Track, INEX 2008, LNCS 5631, N. Fuhr et al. (Eds.), pp. 314-325.
4. Huang, W. C., Xu, Y., Trotman, A. and Geva, S. (2009) The Methodology of Manual Assessment in the Evaluation of Link Discovery, In Proceedings of the 14th Australian Document Computing Symposium.

Link Prediction for Interlinked Documents by using Probability Measure Self Organizing Maps for Structured Domains.

M. Kc¹, R. Chau³, M. Hagenbuchner¹, A.C. Tsoi², V. Lee³

¹ University of Wollongong, Wollongong, Australia.

Email:{millykc,markus}@uow.edu.au

² Hong Kong Baptist University, Hong Kong. Email:act@hkbu.edu.hk

³ Monash University, Melbourne, Australia.

Email:{Rowena.Chau,Vincent.lee}@infotech.monash.edu.au

Abstract. This paper explains how a recent development in an area of machine learning, namely, Self-Organizing Maps (SOM) can be used for the generation of links between referenced or otherwise interlinked documents. These new generation of SOM models are capable of projecting generic graph structured data onto a fixed sized display space. Such a mechanism is normally used for dimension reduction, visualization, or clustering purposes. This paper shows that the SOM training algorithm “inadvertently” encodes relations that exist between the atomic elements in a graph. If the nodes in the graph represent documents, and the links in the graph represent the reference (or hyperlink) structure of the documents, then it is possible to obtain a set of links for a test document whose link structure is unknown. It will be shown that the proposed approach is scalable in that links can be extracted in linear time. It will also be shown that the proposed approach is capable of predicting the pages which would be linked to a new document, and is capable of predicting the links to other documents from a given test document. The approach is applied to web pages from Wikipedia, a relatively large XML text database consisting of many referenced documents.

1 Introduction

Self-Organizing Maps are a popular unsupervised machine learning approach for the clustering and projection of high dimensional data vectors [1]. Recent developments extended the algorithms’ ability to encode and cluster graph structured data. The methodology has been applied successfully to a clustering tasks involving documents retrieved from Wikipedia as part of a participation in the INEX (Initiatives for the Evaluation of XML retrieval) document mining competition.

Graph data structures allow the representation of almost any kind of learning problems. A graph consists of a set of nodes and a set of binary relations called links. A link connects any two nodes in a graph if these nodes are related to each other in some way. If the relationship is directed, then this is represented by a directed link. Otherwise it is undirected. With directed links, the source node is called a *parent* and the destination node is called a *child*. With undirected links, each node connected by a link to a given node is called a *neighbour*. A node with n children and m parents is said to have an out-degree of n and an in-degree of m . A node with k neighbors is said to have a degree

of k . A node can be labelled by a numeric vector so as to provide a description of the properties of the associated object. Sequences are a special case of graphs in which the maximum in-degree and maximum out-degree is one. Traditional data vectors are also a special type of graphs for which there exists no link between the nodes (all nodes are independent). Thus, any approach capable of dealing with graphs can also deal with data sequences and vectors.

Many problems are appropriately represented by a graph data structure. For example, Chemical Molecules are naturally represented as a graph in which the nodes represent the atoms, and the (undirected) links represent the atomic binding between two atoms. This paper addresses another domain which is appropriately represented by a directed graph: Web pages. Web pages can contain hyperlinks which point from a given web page to another one. Thus, this defines a directed graph in which the web documents are represented by a node in the graph, and the hyperlinks are the directed links in the graph. A data label associated with each node in the graph can add a description of the content of a document. For example, the well-known Bag of Words approach would summarize the content of a document in vectorial form.

There has been much activities in recent years on approaches that can process graph structured information. Some of the most successful approaches were developed in the area of machine learning. This success stems from the fact that the approaches are scalable to real world data mining tasks. Moreover, some of these approaches have been proven to be capable of solving any given problem involving graphs optimally [2].

This paper describes the latest of the unsupervised machine learning approaches, and shows that the method can also be used for the purpose of link prediction. Link prediction has particularly important practical applications in the World Wide Web (WWW) domain. For example, a central algorithm in the search engine Google, known as PageRank, relies on link analysis for the purpose of ranking a set of linked documents. PageRank produces a large value for pages with many parents and few children. A known weakness of PageRank is that it neglects pages which have been newly added to the Web. Such new pages do not have any parents by default (as they are not referenced by other web pages due to their newness in the WWW), and hence would be ranked lowly. This in turn causes Google not to rate new pages highly producing an effect known as “the-rich-get-richer” effect or “the-poor-stay-poor” effect. Another problem with the Web domain is that it is unregulated. Anyone can add a new document containing any hyperlink and any number of hyperlinks. This is often exploited by companies and individuals which create *link-farms* designed at increasing the rank of a target page. These examples highlight the need for algorithms capable of predicting links between any two documents in the Web. Such algorithms can then be used to automatically suggest parent nodes which should link to a newly created document, and can be used to verify whether existing links are valid ⁴.

This paper is organized as follows: Section 2 describes the Probability Measure Graph SOM model, and how it can be used to predict links in a hyperlinked domain. Section 3 applies the proposed approach to a relatively large real world problem on link predictions. Conclusions are drawn in Section 4.

⁴ Links which exist for the sole purpose of increasing a rank of a given page are said to be *spam* links or *invalid*, otherwise the link is said to be valid.

2 The approach

The SOM is a well-known algorithm in unsupervised learning [1] of data vectors. The SOM is a topological preserving map, in that data which is close to one another in the high dimensional feature space will remain close in the low dimensional display space. In its most basic form, we consider a two-dimensional display space being discretized into an $N \times M$ grid [1]. Each grid point i, j has an associated m dimensional codebook vector $\mathbf{c}_{i,j}$. Thus, there are $N \times M$ of these m dimensional codebook vectors. These are initialized randomly. The input vectors to the SOM are also m dimensional ones. The SOM is trained in two steps:

- **Competition:** A given input vector is compared with each of the m dimensional codebook vectors in the $N \times M$ grid by using the Euclidean distance measure. The best matching codebook \mathbf{c}_{i_k, j_k} is said to be the winner.
- **Parameter adjustment:** The elements of codebook \mathbf{c}_{i_k, j_k} and all its neighbors are pulled closer to the elements of the input vector as follows:

$$\Delta \mathbf{c}_{i,j} = \alpha(t) f(\Delta_{i, \{i_k, j_k\}}) (\mathbf{c}_{i,j} - \mathbf{u}) \quad (1)$$

where \mathbf{u} denotes the input vector, α is a learning rate which decreases steadily towards 0, and $\Delta_{i, \{i_k, j_k\}}$ is the neighborhood of the winning vector \mathbf{c}_{i_k, j_k} , and $f(\cdot)$ is a nonlinear function, often chosen to be a Gaussian function.

These two steps are repeated for each input vector in a training set, and for a given number of cycles. When the algorithm converges, the elements of the two dimensional grid $N \times M$ encapsulate the set of high dimensional input vectors \mathbf{u} .

The SOM is trained on vectorial information under the assumption that the input vectors are independent. The algorithm needs to be modified if there is a dependency defined on the input vectors. Such dependencies are normally represented as a graph structure. A first approach to process trees was made with the introduction of a SOM for Structured Data (SOM-SD) [3]. A tree is a special class of graphs which is a rooted, directed, acyclic graph. In this case, the issue is how to encode the graph structure. One way in which this can be obtained is to consider each node in the graph as independent and can be modelled using a SOM. As this is a tree structure consisting of directed links, it makes sense to process the data from the leaf nodes to the root node. The issue is then how to connect the SOM model for each node with others (as there is no obvious way in which this can be performed). A simple minded approach would be to consider the winning node in the SOM model of each node, and then to somehow connect it to the parent node⁵. The winning vector of the SOM model of the node represents the outcome of the SOM model of the node. Hence it makes sense to use this vector and to somehow connect it to the parent node (the node in which it has a directed connection to). One way in which this can be performed is through a concatenation of the input and the vectors associated with the winning nodes (the children of the current node). If we assume that there are only a fixed number of incoming links from the children nodes, then for nodes which have lesser number of incoming children links, the input

⁵ Since we are processing the tree from leaf nodes first towards the root node, the links from a particular node is pointing towards the parents rather than the child.

vector can be suitably augmented with zeros. Thus as far as the current parent node is concerned, the input is a fixed sized vector, and hence the standard SOM training algorithm can be applied; the only variation is that we will need to weigh the relative importance of the input vector to the node, and the vectors associated with the children nodes. This algorithm produced quite good results when processing tree structured data.

One way in which this approach can be extended to the un-directed graphs, cyclic graphs is to consider each node as modelled by a “state”. This is a vector which encapsulates the past information which is processed by the node. Thus for un-directed graphs, or cyclic graphs what it means is that there are both parents and children which a given node is required to consider when processing information. If we assume that the parents and the children nodes are presented by “states”, then their values are available when we process the current node. This trick then allows us to process the data in an un-directed graph or a cyclic graph using the standard SOM training algorithm; again the only difference is that we will need to worry about the relative weights among the input vector to the current node, and the vectors associated with states in the parent and children nodes of the current node. Again these algorithms were found to perform satisfactorily.

The recently introduced Probability Measure Graph SOM (PM-GraphSOM), the latest version of SOM-SD algorithms, addresses the question on how cyclic dependencies can be encoded by a SOM. To achieve this, the PMGraphSOM modifies the interpretation of the *state* vector. We associate with each node a *state* which describes the activation of the SOM for this node. In the context of using self organizing map, we may consider the state as the location of the winning node in the $N \times M$ display map. In this case, for the current node, if we assume in the display map, there are additional inputs from the antecedent (parent) nodes, and the descendant (child) nodes, the locations of these additional inputs are the coordinates of the winning nodes in these antecedent nodes and descendant nodes, together with the associated winning vectors [4]. A weakness of this method is that the Euclidean distance measure does not make a distinction as whether any change of a mapping during the update step has been to a nearby location or to a location far away on the map. To counter this behavior it has been proposed to *soft code* the mappings of neighbors to account for the probabilities of any changes in the mapping of nodes. In other words, instead of *hard coding* the mappings of nodes to be either 1 if there is a mapping at a given location, or 0 if there is no mapping at a given location, we encode the likelihood of a mapping in a subsequent iteration with a probability value. We note that due to the effects of the training algorithm it is most likely that the mapping of a node will be unchanged at the next iteration. But since all vectors associated with the grid points in the display map are updated, and since those vectors which are close to a winning entry (as measured by Euclidean distance) are updated more strongly (controlled by the Gaussian function), and, hence, it is more likely that any change of a mapping will be to a nearby location rather than to a location far away from the last update. These likelihoods are directly influenced by the neighborhood function and its spread. Hence, one can incorporate the likelihood of a mapping in subsequent iterations as follows:

$$M_i = \frac{e^{-\frac{\|\{i_1, j_1\} - \{i_k, j_k\}\|^2}{2\sigma(t)^2}}}{\sqrt{2\pi}\sigma(t)}, \quad (2)$$

where $\sigma(t)$ decreases with time t towards zero, and $\{i_k, j_k\}$ are the coordinates of the winning vector, while $\{i_1, j_1\}$ are the coordinates of the vector in the display space. The computation is cumulative for all the i -th node's neighbors. Note that the term $\frac{1}{\sqrt{2\pi\sigma(t)}}$ normalizes the states such that $\sum_i M_i \approx 1.0$. It can be observed that this approach accounts for the fact that during the early stages of the training process it is likely that mappings can change significantly, whereas towards the end of the training process, as $\sigma(t) \rightarrow 0$, the state vectors become more and more similar to the hard coding method. This approach helps to improve the stability of the GraphSOM significantly, which allows the setting of large learning rates, and reduces the required training time significantly while providing an overall improvement in the clustering performance. This is referred to as the probability mapping GraphSOM (PMGraphSOM) [5].

We note that the PMGraphSOM updates the codebook vectors in the direction of data vectors. We note further that the input vectors contain *state* information which represents the dependencies on other nodes in the graph. Hence, an idea was born to derive the links to a new node from the state component of the the best matching codebook vector. In other words, given a PMGraphSOM which has been trained on a Web graph, and given a new document whose links are not yet known, we can find the best matching codebook vector for this document and obtain the most likely link structure by “reverse engineering” the part of the codebook which represents the state vector.

As a comparison, and as an alternative approach to obtaining links for a new document, we can furthermore consider the following property of the PMGraphSOM: during training, a best matching codebook has been obtained for each of the nodes in a graph. Since each codebook is activated by a number of nodes, and hence, these codebooks are said to be a *representation* of these nodes. Due to the topology preserving ability of the PMGraphSOM, it can be said that all nodes which activated the same codebook are most closely related to each other in terms of content and hyperlinks. Let us now compute a winning codebook for a new document, it makes much sense that this new document shares greatest similarity with all nodes from the training dataset which activated the same codebook. Hence, we can propose a link structure for this new document based on the links contained in the documents which were mapped at the same location.

In the following, we will apply both concepts to predict links for a set of test documents within the Web domain of Wikipedia.

3 Experiments

Given a new Wikipedia document, the file-to-file link discovery task is to analyze the text and recommend a set of up to 250 incoming and 250 outgoing links from one document to other documents in the collection.

Documents from the INEX 2009 Wikipedia collection are used for this task. This collection contains 2,666,190 articles; it is a dump of the Wikipedia taken on 8 October 2008. It is annotated with the 2008-w40-2 version of YAGO. It is 50.7GB in size.

A set of 5000 existing Wikipedia documents, randomly selected from the collection, are used as the test set for the file-to-file link discovery task. Since the topics are not truly orphaned documents (as there are orphaned documents which contain no incoming or outgoing links, a situation similar to the newly introduced documents onto the Web),

we must delete these orphaned files from the collection to simulate the situation of genuinely introducing new documents into the Web.

This task has specific restrictions about the use of link-based information. It was recommended that the link information be processed so that all links pointing to and coming from the documents in the test set are discarded. This is in fact conducted as recommended. The 5000 documents in the test set were also removed from the training set, as is the common practice for machine learning tasks. This results in a training set consisting of 2,661,190 documents.

The whole INEX 2009 Wikipedia collection is represented as a directed graph where each node is a web page. Each page is then represented by a state vector encoding both its contextual and link structural information.

Before the data can be trained by PMGraphSOM, decisions need to be taken regarding the selection of features to be used as node labels for each document. It was important to select a feature which could represent each document, but such features do not contain any link information, as otherwise the testing process would not be able to use the same feature as node labels to represent the test data. Some analysis revealed that the category information of each document could be used as a representative feature. However, the category extraction process identified 8,918,924 categories in total, within which there are 362,251 unique categories. The maximum number of categories in a document reached 2,022, but there are also 118,209 documents with no associated category information at all.

The un-processed category information as node labels would prevent the training process from being completed within a reasonable amount of time; therefore, some dimension reduction is required. Singular Value Decomposition (SVD) was considered for a dimension reduction step. However, SVD requires the building of a 2,661,190 x 362,251 matrix which is far too large for the capacity of computers which we had available for this project. Hence, another approach to dimension reduction was taken. This second approach utilizes a well known Multi-Layer Perceptron (MLP) algorithm to assist in dimension reduction. The MLP algorithm is generally applied to neural network architectures which consist of an input layer, followed by one or more hidden layers of neurons, and then an output layer of neurons, where all neurons in a layer are fully connected to all neurons in the next layer. To utilize MLP for dimension reduction we use an architecture known as the "Auto-associative Memory" (AAM) architecture. In an AAM, both the input and output dimension are the number of unique categories (362,251), and the dimension of the single hidden layer will be the dimension which we would like to reduce to. Using this configuration, the number of neurons (dimension) in the hidden layer is less than the dimension of the input layer, so the encoding process in the hidden layer is, in a sense, compressing the information from the input. Then the connection between the single hidden layer neurons and the output layer neurons can be seen as uncompressing the information back to the original dimension. For training purposes, the input data is also used as the target, so that the MLP can learn a mapping which loses the least amount of information through the compression (hidden) layer of an auto-associative memory. It is known that an MLP trained in this fashion results in a dimension reduction which is qualitatively equivalent to those obtained by using SVD algorithm but without the need of having to store a large matrix in memory.

Table 1. Statistics of the training set’s link structure

	Max	Min	Standard dev.	Number of documents with no link
Out-degree	5295	0	92.96	36,978
In-degree	549,658	0	476.18	304,518

Table 2. Statistics of the test set’s link structure

	Total	Max	Min	Mean	Std dev.	Number of documents with no link
Out-degree	461,741	245	1	92.35	46.05	0
In-degree	311,423	3095	0	62.28	131.50	372

After the node labels are reduced to a more manageable dimension (here we use 16) using the MLP technique as indicated, attention was shifted to the incorporation of link structures within the training set. PMGraphSOM is capable of incorporating link information to assist in the training process, therefore the link structure of the training set after the required pre-processing could be used for training purposes. Some analyses of the links reveal that the number of links from the 2,661,190 training documents total 136,304,216, this equates to a mean of approximately 51.22 links per page. These links are unlikely to be distributed equally, therefore, separate analysis of the out-links and in-links were also carried out. The statistical results can be found in Table 1. The standard deviation indicates that the number of in-links varies much more than the number of out-links. This property is important since it implies that the dataset is unbalanced with respect to the incoming and outgoing links.

Such unbalances in the training dataset are known to potentially cause problems with any machine learning approach.

Although the link structure of the test set will not be used during training, some analyses were carried out to investigate whether the test set is comparable to the training set. This is especially important for machine learning tasks, as a training dataset, which is representative of the testing dataset, will be able to provide more accurate results for the documents the network has not encountered previously, which is the test set in this case. The statistics of the link structure for the testing dataset are included in Table 2.

As can be observed from comparing the statistical information of the link structure, the training dataset and the testing dataset have a number of significant differences.

- The number of documents with no links - It can be observed that a little more than 1% of documents in the training set have no out-links. Based on this ratio, approximately 50 documents in the testing dataset are expected to have no out-links, but the testing dataset contains no such type of documents. The in-link also has a similar problem: with the training dataset containing approximately 11% of documents with no link, in comparison with 7% in the testing dataset.
- A higher number of links in the testing dataset - The average number of out-links and in-links per document in the testing dataset are consistently higher than the 51.22 links per page average in the training set.

These differences suggest that the testing dataset is not a random sampling from the problem domain, but rather a subset selected by some (unknown) criteria. Again, this can impose some added challenges to a machine learning approach.

3.1 Learning for link discovery

We learn patterns of the Wikipedia links by feeding the training set to a PMGraphSOM. After training, all Wikipedia pages will be mapped onto a 2-dimensional display map space where pages with similar contextual content and link structure will be mapped onto the same codebook or onto nearby codebooks. Based on this property of the SOM training algorithm, we claim to be able to discover links for the 5,000 testing pages by inferring from the codebooks of a trained map.

We will use two approaches to infer from the trained map. The first approach maps the test data onto the trained map by comparing the model label of the test documents with that of each of the codebook. The best mapping is the one which has the least Euclidean distance. Then, after the best matching codebook is identified, the state vector component of the winning codebook vector, which comprises of in-link and out-link information, is investigated. This provides information about the likeliness of an in-link or out-link to be mapped at each codebook. Based on this, we were able to identify the codebooks which the child or parent documents are most likely to be mapped, and then identify the documents mapped in the most likely codebook as the proposed links. We refer to this as *codebook-based* link inference.

The second approach also performs a mapping of the test data onto the trained map by finding the best matching codebook. Then, we identify the set of training documents which were mapped onto the same codebook, as these documents are likely to have similar contextual content as the test document. The links from all the corresponding training documents are then collected as the proposed links for the task. We refer to this as *content-based* link inference.

Each of these two approaches can produce an arbitrary number of links which is only limited by the size of the map (for the codebook-based approach), or by the existing link structure of the training set (for the content-based approach). These inferred links are then to be ranked in descending order from the most likely link to the least likely link. Then we will truncate the list of links to the maximum allowable 250 for both, the in-links as well as the out-links. Note that the computed rank values will also play an important role in the evaluation of the results.

Three ranking algorithms were considered for this task. The first is based on the energy flow of a page. This is calculated by accumulating scores when a page receives in-links, but distributing scores when a page contains out-links. The list of proposed out-links for each of the documents in the test set are ordered according to their associated scores. The reverse of accumulating scores from the out-links, and distributing scores to the in-links, is used to order the list of proposed in-links for each test document.

The second ranking algorithm is based on frequency. For example, if many of the training documents indicate that a link is a likely in-link or out-link, then it has a higher frequency of being proposed, and therefore will be ranked higher.

The third ranking algorithm is based on the Euclidean distance of the test document and the training documents. This ensures that the training documents with more contextual feature similarity are ranked higher.

Table 3. Statistics of the test set’s link structure

Submission ID	Map size	μ_1	μ_2	μ_3	Trained map
01	-	-	-	-	-
02	20x40	0.999	0.0005	0.0005	Figure 1
03	10x30	1.0	0.0	0.0	Figure 2
04	20x40	0.991	0.0045	0.0045	Figure 3
05	20x40	0.991	0.0045	0.0045	Figure 3

As mentioned previously, there are two approaches of obtaining an estimation of the local link structure: Firstly by analyzing the state vector of the codebooks in the SOM, and secondly, through association with training patterns. We have submitted results for each of these two approaches. We also investigated the impact of three ranking mechanisms which aim at obtaining the most likely links first.

Unless specified otherwise, in the following, for the training of the PMGraphSOM, we used the following training parameters:

- iterations = 3
- radius is dependent on the map size, but is most often set to 10
- learning rate = 0.9
- seed = 7,

Other parameters such as the size of the map, and the weight μ were varied as indicated later in the paper.

Fixing the above mentioned training parameters allows other parameters to be varied and tested. The parameters under investigation here are the map size and the weights μ . We attempted a large number of combinations of map sizes and weights, the resulting trained maps were analyzed based on the test data performance. We selected the five most representative results for the submission to the INEX LinkTheWiki track, and for the visualization purposes in this paper. These submitted training tasks produced results which were amongst the best from any of the approaches attempted.

The first submission does not have any associated information about the training process, because it was not trained, but merely ranked. For the other submissions, training using PMgraphSOM was carried out on a computing cluster. The map size refers to the size of the 2-dimensional map used by PMgraphSOM during training. The three parameters μ_1 , μ_2 and μ_3 are weights, and $\sum_{i=1}^3 \mu_i = 1$. μ_1 is the weight associated with the node labels, μ_2 is the weight associated with the out-links (the vectors associated with the out-links), and μ_3 is the weight associated with the in-links (the vectors associated with the in-links). A large variation of weights were used during training, and these are the weights that produced the best performance. It can be observed that the weights associated with the links are significantly less than the weights assigned to node labels; this could be attributed to the differences in the dimensions. For example, in this experimental set up, a 16 dimensional vector is used to represent the node labels, whereas the information about the links (both in-links and out-links) is represented by much larger vectors that are dependent on the map size. It should be noted that submissions 04 and 05 are based on the same trained map, but different ranking algorithms were applied to produce different sets of results.

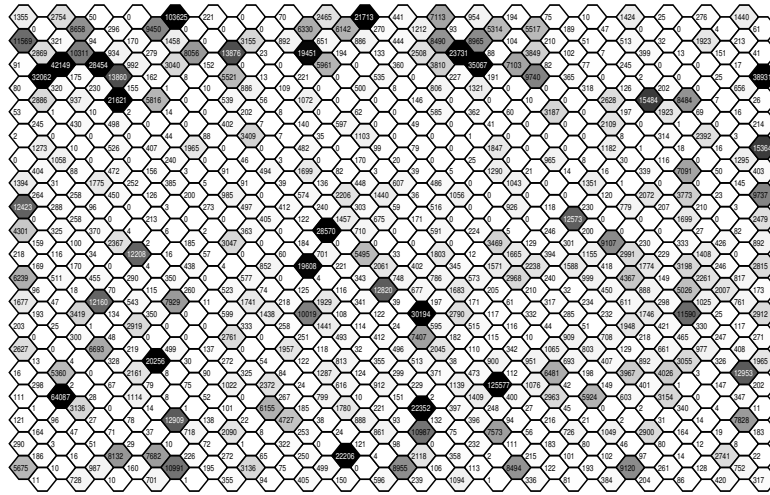


Fig. 1. The trained map of submission 02

The frequency of activation of each codebook after the training process (the trained map) can be observed in Figures 1, 2 and 3 respectively.

3.2 Results and evaluation

The INEX challenge did not describe the evaluation mechanism that will be used to assess the performance of the results. However, some evaluation mechanism is required in order to compare the results and assess the effectiveness of the training configurations.

The evaluation mechanism used for this task includes the precision and recall measures commonly used for information retrieval experiments. Here, precision is defined as $\frac{|t \cup r|}{|r|}$, and recall is defined as $\frac{|t \cup r|}{|t|}$, where t refers to the actual links contained in the test set, and r the links proposed by us. Precision is evaluated on different granularities to observe the performance achieved when varying the number of top ranked links considered. For example, calculating precision for the top n links for each page would be calculated by restricting the size of t to a maximum of n . This is expected to reveal the effectiveness of the ranking algorithms used.

The performance measured using precision and recall for the 5 submissions are included in Table 4. Each of the submissions use a different combination of training configuration and ranking algorithm.

Submission 01 is the best result achievable without using a trained map, but instead, simply apply the ranking algorithm over the entire dataset, and identify the top few ranked pages. As may be observed from Table 4 this simple procedure produces the worst results when compared with those obtained using the training process. The best performance using the trained data, is obtained from submission 03.

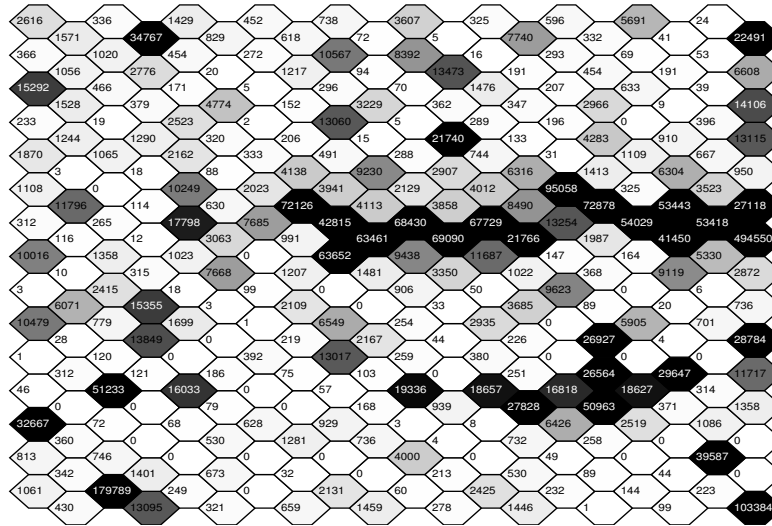


Fig. 2. The trained map of submission 03

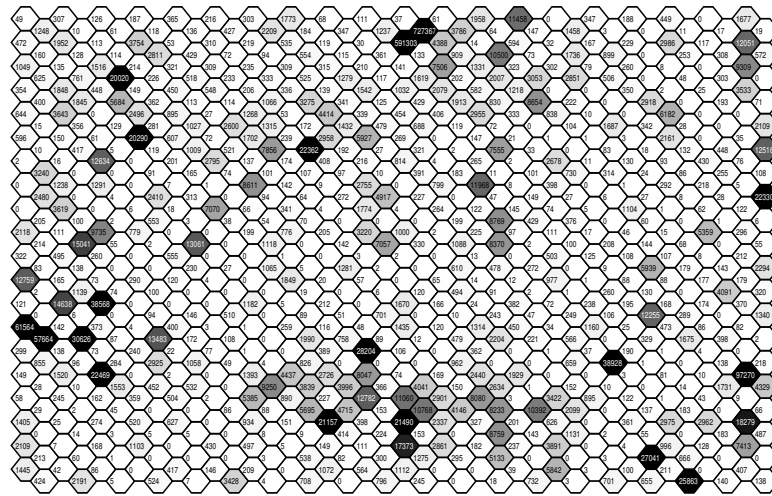


Fig. 3. Visualization of the mappings on the trained map (submissions 04 and 05)

Another observation from Table 4 is that no matter which inference method or ranking algorithm is used, the in-links proposed are less accurate than the out-links proposed.

Table 4. The performance of proposed links as predicted by our proposed algorithms

Submission ID	01	02	03	04	05
Link inference method	Consider all	Content-based	Content-based	Codebook-based	Content-based
Ranking algorithm	Energy flow	Link frequency	Link frequency	Euclid. distance	Euclid. distance
Recall for out-links	0.00377	0.02942	0.03202	0.00070	0.00448
Precision@250 out-links	0.00136	0.01057	0.01817	0.00025	0.00161
Precision@100 out-links	0.00168	0.01795	0.02162	0.00002	0.00147
Precision@20 out-links	0.00040	0.03941	0.04489	0.00003	0.00268
Recall for in-links	0.00032	0.00050	0.00095	0.00010	0.00029
Precision@250 in-links	0.00008	0.00012	0.00025	0.00002	0.00007
Precision@100 in-links	0.00007	0.00008	0.00037	0.00003	0.00008
Precision@20 in-links	0.00007	0.00018	0.00107	0.00001	0.00008

4 Conclusions

Self-Organizing Maps are popularly applied to many data mining tasks for the purpose of clustering, dimension reduction, and visualization. This paper proposes the utilization of a recently developed self-organizing map approach, which is capable of mapping graph structured data, for the purpose of link prediction for “orphaned” documents (the documents which do not have any in-links or out-links yet as they are introduced newly to the Web) in an interlinked domain, Wikipedia. The approach has been applied to a relatively large collection of documents from Wikipedia. It was shown that the approach provides two main alternatives to predict both, incoming and outgoing links for any given “orphaned” document. Some indicative results were obtained by training some relatively small maps. It was shown that the links predicted are substantially better than a random process. Hence, it can be assumed that the accuracy of the prediction will increase with the size of the network. The training of larger networks is left as a future task.

Acknowledgment: This work has received financial support from the Australian Research Council through Discovery Project grant DP0774168 (2007 - 2009).

References

1. Kohonen, T.: Self-Organizing Maps. Volume 30 of Springer Series in Information Sciences. Springer, Berlin, Heidelberg (1995)
2. Scarselli, F., Gori, M., Tsoi, A., Hagenbuchner, M., Monfardini, G.: Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks* **volume 20**(number 1) (January 2009) 81–102
3. Hagenbuchner, M., Sperduti, A., Tsoi, A.: A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks* **14**(3) (May 2003) 491–505
4. Hagenbuchner, M., Sperduti, A., Tsoi, A.: Self-organizing maps for cyclic and unbound graphs. In: European symposium on Artificial Neural Networks. (23-25 April 2008) to appear.
5. Hagenbuchner, M., Zhang, S., Tsoi, A., Sperduti, A.: Projection of undirected and non-positional graphs using self organizing maps. In: European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. (22-24 April 2009)

Discovering Links Using Semantic Relatedness

Johannes Hoffart, Daniel Bär, Torsten Zesch, and Iryna Gurevych

Ubiquitous Knowledge Processing Lab
Computer Science Department, Technische Universität Darmstadt
Hochschulstraße 10, D-64289 Darmstadt, Germany
www.ukp.tu-darmstadt.de

Abstract. We present our approaches for link discovery in document collections with or without existing links. In collections containing links, we discover links using measures of link anchor ranking based on existing links. In collections without links, we gather noun phrases as anchor candidates. To discover targets, we use a measure of semantic relatedness between texts.

1 Introduction

Links are a crucial feature of hypertext to navigate document collections, but creating links is a daunting task. It requires a huge effort to decide which phrases are important enough for the reader to serve as link anchor, and which documents are good targets for that phrase. Additionally, knowledge of the best target implies knowledge of the complete document collection, something which is hard to achieve. Thus, automatically discovering links is an important research topic.

We distinguish two types of document collections: those already containing links and those without any links. The information of document collections that already contain links, e. g. which phrases are often used as links or which documents are linked to by which phrase, can be used for discovering new links. One such document collection that contains collaboratively created links is the Wikipedia¹. It has been the subject of a lot of link discovery research [1, 4, 7, 9, 11, 14]. In document collections without links, link discovery can make use only of the textual content of the documents, e. g. using methods of information retrieval [2, 8].

In this paper, we aim at creating unsupervised link discovery algorithms that work both on document collections that already contain links, as well as on document collections that do not. To channel the research effort in link discovery, there is the Link-the-Wiki track at INEX², in which we participated. In this work, we describe our contributions to the Link-the-Wiki track at INEX 2009, and qualitatively discuss the results.

In the next section, we formally define the task of link discovery and describe related work. In Section 3, we give a brief overview of the tasks in the Link-the-Wiki track, and describe the two document collections used in the track.

¹ <http://www.wikipedia.org>

² <http://www.inex.otago.ac.nz>

In Sections 4 and 5, we detail our link discovery approaches and our results qualitatively. Quantitative results will only be available after INEX 2009.

2 Link Discovery in Document Collections

We distinguish between two types of links in document collections, as shown in Figure 1.

Document-level links relate a source to a target document.

Anchor-level links connect a specific phrase in the source document to a target document. Within the target document, a concrete entry point may be specified, e.g. section headings or paragraphs. They can be represented as character offset in the document.

Formally defined, let \mathcal{D} be the document collection. The goal of link discovery is to connect a source document $s \in \mathcal{D}$ to a target document $t \in \mathcal{D}$ by means of hyperlinks. Such links are denoted by $l(s, t)$ and called document-level links. From the perspective of a single document $d \in \mathcal{D}$, outgoing links are links that have d as source, $l(d, t)$, and incoming links have d as target, $l(s, d)$.

Additionally, we distinguish links more fine-grained: Links that originate from a specific anchor phrase p in s and link to a target document t are denoted by $l(s_p, t)$. Links from a document s to a certain entry point e in t are denoted by $l(s, t_e)$. Link from p in s to e in t , the most specific links, are denoted by $l(s_p, t_e)$. We call both $l(s_p, t)$ and $l(s_p, t_e)$ anchor-level links.

Finally, we define \mathcal{D}_p as the set of documents containing a phrase p and $\mathcal{D}_{l(s_p, t_e)}$ as the set of documents containing the link $l(s_p, t_e)$ where both p and e could be omitted. Documents that do not contain any links at all are called *orphans*.

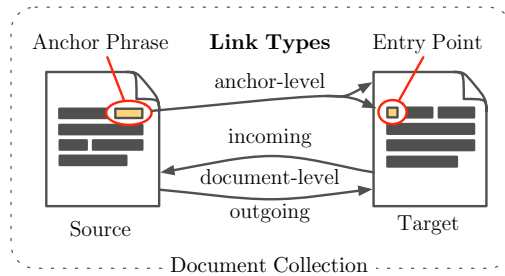


Fig. 1. Link Types in Document Collections

2.1 Anchor-Level Link Discovery

Discovering anchor-level links comprises two tasks: anchor discovery and target discovery. Figure 2 shows a categorization of the first task, Figure 3 of the second one. Anchor phrases that are relevant to the reader of the document, and thus should be connected to further information, need to be identified. In the target discovery step, the best matching target is retrieved. However, if there is no valid target in the collection, the link might be rejected.

Anchor Discovery Anchor discovery is done in two steps: identifying anchor candidates, followed by ranking the candidates. Potential **anchor candidates** are:

- **N-Grams**: term groups of length N, used by Geva [7].
- **Noun phrases (NPs)**: groups of determiners, prepositions, adjectives and nouns, e.g “the president of a country” or “natural language processing”.
- **Document or section titles** extracted from the document structure.

In document collections containing links there is another source of anchor candidates, namely the phrases that have already been used as a link anchor in some document.

A measure to **rank anchors** that does not rely on an existing link structure but on the distribution of terms is tf.idf, which is used by Csomai and Mihalcea [4]. It can be used to rate both single words and phrases consisting of

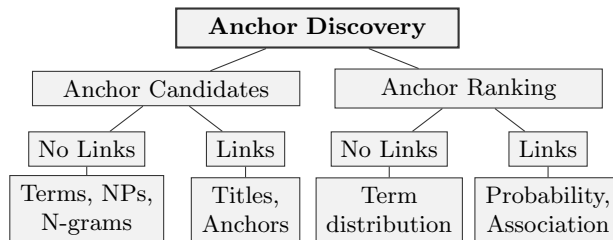


Fig. 2. Discovering Anchors

more than a single word. These multi-word phrases are assigned the same rating as the single word with the maximum tf.idf value inside the phrase.

Ranking anchors can be improved using the information of existing links. Thus, most of the approaches for ranking anchor-level links make heavy use of existing links in the collection. Csomai and Mihalcea [4] propose such a measure to rank the anchor candidates. It is called *keyphraseness*, motivated by the notion that using a phrase as a link anchor is a hint that it is a keyphrase in the document. It rates a phrase p according to the probability of p being used as anchor in a collection. The *keyphraseness* of p is the number of times it was used as link anchor in an article, divided by the total number of articles the phrase appears in. It is calculated as follows:

$$keyphraseness(p) = P(anchor|p) \approx \frac{|\mathcal{D}_{l(s_p,t)}|}{|\mathcal{D}_p|} \quad (1)$$

Using this measure, Csomai and Mihalcea achieve an f-measure of 0.55 in discovering anchors in Wikipedia.

Another measure to rank anchor candidates based on existing links was introduced in the 2007 Link-the-Wiki track by Itakura and Clarke [9]. It rates a phrase p according to the strength a link $l(s_p, t)$ anchored on p with target t is associated with its most frequent target. Let $\mathcal{T} := \{l(s_p, t) | t \in \mathcal{T}\}$ be the set of link targets of a phrase p in the existing document collection, $\mathcal{T} \subseteq \mathcal{D}$. We will first define the *association strength* $as(p, z)$ between p and a specific target $z \in \mathcal{T}$:

$$as(p, z) = \frac{|\mathcal{D}_{l(s_p,z)}|}{\sum_{t \in \mathcal{T}} |\mathcal{D}_{l(s_p,t)}|}$$

This measures the strength of association of phrase p to target s . The measure introduced by Itakura and Clarke takes the *maximum association strength* $as_{\max}(p)$ as the rating of p :

$$as_{\max}(p) = \max\{as(p, t) | t \in \mathcal{T}\} \quad (2)$$

This measure favors phrases with one highly probable link target. Phrases that have multiple equally common targets will get a lower as_{\max} -rating.

Target Discovery In document collections without links, targets can be discovered using all kinds of information retrieval models, simply by searching for the anchor phrase. Targets can either be documents or entry point in documents, depending of the granularity of the search.

In a document collection containing links, the targets of existing links can be used. However, some phrases are ambiguous, i. e. have different meanings, e. g. “bank” can mean “edge of a river” or “financial institution”. In Wikipedia, such phrases have different link targets. Csomai and Mihalcea use this information to train a machine learning classifier to disambiguate targets, achieving an f-measure of 0.87 on Wikipedia.

Milne and Witten [11] propose an approach that **intertwines anchor and target discovery** for Wikipedia links. They train a machine learning classifier for both anchor identification and target disambiguation. One important feature of their target disambiguation classifier is the semantic similarity of two Wikipedia articles, which is measured by comparing article links [10]. The confidence of this disambiguation classifier is used as one of the features for identifying anchors. Among other features for training their anchor classifier are the key-phraseness value (see Equation 1) and the generality of the anchor phrase. Their approach results in an overall f-measure of 0.74 for recreating Wikipedia links.

2.2 Document-Level Link Discovery

One method to discover links on the document level is to generalize anchor-level links. There are also methods that work directly on the document level. They can roughly be classified as shown in Figure 3.

Unlinked Document Collections Allan [2] uses vector-based information retrieval to discover targets, using the document text as query. Green [8] tries to improve link discovery by using a semantic relatedness measure based on WordNet [5] and lexical chains. In his evaluation, the measure is compared to links created by vector-space methods like Allan’s above, but without significant improvements. Chen et al. [3] propose to link documents that have a high number of overlapping frequent phrases, which achieved the best result in discovering incoming links at INEX 2008.

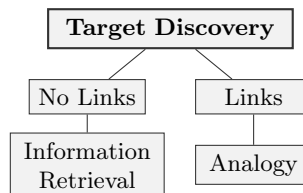


Fig. 3. Discovering Targets

A measure to calculate the semantic relatedness between text was introduced by Gabrilovich and Markovitch [6], the *Explicit Semantic Analysis* (ESA). It can be used to identify similar pages as targets. ESA uses the text of all Wikipedia articles to construct a vector representation of a term, and the semantic relatedness of two terms can then be computed as the cosine of their corresponding vectors. ESA can be modified to work on different corpora, e.g. Wiktionary³ [15].

Linked Document Collections provide additional knowledge based on the assumption that similar pages should have similar links. Adafre and de Rijke [1] employ a two step process: first they identify topically related pages using vector-space search. The second step is to add links that are missing in the source document, but exist in related documents. West et al. [14] discover missing links in documents by reducing the dimensions of an article-link matrix. They argue that the error introduced when reconstructing the original matrix from the dimension-reduced one is a good measure of which links are missing in an article. In their evaluation, they show that humans judge their links better than the ones created by the Milne and Witten [11] approach described above.

3 INEX Link-the-Wiki

We participated in the INEX 2009 Link-The-Wiki track, an international link discovery competition. We participated in two tasks:

Link-the-Wiki Discover incoming and outgoing document-level links for 5000 orphans (existing Wikipedia articles with links removed). Formally, given a document d , the task is discovering links $l(d, \cdot)$ and $l(\cdot, d)$.

Link-Te-Ara Discover outgoing anchor-level links for all Te Ara articles. Formally, given a document d , the task is discovering links $l(d_p, t_e)$ from phrases p in d to entry points e in target document t .

Our main interest in this challenge is to research unsupervised methods of discovering links in document collections. The Wikipedia is a good collection to test methods that rely on an existing, well-gardened link structure, whereas the Te Ara — without any links at all — serves for testing methods that have to rely on the textual content alone.

3.1 Wikipedia

Wikipedia is a large-scale, general-purpose encyclopedia. The articles are collaboratively created by a very active community, and the community also maintains a very dense and well-gardened link structure connecting the articles.

³ <http://www.wiktionary.org>

For the INEX challenge, an XML dump of the English Wikipedia was provided⁴. It comprises 2,666,190 individual encyclopedia articles that have been converted from wiki syntax to semantically enriched, well-formed XML documents [12]. Structural elements like sections, paragraphs, etc. are preserved. The annotated XML data accounts for 50.7 GB in size.

3.2 Te Ara

Contrary to Wikipedia, which is covering general knowledge, the *Te Ara Encyclopedia of New Zealand*⁵ solely focuses on topics related to matters and facts of New Zealand. The texts are authored by local experts of the Ministry of Culture and Heritage⁶.

For the challenge, an XML dump of the Te Ara Encyclopedia was provided. It is a lot smaller than Wikipedia, comprising only 438 handpicked articles, containing a total of 3180 XML files (due to each individual article usually comprising more than one file).

The structure of Te Ara’s XML files differs from the Wikipedia collection. Each article is not represented by a single self-contained file, but rather consists of a main file along with one or more resource description documents for multimedia content. Same as the Wikipedia dump, structural elements like headings or paragraphs are annotated in the textual parts of the files. The main difference to the Wikipedia collection is that Te Ara texts do not include any reference links to related articles other than links to resources.

The main file is comprised of a metadata header, an article abstract, and followed by a numbered list of subentries. Subentry, though, is misleading in this case: It actually addresses a certain part of the article and is intended for being rendered on a separate web page. Each subentry encloses an internal index number, a title element and a body section. Any resources—like photos along with their titles and descriptions—are described in separate XML files which are named according to the subentry index they refer to.

4 Link-the-Wiki: Wikipedia Document-Level Links

4.1 Discovering Outgoing Links

To discover outgoing links, we first identify potential anchors, followed by an appropriate target for each anchor, making use of the existing link structure. We combine the *keyphraseness* and as_{\max} measure to rank anchor candidates, and expect this to improve the ranking quality, as the measures are quite complementary: *keyphraseness* prefers phrases that are often used as anchor, and as_{\max} prefers phrases that have one highly probable link target. To disambiguate between potential targets, we use the ESA semantic relatedness measure based on

⁴ Snapshot taken on Oct 8, 2008

⁵ <http://www.teara.govt.nz>

⁶ <http://www.mch.govt.nz>

<i>UKP-LTWF2F</i>	<i>Anchor Ranking</i>			<i>Target Ranking</i>
Experiment ID	Keyphraseness	as_{max}	TF.IDF	Frequency ESA
out_k_esa	•			•
out_s_esa		•		•
out_sk_esa	•	•		•
out_sk_freq	•	•		•
out_tfidf_freq			•	•

Table 1. Configurations for Discovering Outgoing Document-Level Links in Wikipedia

Wiktionary [15], to capture the article relatedness on a conceptual level. As the measure is based on general knowledge provided by a knowledge base (in this case Wiktionary), we do not need to train it for a specific document collection.

Experiment Configurations We run different combinations of candidate and target identification in each experiment to discover outgoing document-level links in Wikipedia, Table 1 gives an overview. For each orphan, we perform the following steps:

1. **Preprocessing:** Tokenize, lemmatize⁷, and remove stop-words.
2. **Annotate anchor candidates:** Each word or phrase in the orphan article that corresponds to a Wikipedia article title (minus the disambiguation string, denoted as the part in braces in the title) or has been used as link anchor in Wikipedia at least five times (like in [4]).
3. **Merge anchor candidates:** Overlapping anchor candidates are merged, removing all anchors that are fully contained in another candidate.
4. **Rank anchor candidates:** Using *keyphraseness* (denoted as *k* in the run id), *as_{max}* (*s* in the run id), the arithmetic mean of *keyphraseness* and *as_{max}* (*sk* in the run id), or *tf.idf*, denoted as *tfidf*.
5. **Identify targets:** Potential targets are all link targets of the phrase in the existing collection.
6. **Rank targets:** Take the most frequent target in the collection, denoted as *freq*, or compare the orphan text to all potential target texts using ESA, denoted as *esa*.
7. **Generalize to document level:** Take the highest ranked link target of the highest ranked anchors, until 250 distinct targets have been accumulated.

4.2 Discovering Incoming Links

To discover incoming links, we run two experiments. In the run with the id *UKP-LTWF2F_in_lucene*, we execute a full-text search for the article title (without the disambiguation string) using the standard Lucene⁸ retrieval model, and

⁷ Using the TreeTagger [13]

⁸ <http://lucene.apache.org>

take the top 250 results as sources of incoming links. In our second experiment, *UKP-LTWF2F_in_esa*, we re-rank the top 2000 search results returned by the Lucene search using ESA, taking into account the semantic relatedness between the orphan and potential source. Due to the size of Wikipedia and the computational effort needed to do ESA comparison, we could not use ESA for the complete retrieval process.

4.3 Qualitative Results

The heuristic of using the most frequent link target for a given anchor works fine for many link anchors, but lacks the ability to adapt to a given context. This is why we used an ESA ranking model which also includes context information as described in Section 2.2. Consider, for example, the Wikipedia article about *Bülent Arınç*, a Turkish politician. In the article text he is said to be born in a city called *Bursa* in Turkey. The most frequent target for this phrase is the article *Bursa Province*, not the city with the very same name. ESA, on the contrary, identifies the correct target.

Infrequently, both models identify inadequate documents as potential targets. For example, the word *track* in context of a Silverstone Formula One race is linked to an article about *track cycling*, a bicycle racing sport, instead of *Silverstone Circuit*, both by using the most frequent target and using ESA.

In some cases, ESA even performs worse than the baseline. A sample sentence is “*As such, it is used for cervical cancer screening in gynecology.*”—with the underlined word being our link candidate. The baseline approach links to *Screening (Medicine)* which is the correct target. ESA provides a higher value to the *Halftone* article instead, which describes a graphics reproduction technique.

In conclusion, ESA seems to work as expected to disambiguate targets, but not in all cases. The context for ESA comparisons is the whole article where the anchor appears, which might be too much — this will be subject to further research.

5 Link-Te-Ara: Discovering Anchor-Level Links

In this task, the goal is to create links for the complete collection. Outgoing links for all documents include incoming links, so we do not need to distinguish anymore. In Wikipedia we gathered anchor candidates using existing links. To make up for the smaller number of candidates — because of missing links in Te Ara — we also use noun phrases as anchor candidates. We use ESA as measure to discover good link targets, which is expected to improve discovery in cases where bag-of-word approaches fail because of the vocabulary gap.

5.1 Experiment Configurations

We run different combinations of anchor identification and target ranking to discover anchor-based links, Table 2 gives an overview. For each document in the Te Ara collection, we do the following:

<i>UKP-LTAraA2B</i>	<i>Anchor Candidates</i>		<i>Target Identification</i>		<i>Target Ranking</i>	
Experiment ID	Titles	Noun Phrases	Titles	Full Text	Lucene	ESA
c_esa	•		•			•
nc_esa		•	•			•
cnc_esa	•	•	•			•
cnc_lucene	•	•	•		•	
cnc_lucene_full	•	•		•	•	

Table 2. Configurations for Discovering Anchor-Level links in Te Ara

1. **Preprocessing:** Tokenize, PoS tag and noun chunk⁹, and remove stop-words.
2. **Annotate anchor candidates:** Document and section titles (denoted as c in the run id), noun phrases in the document (denoted as nc), or a combination of both (denoted as cnc) are anchor candidates. All XML title elements in Table 2 are taken as titles to annotate. The annotation is restricted to the content XML elements in Table 2, on the premise that anchors should only appear in the content parts of a document.
3. **Merge anchor candidates:** Overlapping anchor candidates are merged, removing all anchors that are wholly contained in another candidate.
4. **Rank anchor candidates** using tf.idf.
5. **Remove superfluous anchors:** Take the best 50 or 6% of the number of terms in the document, whichever is less. The Link-Te-Ara task limits the anchor links to a maximum of 50, and for shorter documents, we limit this number even more (in accordance with [4]).
6. **Target identification:** Search for each anchor phrase, either in the titles (see XML elements in Table 2), or in the complete documents. When searching in titles only, the title’s position is the entry point, when searching in complete documents, the entry-point is set to 0.
7. **Target ranking:** Using the Lucene¹⁰ retrieval model (denoted as *lucene* for title-search, *lucene_full* for document-search), or using ESA to compare anchor phrases to all titles (denoted as *esa*). We take the top 5 results as targets.

5.2 Qualitative Results

Anchor Identification A snippet of the results of our Te Ara anchor identification is shown in Figure 4, the first paragraph of the article *Wine*.

All annotated phrases, except *licensing*, are good anchor candidates. The noun phrases are more precise, though, describing more specific concepts. Whereas the titles only allow for the identification of the term *industry*, the noun phrase

⁹ Using the TreeTagger [13]

¹⁰ <http://lucene.apache.org>

Type	Elements	Namespace
Title	/Entry/Name //SubEntry/Name	enz.govt.nz/Entry
	//EnglishName	enz.govt.nz/Resources
	//TopicBox/Heading	enz.govt.nz/SubEntrySectionElements
	//h3	w3.org/1999/xhtml
Content	//p //li //blockquote	w3.org/1999/xhtml
	//Text	enz.govt.nz/Entry

Table 3. Title and Content XML Elements in Te Ara

Sauvignon blanc, with its grassy smell, put *New Zealand wine* in the international spotlight in the 1980s. Since then, wine *exports* have boomed, with pinot noir another big hit. But for many years, tough *licensing* laws and New Zealanders’ taste for *fortified wines* limited the wine *industry*.

Fig. 4. Comparison of noun phrases (*nc*) to *extracted candidates* (*c*) in the abstract of the article “Wine”

is in this case more apt to the topic of the overall article: the wine industry. In the experimental configuration where we use both types of candidates, only the noun phrase remains. This is because anchor candidates are merged, and the longer phrase is preferred. Licensing is ranked too low by tf.idf, and not annotated as anchor in the combined run, showing that combining both candidates can improve the result.

Identifying good anchors is important in itself, but even if the anchors seem to be a good fit, we still have to take into account that each anchor needs a valid target. The term “fortified wine” in the example of Figure 4 seems reasonable, because it is an important concept and it would be a good idea to provide further information about it. In the Te Ara, though, no definition for the term is available — it is a good example were linking to a general encyclopedia like Wikipedia would make sense. An approach to improve this might be to use terminology extraction algorithms for ranking the candidates, as they focus on term domain specificity, and not only importance. This will be subject to further research.

Target Identification We will now discuss all three methods exemplarily using the anchor “invasive species” taken from the *Biosecurity* article. When executing a full-text Lucene search for the query “invasive species”, the *Marine invaders* article is the first target, which discusses invasive species from the seas. It is relevant to the anchor, and although does not cover the generic concept of invasive species but rather a special case, it includes a perfect definition of the generic concept of “invasive species”. The other targets are also relevant.

Restricting the search to titles only, the second target includes the section “Introduced and invasive species” in the *Marine invaders* article, a perfect definition for the term. The last target, “Endemic species” in the article *Butterflies*

E. moths, though, is unrelated to the “invasive” part of the query. Here, a problem of vector-space based search becomes evident — partial matches of the title are often misleading, even though the heads of the noun phrases match.

The last target identification method, semantic information retrieval, has “More new species?” in the *Acclimatization* article as target, which is specifically about the problem of invasive species, and thus very relevant as background information. The rest of the results are only related to a part of the query, namely “species”, and not relevant. Semantic information retrieval suffers from the same problems as the Lucene search restricted to titles, as detailed above.

To conclude, we can say that full-text search works to get relevant article targets, but the results are sometimes too broad. Restricting the search to titles remedies this, but the restriction introduces more results that are irrelevant to the anchor text query. Semantic information retrieval does not seem to improve this lack of context, but we have to wait for the results of the evaluation before we draw any final conclusions. For future experiments, one could try to run ESA with more contextual information, e.g. the text following the headings, and use the Wikipedia index instead of the Wiktionary one.

6 Summary

In this paper, we presented our experimental runs at the Link-the-Wiki track at INEX 2009. We participated in two of the tasks: Discovering document-level links in Wikipedia orphans and discovering anchor-level links in the Te Ara. For Wikipedia, we combined *keyphraseness* and *maximum association strength* for anchor ranking with Explicit Semantic Analysis for target disambiguation. In Te Ara, which in contrast to Wikipedia contains no links, we used natural language processing to identify anchor candidates. For identifying entry points, we restricted the search to the article and section titles, using Lucene and Explicit Semantic Analysis to rank them. We concluded with a qualitative discussion of our results, as the evaluation results were not available at the time of writing this paper.

7 Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806, and by the Klaus Tschira Foundation under project No. 00.133.2008.

References

1. Adafre, S.F., de Rijke, M.: Discovering Missing Links in Wikipedia. In: LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery. pp. 90–97. ACM, New York, NY, USA (2005)
2. Allan, J.: Building Hypertext Using Information Retrieval. *Information Processing & Management* 33(2), 145–159 (1997)

3. Chen, M.L.E., Nayak, R., Geva, S.: Link-the-Wiki: Performance Evaluation Based on Frequent Phrases pp. 326–336 (2009)
4. Csomai, A., Mihalcea, R.: Linking documents to encyclopedic knowledge. *IEEE Intelligent Systems* 23(5), 34–41 (2008)
5. Fellbaum, C.: *WordNet: An Electronic Lexical Database*. MIT Press (1998)
6. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In: *Proceedings of The Twentieth International Joint Conference for Artificial Intelligence*. pp. 1606–1611. Hyderabad, India (2007)
7. Geva, S.: GPX: Ad-Hoc Queries and Automated Link Discovery in the Wikipedia. In: Fuhr, N., Kamps, J., Lalmas, M., Trotman, A. (eds.) *INEX. Lecture Notes in Computer Science*, vol. 4862, pp. 404–416. Springer (2007)
8. Green, S.J.: Building hypertext links by computing semantic similarity. *IEEE Transactions on Knowledge and Data Engineering* 11(5), 713–730 (Sep 1999)
9. Itakura, K.Y., Clarke, C.L.A.: University of Waterloo at INEX2007: Adhoc and Link-the-Wiki Tracks. In: *Focused Access to XML Documents. Lecture Notes in Computer Science*, vol. 4862, pp. 417–425. Springer (2008)
10. Milne, D., Witten, I.H.: An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links. In: *Proceedings of the AAAI 2008 Workshop on Wikipedia and Artificial Intelligence (WIKIAI 2008)*. Chicago, IL (2008)
11. Milne, D., Witten, I.H.: Learning to Link with Wikipedia. In: *Proceedings of the 17th ACM Conference on Information and Knowledge Mining*. pp. 509–518. ACM, New York, NY, USA (2008)
12. Schenkel, R., Suchanek, F., Kasneci, G.: YAWN: A semantically annotated Wikipedia XML corpus. In: *12. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web (BTW 2007). Lecture Notes in Informatics*, vol. 103, pp. 277–291. Gesellschaft für Informatik, Aachen, Germany (2007)
13. Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In: *Proceedings of the International Conference on New Methods in Language Processing*. pp. 44–49 (1994)
14. West, R., Precup, D., Pineau, J.: Completing Wikipedia’s Hyperlink Structure through Dimensionality Reduction. In: *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*. pp. 1097–1106. ACM, New York, NY, USA (2009)
15. Zesch, T., Müller, C., Gurevych, I.: Using Wiktionary for Computing Semantic Relatedness. In: *Proceedings of AAAI*. pp. 861–867 (2008)

Link Discovery in the Wikipedia

Shlomo Geva¹, Andrew Trotman², Ling-Xiang Tang¹,

¹ Faculty of Science and Technology,
Queensland University of Technology,
Brisbane, Australia
{s.geva, l4.tang}@qut.edu.au

² Department of Computer Science,
University of Otago,
Dunedin, New Zealand
andrew@cs.otago.ac.nz

Abstract. In this paper we describe our approaches taken in the Link-the-Wiki track. We submitted runs for all three Link-the-Wiki tasks: Link-the-Wiki, Link-Te-Ara, and Link-Te-Ara-to-the-Wiki. To generate outgoing links for each task, our link discovery system employs the top ranking algorithms from previous LTW tracks and a hybrid method derived from them. For incoming links, we used traditional information retrieval strategy on the Wikipedia XML collection. The official results for the INEX 2009 Link-the-Wiki track show encouraging performance of our system.

Keywords: Wikipedia, Link Discovery, Best Entry Point.

1 Introduction

We submitted runs for all Link-The-Wiki tasks: Link-the-Wiki, Link-Te-Ara, and Link-Te-Ara-to-the-Wiki. The Link-the-Wiki task requires the identification of incoming links and outgoing links at both file-to-file and anchor-to-bep levels; the Link-Te-Ara requires the identification of anchor-to-bep outgoing links for all documents in the Te Ara Encyclopedia; and the Link-Te-Ara-to-the-Wiki requires the identification of anchors in Te Ara pages and their corresponding BEPs in the Wikipedia corpus.

The algorithms we used to generate the outgoing links for each task are based on the top-ranking link generation algorithms from previous years: Itakura's algorithm; and Geva's algorithm. For the incoming links, we used the topic title as a query to a search engine and took the top ranked results.

To place BEPs, we tried two different approaches: one was to place the BEP at the location of the anchor phrase in the target document (where the entire phrase, or part of it, appears); the other was to set the BEP to the beginning of the text block which contains terms similar to those of the text surrounding the anchor in the source document.

2 Link-The-Wiki Track

Regardless of the task the procedure for automated link discovery is the same and includes the following steps: identifying anchors, recommending a group of outgoing and incoming links, and locating BEPs for these links.

2.1 Anchor Identification

Anchor identification is the first step in link discovery. Identifying anchors can be done using two methods: best match; and partial match.

After an anchor is identified, a link, $a \rightarrow d$, can be created. In this case a is an anchor and d is a corresponding target document.

2.2 Link Recommendation

2.2.1 Outgoing Links

Good algorithms for recommending outgoing links were seen at the first INEX Link-the-Wiki track in 2007 [1]. This year we employed the top two ranking algorithms from that (and subsequent) Link-the-Wiki tracks: Itakura's link mining (ICLM) algorithm [2] and Geva's page name matching (GPNM) algorithm [3]. The INEX 2009 document collection is, however, a much larger collection than the previous collection.

The ICLM algorithm relies on the pre-existing link graph in the Wikipedia. The anchor text (a) target document (d) pairs are all extracted from the collection. The document frequency of each anchor text is then computed. The algorithm proceeds by finding all anchor texts that exist within the orphaned topic document and ranking those on an anchor weight, γ :

$$\gamma = \frac{\text{number of pages that has link}(a \rightarrow d)}{\text{number of pages that has text of anchor}(a)} \quad (1)$$

The GPNM algorithm generates a table containing the title and the id of each document in the collection. A sliding window with size varying from 1 to 12 terms is then run over the orphan topic document looking for titles from the table. These are ranked on target document title length; longer anchors having higher scores.

In an attempt to improve the performance of both algorithms we combined them into one. First, two sorted lists of outgoing links are created separately using the two algorithms. Links are assigned a score using following method:

$$\text{Score}(L) = \text{Score}_s(L) + \text{Score}_k(L) \quad (2)$$

Where $\text{Score}(L)$ is the score for link L , $\text{Score}_s(L)$ is the score from the GPNM algorithm and $\text{Score}_k(L)$ is a normalized ICLM γ :

$$Score_k(L) = \frac{\max(\gamma) - \min(\gamma)}{N} \times m \quad (3)$$

Where $\max(\gamma)$ is the highest γ value of any link in the GPNM list; $\min(\gamma)$ is the lowest γ value of any link in the list; N is length of the longest anchor in the list, and m is the number of terms in L .

Finally, links are ranked on $Score(L)$. Either $Score_s(L)$ or $Score_k(L)$ might be zero if the anchor link only appears in one of the lists. Scores for links in both lists are boosted. The first 250 (for F2F) or 50 (for A2B) links of the list are selected as the links to return.

2.2.2 Incoming Links

Finding incoming links for a topic document is performed by retrieving the first 250 pages returned from a BM25 search engine. The orphaned document's title was used as the query terms.

2.3 BEP Location

Best entry points (BEPs) play an important role in providing readers direct access to relevant document passages [4]. However, deciding where the BEP should be is a difficult focused retrieval problem.

Our first approach to find the BEP is to find the first location of the anchor terms. There are two scenarios:

- If there is an exact match for the anchor in the destination page, the BEP is the offset of the first occurrence.
- If there is no exact match, then we use the location of the first term from the anchor text. If that cannot be found then we move on to the second term, and so in until a term is found. If no term is found then the start of the document is used.

Our second approach was a technique similar to that used in image matching. Given two images, the more features in those images that match, the more certain we can be that the two images depict similar objects. BEP finding can be treated as a feature finding problem. First, a text window of length 200 characters surrounding the anchor in the source document is used for creating a source text template. Terms are identified and Porter stemmed. These terms are the features. Next, a sliding window of the same length is passed over the target document, and features are extracted similarly. A score is calculated for the window by counting the number of matching features (stemmed terms). The window is moved forward 100 characters at a time and the score calculation for matched features is repeated. The beginning of the text block with the highest score is chosen as the BEP.

3 Link-the-Wiki Experiments

3.1 Link-the-Wiki Runs

Table 1. Link-the-Wiki runs.

Run name
QUT_LTW_F2F_SEA_BASELINE01
QUT_LTW_F2F_SEA_BASELINE02
QUT_LTW_F2F_SEA_BASELINE03 (unofficial)
QUT_LTW_F2F_SEA_01(disqualified)
QUT_LTW_F2F_SEA_02
QUT_LTW_F2F_SEA_04 (unofficial)
QUT_LTW_F2FonA2B_SEA_03(unofficial)
QUT_LTW_A2B_SEA_BASELINE01
QUT_LTW_A2B_SEA_BASELINE02
QUT_LTW_A2B_SEA_01
QUT_LTW_A2B_SEA_02

We submitted runs in this task at both levels: file-to-file (F2F) and anchor-to-bep (A2B). All the baseline runs (with BASELINE0X suffix) were created using the ICLM algorithm; and the other runs (with SEA_0X suffix) were generated using our new algorithm that combines ICLM and The GPNM algorithms.

The link table from ICLM algorithm included links in all pages including the orphan document (before it was orphaned) and so the link information in the orphan was removed from the table when calculating γ scores. In order to determine the impact of this necessary correction we submitted runs without the correction ("01" suffix). These runs are "cheating". All other runs are correctly orphaned.

Table 2. Link-Te-Ara runs.

Run name
QUT_LTAra_A2B_SEA_BASELINE01
QUT_LTAra_A2B_SEA_BASELINE02

The baseline runs for Link-Te-Ara task were generated using the GPNM algorithm. The Te Ara document format is very different from that of the Wikipedia corpus. In a Te Ara page, there is no unique tag for the page title. For example, there may be only one <Name> tag, or many <*Name> tags.

A name to document pairing table for these name tags was created. The difference between Link-Te-Ara BASELINE01 and BASELINE02 lies in BEP identification: BASELINE01 uses the term matching technique for BEP identification; while BASELINE02 uses the text template matching technique.

Table 3. Link-Te-Ara-to-Wiki runs.

Run name
QUT LTArATW A2B SEA BASELINE01

We only submitted only one run for Link-Te-Ara-to-Wiki task. This run was created using ICLM algorithm.

Separately, we submitted two further unsuccessful runs. The first was the ICLM algorithms: OTAGO_LINKPROBABILITY_A2B. The second was a modified ICLM that was generated by taking a proxy log of university of Otago student Wikipedia use and augmenting γ with a weight based on the number of times the link was clicked by a user: OTAGO_LINKPROBABILITYANDCLICKRATE_V1_A2B.

In this second run the new γ (η) was computed thus:

$$\eta = \gamma \times \frac{\text{number of anchor text clicks}}{\text{number page(with the anchor text)views}}$$

Unfortunately this second experiment was unsuccessful due to implementation issues. We will further this line of investigation in future work.

3.2 Link-the-Wiki Results

Since there are (at time of writing) no manual assessments or ground-true for the Link-Te-Ara task and the Link-Te-Ara-to-Wiki task, only the results from Link-the-Wiki task are discussed in this section.

The evaluation results for outgoing links on the file-to-file and anchor-to-bep topics are presented in figures 1 and 2 respectively. The results for incoming links are presented in figures 3 and 4. The results shown in these four figures are against the automatic assessments (links in the topic documents before orphaning). The results of our unofficial runs are included for comparison.

Among our runs which are correctly orphaned the best ones, marked as black curves shown in all the plots, indicate encouraging performance of our system. Figure 1 demonstrates that our run QUT_LTW_F2F_SEA_BASELINE02 has the highest score (for the correctly orphaned topic) run. However, run QUT_LTW_F2F_SEA_02 using the new algorithm has lower accuracy than the baseline runs have, even though still higher than others.

In figure 2 our system is out-performed by Waterloo's run in submissions for 50 outgoing links in file-to-file level on 33 topics for anchor-to-bep task.

Figure 5 presents the precision-recall curves for the unofficial runs of three different systems. The curve with the highest accuracy is from the Otago's system with the implementation using the ICLM algorithm. This figure indicates there might be a faulty in implementing ICLM algorithm in our system, since our run QUT_LTW_F2FonA2B_SEA_03 achieves lowest accuracy in terms of precision and recall comparing with others.

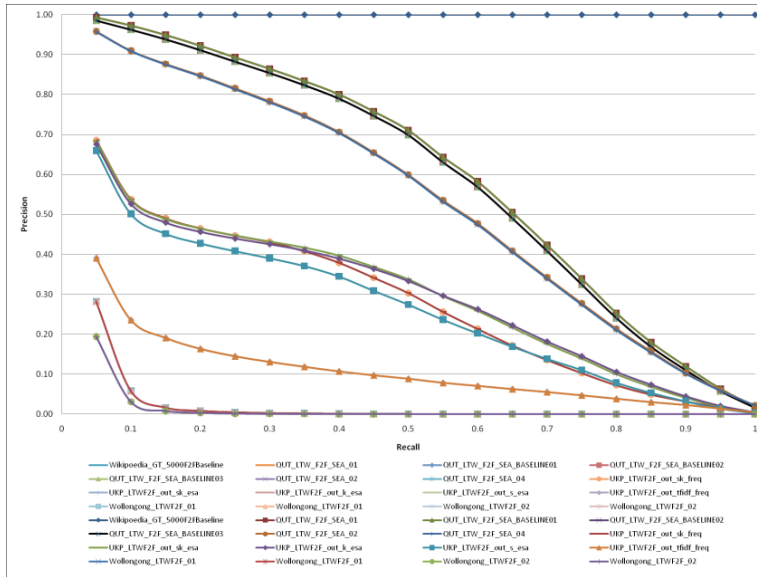


Fig. 1. Link-the-Wiki automatic outgoing F2F assessment on 5000 F2F topics.

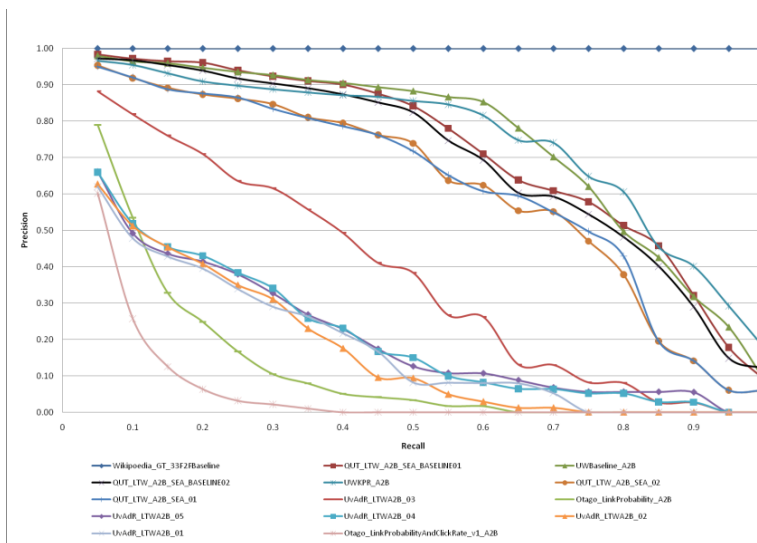


Fig. 2. Link-the-Wiki automatic outgoing F2F assessment on 33 A2B topics.

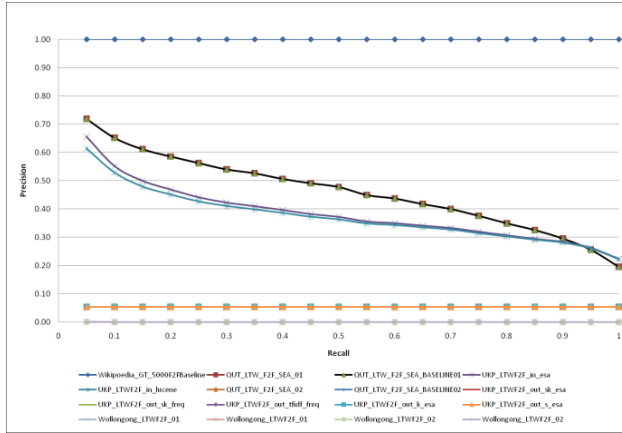


Fig. 3. Link-the-Wiki automatic incoming F2F assessment on 5000 F2F topics.

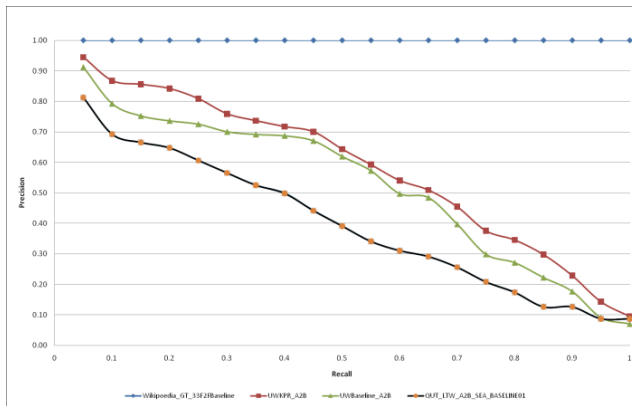


Fig. 4. Link-the-Wiki automatic incoming F2F assessment on 33 A2B topics.

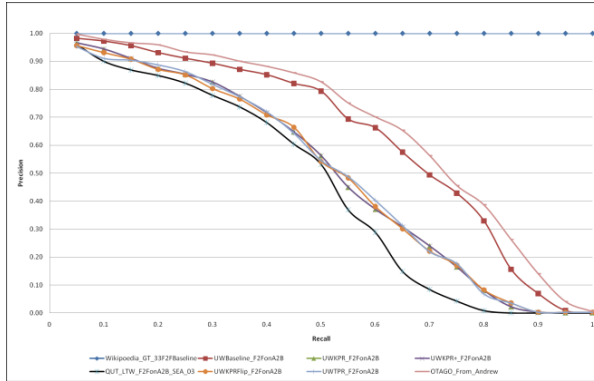


Fig. 5. Unofficial automatic outgoing F2F assessment on 33 A2B topics.

4 Conclusions and Future Work

In our link discovery system, we implemented the ICLM algorithm proposed in the first LTW track, and a new method that combines both ICLM and GPNM algorithms to generate outgoing links. Our results from the official evaluation of outgoing and incoming links show reasonable good performance of our system. Using traditional information retrieval technique on Wikipedia XML collection for creating incoming links is every effective.

The new hybrid method for recommending outgoing links doesn't work as well as the original ICLM algorithm. Finding out the reason for the degraded performance of the hybrid approach could be treated as our remaining task for next round of Link-the-Wiki evaluations in 2010.

References

1. Huang, D., Xu, Y., Trotman, A., Geva, S.: Overview of INEX 2007 Link the Wiki Track. Focused Access to XML Documents 373-387 (2008)
2. Itakura, K., Clarke, C.: University of Waterloo at INEX2007: Adhoc and Link-the-Wiki Tracks. Focused Access to XML Documents 417-425 (2008)
3. Geva, S.: GPX: Ad-Hoc Queries and Automated Link Discovery in the Wikipedia. Focused Access to XML Documents 404-416 (2008)
4. Reid, J., Lalmas, M., Finesilver, K., Hertzum, M.: Best entry points for structured document retrieval--Part I: Characteristics. Information Processing & Management 42 74-88 (2006)

QA@INEX 2009: A common task for QA, focused IR and automatic summarization systems

Veronique Moriceau¹, Eric SanJuan², and Xavier Tannier¹

¹ LIMSI-CNRS, University Paris-Sud 11
{moriceau,Xavier.Tannier}@limsi.fr

² LIA & IUT STID, Université d'Avignon
eric.sanjuan@univ-avignon.fr

Abstract. QA@INEX aims to evaluate a complex question-answering task. In such a task, the set of questions is composed of factoid, precise questions that expect short answers, as well as more complex questions that can be answered by several sentences or by an aggregation of texts from different documents. Question-answering, XML/passage retrieval and automatic summarization are combined in order to get closer to real information needs.

1 Introduction

The INEX 2009 QA@INEX track aims to compare the performance of QA, XML/passage retrieval and automatic summarization systems on an encyclopedic resource (Wikipedia). The track considers two types of questions: factual questions which require a single precise answer to be found in the corpus if it exists and more complex questions whose answers require the aggregation of several passages. For example, this is the case for questions expecting an answer composed of several items. Current evaluation campaigns artificially restrict list of questions to items present in the same sentence. The reason is that traditional QA systems are not designed to merge answers from different sources, and that human assessment would be made quite harder without this restriction. However, this corresponds to an important user need in several manners:

- Compiling different elements scattered in the collection into a single list of items;
- Finding several valid answers to a single question (“Who is Nicolas Sarkozy?” leads to “French president”, “former french interior minister”, “Carla Bruni’s husband”, etc.);
- Gathering different answers with different restrictions: temporal (“Who is the French president?”: “Jacques Chirac” from 1995 to 2007, “Nicolas Sarkozy” from 2007), spatial or others.

This is also the case of more complex questions that have not been studied in details so far: see [1], [2, 3] for why questions and [4] for opinion questions).

Questions concerning procedures (in short, “how” questions), reasons (“why”) or opinions can hardly find a complete answer in a single part of a document. For example, concerning opinion questions, a QA system should be able to locate opinions in documents and to produce or generate a synthetic “answer” in a suitable way.

A extended range of evaluation methods are used to compare QA vs focused IR when a short answers is required (§2) and QA vs summarization systems by extraction on aggregated answers (§3).

2 Short answers

Short parts of text (one or a very few words) are the usual way to answer questions in so-called question-answering systems. Mostly, answers are named entities (person, date, number... answering to factual questions) or short nominal phrases, often representing a definition (*Who was Kurt Cobain?* → *the leader of Nirvana*; *What is Linux?* → *an operating system*).

The results are presented as a ranked-list of answers together with an explanation passage or element involving the answer. Therefore participants need to provide:

- A small ordered set (10) of non overlapping XML elements or passages that contains a possible answer to the question.
- For each element or passage, the position of the answer in the passage. They are evaluated by computing their distance to the answer.

This evaluation methodology differs from traditional QA campaigns, where a short answer must be provided besides the supporting passage. This is a major difference in terms of metrics used to rank the participating systems.

In traditional campaigns, an important technical issue for QA system is the boundaries of the short answer in the passage. In the quite simple question *Who is Javier Solana?*, the following passage would be relevant:

Javier Solana, the Secretary General of NATO, has just announced that the bombing of Yugoslavia may start as soon as the next few hours.

A system answering only “the Secretary General” (skipping “of NATO”) as a short answer would be penalized for its incomplete (or *inexact*) answer.

However, this metric does not correspond to a real user need. In a end-user QA application, the obvious way to exhibit the answer is to point directly towards it into the supporting text. In this situation, the user does not need a perfect segmentation of the answer, but rather a good entry point inside the text. He/she is able to estimate the full answer by him/herself, by reading the text surrounding the entry point.

For this reason, we suggest to assess a good answer not through the full/incomplete paradigm, but rather by the distance between the indicated answer entry point and the real one.

This new way to evaluate QA systems has an interesting side effect: it allows focused IR systems to participate in this task using the same evaluation, even if

they are unable to extract a short answer or if they have very basic techniques to do so. These systems may simply provide the most relevant and short extracted passages they retrieve, and set an entry point wherever they can in this text. This makes then possible the junction between QA, XML retrieval and other focused IR systems.

3 Long answers

INEX has a thorough experience in evaluating focused retrieval systems, however the QA the “long answer” subtask is new in this context.

Following the first edition of Text Analysis Conference (TAC)³, that brings together QA and automatic summarization, the idea here is to propose a common task that can be processed by three different kind of systems: QA systems providing list of answers, automatic summarization systems by extraction and focused IR systems.

In this QA task, answers have to be built by aggregation of several passages from different documents on the Wikipedia. The questions themselves can be the same as in the short answer task. Let us consider again the previous example “Who is the leader of Nirvana”. The difference with the short answer task is that here we require a short readable abstract of all the information in the Wikipedia related to this question. In this example, the abstract could not only involve references to iconic leader singer of this pop music group, but also on the group itself, on the other members that assumed part of the leadership and that heavily influenced the music style. Passages that explain the terms of the question can also be relevant, by example, why and who decided to take the name of Nirvana for this band.

The maximal length of the abstract being fixed, the systems have to make a selection of the most relevant information. Standard QA systems can produce a list of answers with their support passages. Focused IR systems can return the list of the most relevant XML elements. Note that in this task, IR systems that only retrieve entire documents are strongly handicapped, except if they are combined with automatic summarization systems that builds an abstract of the most relevant documents.

Two main qualities of the resulting abstracts need to be evaluated: readability and informative content.

The readability and coherence is evaluated according to “the last point of interest” in the answer which is the counterpart of the “best entry point” in INEX ad-hoc task. It requires a human evaluation where the assessor indicates where he misses the point of the answers because of highly incoherent grammatical structures, unsolved anaphora, or redundant passages.

The informative content of the answer has to be evaluated according to the way they overlap with relevant passages that will be assessed by participants as in the INEX ad-hoc task. For that we plan to apply recent results on automatic

³ <http://www.nist.gov/tac/publications/2008/>

summary evaluation based on the source text. Given a list of relevant passages, these passages can be whole Wikipedia articles, we intend to compare the word distributions in these passages with the word distribution in the long answer following the experiment in [5] done on TAC 2008 automatic summarization evaluation data. This allows to directly evaluate summaries based on a selection of relevant passages without requiring reference summaries written by experts as in TAC. Indeed, such manual summaries based on such large corpus as the Wikipedia would be very difficult to produce. Therefore, this long answer task is a first tentative of evaluating summarization tools on large data.

Given a set R of relevant passages and a text T , let us denote by $p_X(w)$ the probability of finding a word w from the Wikipedia in $X \in \{R, T\}$. We use standard Dirichlet smoothing with default $\mu = 2500$ to estimate these probabilities over the whole corpus.

The two metrics of distributional similarity that we implemented in a perl program are the following. The perl program applies these metrics after stemming of the words and relies on an Indri index to estimate a priori probabilities.

– Kullback Leibler divergence:

$$KL(p_T, p_R) = \sum_{w \in R \cup T} p_T(w) \times \log_2 \frac{p_T(w)}{p_R(w)}$$

– Jensen Shannon divergence:

$$JS(p_T, p_R) = \frac{1}{2}(KL(p_T, p_{T \cup R}) + KL(p_R, p_{T \cup R}))$$

Since all answers have to be extracted from the same INEX corpus, we can use smoothing methods that allow to avoid null probabilities. Therefore KL is well founded. JS allows to reduce the impact of smoothing parameters since it is always defined. In [5] this is the metric that obtained the best correlation scores with ROUGE semi automatic evaluations of abstracts used in DUC and TAC. However, since we can compute these probabilities by taking the INEX corpus as referential for the probabilistic space, KL metric should also perform well in this track.

We also implemented the standard cosine distance.

4 Status of the track and time line

We intend to run this track over two years (2009 - 2010). The track is open to new participant teams.

2009 has been devoted to fix the tasks and the overall evaluation methodology based on the corpus, topics and qrels from INEX 2009 ad-hoc track. A first list of questions have been released for test. They all deal with 2009 INEX topics. Hence answers should be part of ad-hoc relevant passages. The process of annotating correct answers among passages is on going by organizers and actual participants.

Based on that we intend to release the software to automatically evaluate content selection for this first set of questions. Results from baseline systems proposed by actual participants will be also released. The track is still open to the submission of baseline runs and each participant is invited to submit at least one. In order to facilitate submissions from Focused IR systems, a perl program that converts a run in INEX ad-hoc submission FOL format into QA format is available.

This will allow to fix in accordance with participants all parameters to be used in metrics to evaluate answers content. In particular, the results of *KL* and *JL* metrics for all submitted baseline systems will be available for different smoothing parameters.

In 2010, we shall use the same corpus but participants will be invited to submit a new set of questions on the wikipedia. These questions will not necessarily be related to ad-hoc topics. An additional set of questions on ad-hoc topics will be also proposed by organizers. Once released this 2010 set of questions, participants will have a short time period to submit the results by their systems. This period will be set in accordance with participants. Runs from baseline systems will be also added. Informative content and linguistic quality of answers will be evaluated by participants and organizers based on a short questionnaire.

5 Conclusion

QA@INEX is offering an evaluation framework combining QA, passage retrieval and automatic summarizing by passage extraction. Its main features are the use of the Wikipedia as referential, its proximity with INEX ad-hoc task and the introduction of new evaluation metrics.

References

1. Lee, Y.H., Lee, C.W., Sung, C.L., Tzou, M.T., Wang, C.C., Liu, S.H., Shih, C.W., Yang, P.Y., Hsu, W.L.: Complex Question Answering with ASQA at NTCIR 7 ACLIA. In: Proceeding of the 7th NTCIR Workshop Meeting, Tokyo, Japan (dec 2008) 70–76
2. Verberne, S., Boves, L., Oostdijk, N., Coppen, P.A.: Discourse-based answering of why-questions. *Traitement Automatique des Langues, Discours et document: traitements automatiques* **47**(2) (2007) 21–41
3. Verberne, S., Raaijmakers, S., Theijssen, D., Boves, L.: Learning to Rank Answers to Why-Questions. In: Proceedings of 9th Dutch-Belgian Information Retrieval Workshop (DIR 2009). (2009) 34–41
4. Yu, H., Hatzivassiloglou, V.: Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. In: Proceedings of 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP). (2003) 129–136
5. Louis, A., Nenkova, A.: Performance confidence estimation for automatic summarization. In: EACL, The Association for Computer Linguistics (2009) 541–548

Report on the XML Mining Classification Track at INEX 2009

Categorization of XML Documents in a graph of documents

Ludovic Denoyer and Patrick Gallinari

LIP6 - University of Paris 6

Abstract. We describe here the XML Mining Track at INEX 2009. This track 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 year, the track focuses on the supervised classification of XML documents using links between documents. We consider a corpus of about 55,000 wikipedia pages with the associated hyperlinks. The participants have developed models using the content information, the internal structure information of the XML documents and also the link information between documents.

1 Introduction

The XML Document Mining track¹ 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 five editions during INEX 2005, 2006, 2007, 2008 and 2009. The four first editions have been summarized in [1], [2] and [3] and we focus here on the 2009 edition.

Among the many open problems for handling structured data, the track focuses on the semi-supervised classification task where documents are organized in graph². The goal of the track was therefore to explore algorithmic, theoretical and practical issues regarding the classification of interdependent XML documents. We describe here the characteristics of the corpus provided to the participants, and briefly present the results obtained by the different teams. The methods used are described in the participants papers and will be summarized in the final paper.

¹ <http://xmlmining.lip6.fr>

² Note that one other INEX 2009 track focuses on the unsupervised clustering problem.

2 Categorization/Clustering of a graph of XML Documents organized

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). Moreover, in a large majority of cases (Web collections for example), XML documents collections are also structured by links between documents (hyperlinks for example). These links can be of different types and correspond to different informations: for example, one collection can provide hierarchical links, hyperlinks, citations, Most models developed in the field of XML categorization simultaneously use the content information and the internal structure of XML documents (see [1] and [2] for a list of models) but they rarely use the external structure of the collection i.e the links between documents. Some methods using both content and links have been proposed in [3].

The XML Classification Track focuses on the problem of learning to classify documents organized in a graph of documents. Unlike the 2008 track, we consider here the problem of *Multiple labels classification* where a document belongs to one or many different categories. This task considers a transductive context where, during the training phase, the whole graph of documents is known but the labels of only a part of them are given to the participants (Figure 1).

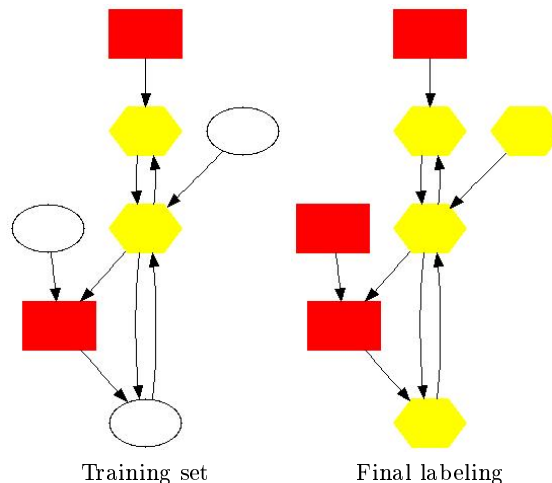


Fig. 1. The supervised classification task. Colors/Shapes correspond to categories, circle/white nodes are unlabeled nodes. Note that in this track, documents may belong to many categories.

3 Corpus

The corpus provided is a subset of the *INEX 2009 Corpus* [?]. We have extracted a set of 54,889 documents and the links between these documents. These links corresponds to the links provided by the authors of the Wikipedia articles. The documents have been transformed into TF-IDF vectors by the organizers. The corpus thus corresponds to a set of 54,889 vectors of dimension 186,723. The documents belong to 39 categories that correspond to 39 Wikipedia portals. We have provided the labels of 20 % of the documents. The corpus is composed of 4,554,203 directed links that correspond to hyperlinks between the documents of the corpus. Each document is concerned by 84.1 links on average.

Number of documents	54,889
Number of training documents	11,028
Number of test documents	43,861
Number of categories	39
Number of links	4,554,203
Number of distinct words	186,723

3.1 Evaluation Measures

In order to evaluate the submissions of the participants, we have used different measures:

- The first set of measures are computed over each category and then averaged over the categories (using a micro or a macro average):
 - *Accuracy* (ACC) corresponds to the classification error. Note that a system that returns zero relevant category for each document has a quite good accuracy.
 - *F1 score* (F1) corresponds to the classical F1 measure and measures the ability of a system to find the relevant categories.
- The second set of measures are computed over each document and then averaged over the documents.
 - *Average precision* (APR) corresponds to the Average Precision computed over the list of categories returned for each document. It measures the ability of a system to rank correctly the relevant categories. This measure is based on a ranking score of each category for each document.

4 Participants and Submissions

Five different teams have participated to the track. They have submitted different runs and we present here only the best results obtained by each team.

Comments on the methods and the results will be added in the final paper, after the INEX workshop.

Team	Micro ACC	Macro ACC	Micro F1	Macro F1	APR
University of Wollongong	92.5	94.6	51.2	47.9	68
University of Peking	94.7	96.2	51.8	48	70.2
XEROX Research Center	96.3	97.4	60	57.1	67.8
University of Saint Etienne	96.2	97.4	56.4	53	68.5
University of Granada	67.8	75.4	26.2	25.3	72.9

5 Conclusion

We have presented the XML Mining Classification Track at INEX 2009. This track concerns the multiple labels classification task in a graph of documents. The results submitted by the participants show the effectiveness of the methods proposed. These methods will be described in the final paper.

Acknowledgments

We would like to thank all the participants for their efforts and hard work.

References

1. Denoyer, L., Gallinari, P.: Report on the xml mining track at inex 2005 and inex 2006: categorization and clustering of xml documents. *41(1)* (2007) 79–90
2. Denoyer, L., Gallinari, P.: Report on the xml mining track at inex 2007 categorization and clustering of xml documents. *42(1)* (2008) 22–28
3. Denoyer, L., Gallinari, P.: Overview of the inex 2008 xml mining track. In: INEX. (2008) 401–411

Report on the XML Mining Track's Clustering Task at INEX 2009

Richi Nayak Chris De Vries Sangeetha Kutty Shlomo Geva

Faculty of Science and Technology
Queensland University of Technology
GPO Box 2434, Brisbane Qld 4001, Australia
{r.nayak, c.devries, s.kutty, s.geva}@qut.edu.au

Abstract. This report explains the objectives, datasets, tasks and evaluation criteria of the clustering task held in INEX 2009. The report also describes the approaches and results obtained by the different participants.

Keywords: Clustering, XML document mining, INEX, Wikipedia, Structure and content.

1 Introduction

In the last decade, we have observed a proliferation of approaches for clustering XML documents based on their structure and content. There have been many approaches developed for diverse application domains. Many applications require data objects to be grouped by similarity of content, tags, paths, structure and semantics. The clustering task in INEX 2009 evaluates clustering approaches in the context of XML information retrieval.

INEX 2009 included two tasks in the XML Mining track namely classification and clustering. The task set in the clustering task requires the participants to group the documents into clusters without any knowledge of cluster labels using an unsupervised learning algorithm. On the other hand, the classification task requires the participants to label the documents in the dataset into known classes using a supervised learning algorithm and a training set. This report gives the details of clustering task.

The INEX 2009 clustering task is different from the previous years due to its incorporation of a different evaluation strategy. The clustering task explicitly tests the Jardine and van Rijsbergen cluster hypothesis (1971), which states that documents that cluster together have a similar relevance to a given query. It uses manual query assessments from the INEX Ad Hoc track. If the cluster hypothesis holds true, and if suitable clustering can be achieved, then a clustering solution will minimise the

number of clusters that need to be searched to satisfy any given query. There are important practical reasons for performing collection selection on a very large corpus. If only a small fraction of clusters (hence documents) need to be searched, then the throughput of an information retrieval system will be greatly improved.

INEX 2009 clustering task provides an evaluation forum to measure the performance of clustering methods for collection selection on a huge scale test collection (consisting of a set of documents, their labels, a set of information needs (queries), and the answers to those information needs).

2 Corpus

The INEX XML Wikipedia collection is used as a dataset in this task. This English Wikipedia collection, marked-up version of the Wikipedia documents, 60 Gigabytes in size contains around 2.7 million documents in XML format. The mark-up includes, for instance, explicit tagging of named entities. In order to enable participation with minimal overheads in data-preparation the collection was pre-processed to provide various representations of the documents. For instance, a bag-of-words representation of terms and frequent phrases in a document, frequencies of various XML structures in the form of trees, links, named entities, etc. These various collection representations made this task a lightweight task that required the participants to submit the clustering solutions only without worrying about pro-processing this huge data collection.

Some statistics about the dataset are as follows. There are a total of 1,970,515 terms after stemming, stopping, and eliminating terms that occur in a single document for this collection of 2,666,190 documents. There are 1,900,075 unique terms that appear more than once enclosed in entity tags. There are 5213 unique entity tags in the collection. There are a total of 110,766,016 links in the collection. Each link is unique.

A subset of collection containing about 50,000 documents (of the entire INEX 2009 corpus) was also used in the task to evaluate the categories labels results only, for teams that were unable to process such a large data collection. There are a total of 348,552 categories for all 2.7 million documents. These categories are derived by using the YAGO ontology. The YAGO categories appear to follow a power law distribution.

3 Tasks and Evaluation Measures

The task in this track was to utilize unsupervised classification techniques to group the documents into clusters. Participants were asked to submit multiple clustering solutions containing different numbers of clusters such as 100, 500, 1000, 2500, 5000 and 10000.

The clustering solutions are evaluated by two means. Firstly, we utilise the *classes-to-clusters evaluation* which assumes that the classification of the documents in a sample is known (i.e., each document has a class label). Then any clustering of these documents can be evaluated with respect to this predefined classification. It is important to note that the class labels are not used in the process of clustering, but only for the purpose of evaluation of the clustering results.

The standard criterion of purity is used to determine the quality of clusters. These evaluation results were provided online and ongoing, starting from mid-October. Entropy and F-Score were not used in evaluation. The reason behind was that a document in the corpus maps to more than one category. Due to multi labels that a document can have, it was possible to obtain higher value of Entropy and F-Score than the ideal solution.

Purity measures the extent to which each cluster contains documents primarily from one class. Each cluster is assigned with the class label of the majority of documents in it. Then the *error* is computed as the proportion of documents with different class and cluster labels. Inversely, the *accuracy* is the proportion of documents with the same class and cluster label. Purity is measured as the ratio of Number of documents with the majority label in cluster to Number of documents in cluster. The macro and micro purity of entire clustering solution is obtained as a weighted sum of the individual cluster purity. In general, larger the value of purity, better the clustering solution is.

$$\text{Purity (k)} = \frac{\text{Number of documents with the majority label in cluster k}}{\text{Number of document in cluster k}}$$

$$\text{Micro-Purity} = \frac{\sum_{k=0}^n \text{Purity(k)} * \text{TotalFoundByClass(k)}}{\sum_{k=0}^n \text{TotalFoundByClass(k)}}$$

$$\text{Macro-Purity} = \frac{\sum_{k=0}^n \text{Purity(k)}}{\text{Total Number of Categories}}$$

The clustering solutions are also evaluated to determine the quality of cluster relative to the optimal collection selection goal, given a set of queries. Better clustering solutions in this context will tend to (on average) group together relevant results for (previously unseen) ad-hoc queries. Real Ad-hoc retrieval queries and their manual

assessment results are utilised in this evaluation. This novel approach evaluates the clustering solutions relative to a very specific objective - clustering a large document collection in an optimal manner in order to satisfy queries while minimising the search space. The Normalised Cumulative Gain is used to calculate the score of the best possible collection selection according to a given clustering solution. Better the score when the query result set contains more cohesive clusters.

The normalized cumulative gain (nCG) for a given topic t is given by,

$$\text{nCG}(t) = \frac{\text{Cumulative gain of the relevant articles in the set of clusters for topic } t}{\text{Ideal Cumulative gain for the topic } t}$$

For a given topic, the relevant documents present in each of the clusters are represented in a vector sorted in descending order. Then the cumulated gain for the vector is calculated which is then normalized on the ideal gain vector.

The mean and the standard deviation of the nCG score over all the topics are then calculated.

$$\text{Mean nCG} = \frac{\sum_{t=0}^n \text{nCG}(t)}{\text{Total Number of topics } (n)}$$

A total of 69 topics were used to evaluate the quality of clusters generated on the full set of collection of about 2.7 million documents. A total of 52 topics were used to evaluate the quality of clusters generated on the subset of collection of about 50,000 documents. A total number of 4859 documents were found relevant by the manual assessors for the 69 topics. An average number of 71 documents were found relevant for a given topic by manual assessors. The nCG value varies from 0 to 1 and for a given topic if the cluster returned contained only one relevant document then it was removed as the nCG value will be infinity.

4 Participants and Submissions

A total of six research teams have participated in the INEX 2009 clustering task. Two of them submitted the results for the subset data only. We briefly summarised the approaches employed by the participants.

4.1 Bilkent Web Databases Research Group (BilWeb)

The research team used Cover-Coefficient Based Clustering Methodology (C3M) to cluster the XML documents. C3M is a single-pass partitioning type clustering algorithm which measures the probability of selecting a document given a term that has been selected from another document. They cluster the XML documents into a

given number of clusters (for several values like 1000, 10000, etc.) using C3M method. As another approach, they adapted term-centric and document-centric index pruning techniques from the literature to obtain more compact representations of the documents. They cluster documents with these reduced representations for various pruning levels, again using C3M algorithm. Their findings show that the clustering quality, in terms of a number of INEX evaluation measures, do not vary much even when a considerable amount of the document content (i.e., around 30%) is pruned. All of the experiments are executed on the subset of INEX 2009 corpus including 50K documents.

4.2 De Vries and Geva

The Random Indexing (RI) K-tree has been used to cluster the entire 2,666,190 XML documents in the INEX 2009 Wikipedia collection. Clusters were created as close as possible to the 100, 500, 1000, 5000 and 10000 clusters required for evaluation. The RI K-tree produces clusters in an unsupervised manner where the exact number of clusters can not be precisely controlled. It is determined by the tree order and the randomized seeding process of k-means used to perform splits in the tree. The algorithm produces clusters of many sizes in a single pass. The desired clustering granularity is selected by choosing a particular level in the tree. Random Indexing (RI) is an efficient dimensionality reduction technique that projects points in a high dimensional space onto a randomly selected lower dimensional space. It is able to preserve the topology of the points. In the context of document representation, topology preserving dimensionality reduction is preserving document meaning – or at least this is the conjecture which we test here. The RI projection produces dense document vectors that work well with the K-tree algorithm. Document structure has been represented by using a bag of words and a bag of tags representation derived from the semantic markup in the INEX 2009 collection. The bag of words is made up of term frequencies contained *within any entity tags* in the collection. The term frequencies were weighted with BM25 where $K1 = 2$ and $b = 0.75$. The bag of tags representation is made up in an analogous manner of XML entity tag frequencies. The tag frequencies were not weighted.

4.3 Pinto et al (BUAP, MEXICO)

The employed approach was quite simple and focused on high scalability. The team used a modified version of the Star clustering method [1] which automatically obtains the number of clusters. This method has been used in a number of papers dealing with clustering of documents[2-4]. In each iteration, this clustering method brings together all those items whose similarity value is higher than a given threshold T , which is typically assumed to be the similarity average of the whole document collection and, therefore, the clustering method "discover" the number of clusters by its own. The run we submitted to the INEX clustering task split the complete document collection into small subsets which are clustered with the above mentioned clustering method. Each cluster C_i obtained is then represented by a vector V_i calculated on the basis of the cluster members. In an iterative process, we cluster the V_i representative vectors until the similarity threshold is too low to bring together items in the clusters.

4.4 Kutty and Nayak

A Multi-Feature model (MFM) built using the common structural features and the corresponding content features of the XML documents has been used to cluster the subset XML Wikipedia collection. Each XML document in the INEX Wikipedia corpus is parsed and modeled as a rooted labeled ordered *document tree*. Each document tree contains nodes which represent the tag names. A frequent subtree mining algorithm is then applied to extract the common structural features from these document trees in the corpus. The common subtrees were used to extract the corresponding content and then represented in MFM. Due to the large size of the dataset and the presence of a number of structural features, MFM built is also very large and hence random projection techniques were applied to project the points to lower dimensional space. Decomposition techniques such as Higher-order Singular Value Decomposition, Tucker decompositions are then applied on the reduced sized MFM. The decomposition factors are then used to derive clustering solution. The term frequencies in the MFM model were weighted with TF-IDF. There were 100 and 500 clusters created for evaluation.

5 Conclusion

The clustering task in INEX 2009 brought together researchers from Information Retrieval, Data Mining, Machine Learning and XML fields. It allowed participants to evaluate clustering methods against a real use case and with significant volumes of data. The task was designed to facilitate participation with minimal effort by providing not only raw data, but also pre-processed data which can be easily used by existing clustering software.

Exploiting Index Pruning Methods for Clustering XML Collections

Ismail Sengor Altingovde, Duygu Atilgan and Özgür Ulusoy

Department of Computer Engineering, Bilkent University, Ankara, Turkey
{ismaila, atilgan, oulusoy}@cs.bilkent.edu.tr

Abstract. In this paper, we first employ Cover-Coefficient Based Clustering Methodology (C³M) for clustering XML documents. Next, we apply index pruning techniques from the literature to reduce the size of the document vectors. Our experiments show that for certain cases, it is possible to prune up to 70% of the collection (or, more specifically, underlying document vectors) and still generate a clustering structure that yields the same quality with that of the original collection, in terms of a set of evaluation metrics.

1 Introduction

As the number and size of XML collections increase rapidly, there occurs the need to manage these collections efficiently and effectively. While there is still an ongoing research in this area, INEX XML Mining Track fulfills the need for an evaluation platform to compare the performance of several clustering methods on the same set of data. Within the Clustering task of XML Mining Track of INEX campaign, clustering methods are evaluated according to cluster quality measures on a real-world Wikipedia collection.

To this end, in the last few workshops, many different approaches are proposed which use structural, content-based and link-based features of XML documents. In INEX 2008, Kutty et al. [9] use both structure and content to cluster XML documents. They reduce the dimensionality of the content features by using only the content in frequent subtrees of an XML document. In another work, Zhang et al. [11] make use of the hyperlink structure between XML documents through an extension of a machine learning method based on the Self Organizing Maps for graphs. De Vries et al. [8] use K-Trees to cluster XML documents so that they can obtain clusters in good quality with a low complexity method. Lastly, Tran et al. [10] construct a latent semantic kernel to measure the similarity between content of the XML documents. However, before constructing the kernel, they apply a dimension reduction method based on the common structural information of XML documents to make the construction process less expensive. In all of these work mentioned above, not only the quality of the clusters but the efficiency of the clustering process is also taken into account.

In this paper, we propose an approach which reduces the dimension of the underlying document vectors without change or with a slight change in the quality of the output clustering structure. More specifically, we use a partitioning type clustering

algorithm, so-called Cover-Coefficient Based Clustering Methodology (C³M) [6], along with some index pruning techniques for clustering XML documents.

2 Approach

2.1 Baseline Clustering with C³M Algorithm

In this work, we use the well-known Cover-Coefficient Based Clustering Methodology (C³M) [6] to cluster the XML documents. C³M is a single-pass partitioning type clustering algorithm which is shown to have good information retrieval performance with flat documents (e.g., see [5]). The algorithm operates on documents represented by vector space model. Using this model, a document collection can be abstracted by a document-term matrix, D ; of size m by n whose individual entries, d_{ij} ($1 < i < m$; $1 < j < n$), indicate the number of occurrences of term j (t_j) in document i (d_i). In C³M, the document-term matrix¹ D is mapped into an m by m cover-coefficient (C) matrix which captures the relationships among the documents of a database. The diagonal entries of C are used to find the number of clusters, denoted as n_c ; and the selection of cluster seeds. During the construction of clusters, the relationships between a nonseed document (d_i) and a seed document (d_j) is determined by calculating the c_{ij} entry of C ; where c_{ij} indicates the extent to which d_i is covered by d_j .

A major strength of C³M is that for a given dataset, the algorithm itself can determine the number of clusters, i.e., there is no need for specifying the number of clusters, as in some other algorithms. However, for the purposes of this track, we cluster the XML documents into a given number of clusters (for several values like 1000, 10000, etc.) using C³M method, as required. In this paper, we simply use the content of XML documents for clustering. Our preliminary experiments that also take the link structure into account did not yield better results than just using the content. Nevertheless; our work in this direction is still under progress.

2.2 Employing Pruning Strategies for Clustering

From the previous works, it is known that static index pruning techniques can reduce the size of an index (and the underlying collection) while providing comparative effectiveness performance with that of the unpruned case [2, 3, 4, 7]. In a more recent study, we show that such pruning techniques can also be adapted for pruning the element-index for an XML collection [1]. Here, with the aim of both improving the quality of clusters and reducing the dataset dimensions for clustering, we apply static pruning techniques on XML documents. We adapt two well-known pruning techniques, namely, term-centric [7] and document-centric pruning [4] from the literature to obtain more compact representations of the documents. Then, we cluster documents with these reduced representations for various pruning levels, again using

¹ Note that, in practice, the document-term matrix only includes non-zero term occurrences for each document.

C³M algorithm. The pruning strategies we employ in this work can be summarized as follows:

- *Document-centric pruning*: This technique is essentially intended to reduce the size of an inverted index by discarding unimportant terms from each document. In the original study, a term's importance for a document is determined by that term's contribution to the document's Kullback-Leibler divergence (KLD) from the entire collection [4]. In a more recent work [2], we show that using the contribution of a term to the retrieval score of a document (by using a function like BM25) also performs quite well. In this paper, we again follow this practice and for each term that appears in a given document, we compute that term's BM25 score for this document. Then, those terms that have the lowest scores are pruned, according to the required pruning level. Once the pruned documents are obtained at a given pruning level, corresponding document vectors are generated to be fed to the C³M clustering algorithm.
- *Term-centric pruning*: This method operates on an inverted index, so we start with creating an index for our collection. Next, we apply term-centric pruning at different pruning levels, and once the pruned index files are obtained, we convert them to the document vectors to be given to the clustering algorithm². In a nutshell, the term-centric pruning strategy works as follows [7]. For each term t , the postings in t 's posting list are sorted according to their score with respect to a ranking function, which is BM25 in our case. Next, the k^{th} highest score in the list, z_t , is determined and all postings that have scores less than $z_t * \epsilon$ are removed, where ϵ specifies the pruning level. In this paper, we skip this last step, i.e. ϵ -based tuning, and simply remove the lowest scoring postings of a list for a given pruning percentage.

3 Experiments

In this paper, we use a subset of the INEX 2009 XML Wikipedia collection provided by XML Mining Track. This subset contains 54575 documents and takes 270 MB.

As the baseline, we form clusters by applying C³M algorithm to XML documents represented with the bag of words representation of terms, as provided by the track organizers. For several different number of output clusters, namely 100, 500, 1000, 2500, 5000 and 10000, we obtain the clusters and evaluate them at the online evaluation website of this track. The website reports micro purity, macro purity, micro entropy, macro entropy, normalized mutual information (NMI), micro F1 score and macro F1 score for a given clustering structure. The results are provided in Table 1. Note that, for almost all evaluation measures, higher number of clusters (i.e., 5000 or

² It is possible to avoid converting the index to the document vectors by slightly modifying the input requirements of the clustering algorithm. Anyway, we did not spend much effort in this direction as this conversion stage, which is nothing but an inversion of the inverted index, can also be realized in an efficient manner.

Table 1. Effectiveness figures of the baseline C³M clustering for different number of clusters. Best results for each measure are shown in bold.

No. of clusters	Micro Entropy	Macro Entropy	Micro F1	Macro F1	Micro Purity	Macro Purity	NMI
100	9.8317	8.5835	0.0027	0.0002	0.1152	0.1343	0.2912
500	8.1088	7.0836	0.0071	0.0016	0.1528	0.1777	0.4179
1000	7.2514	6.2830	0.0111	0.0041	0.1861	0.2147	0.4735
2500	6.1242	5.1329	0.0224	0.0132	0.2487	0.3031	0.5393
5000	5.2457	4.2256	0.0387	0.0298	0.3265	0.4160	0.5858
10000	4.9959	4.0081	0.0332	0.0162	0.4004	0.5416	0.6829

Table 2. Effectiveness comparison of clustering structures based on TCP and DCP at various pruning levels. Number of clusters is 10000. Prune (%) field denotes the percentage of pruning. Best results for each measure are shown in bold.

Pruning Strategy	Prune (%)	Micro Entropy	Macro Entropy	Micro F1	Macro F1	Micro Purity	Macro Purity	NMI
No Prune	0%	4.9959	4.0082	0.0332	0.0162	0.4004	0.5416	0.6829
DCP	30%	4.9874	4.0222	0.0334	0.0160	0.4028	0.5400	0.6835
TCP	30%	4.9740	4.0611	0.0324	0.0156	0.3914	0.5229	0.6835
DCP	50%	4.9780	4.0465	0.0331	0.0156	0.4019	0.5375	0.6840
TCP	50%	4.9674	4.0946	0.0319	0.0151	0.3870	0.5141	0.6837
DCP	70%	4.9464	4.0937	0.0329	0.0150	0.4016	0.5302	0.6853
TCP	70%	4.9780	4.1345	0.0311	0.0142	0.3776	0.5042	0.6834
DCP	90%	4.8723	4.2689	0.0308	0.0125	0.3783	0.4768	0.6884
TCP	90%	5.0387	4.1604	0.0302	0.0136	0.3639	0.5073	0.6817

10000) yields better performance. We suspect that this might be caused by the fact that the collection we have used in our experiments includes too many categories. Thus, as a future work, we plan to obtain results with the larger collection and some alternative evaluation measures.

Next, we experiment with the clusters produced by the pruning-based approaches. For each pruning technique, namely, TCP and DCP, we obtain the document vectors at four different pruning levels, i.e., 30%, 50%, 70% and 90%. Note that, a document vector includes term id and number of occurrences for each term in a document, stored in the binary format (i.e., as a transpose of an inverted index). In Table 2, we provide results for 10000 clusters, for which case C³M baseline yielded the best performance, as discussed above. Our findings reveal that up to 70% pruning with DCP, quality of the clusters is still comparable to or even superior than the corresponding baseline case, in terms of the INEX evaluation measures.

Regarding the comparison of pruning strategies, clusters obtained with DCP yield better results than those obtained with TCP up to 70% pruning for all of the measures except micro entropy. For the pruning levels higher than 70%, DCP and TCP give better results interchangeably for different measures. In [1], we observed a similar behavior regarding the retrieval effectiveness of indexes pruned with TCP and DCP.

From Table 2 we also deduce that DCP-based clustering at 30% pruning level produce the best results for most of the evaluation measures in comparison to the

Table 3. Effectiveness figures of DCP at 30% pruning for different number of clusters.

No. of clusters	Micro Entropy	Macro Entropy	Micro F1	Macro F1	Micro Purity	Macro Purity	NMI
100	10.9671	9.5800	0.0013	0.0001	0.1021	0.1265	0.3428
500	8.9949	7.9655	0.0037	0.0002	0.1347	0.1539	0.4759
1000	8.0647	7.1004	0.0056	0.0005	0.1641	0.1917	0.5308
2500	6.8442	5.9143	0.0110	0.0021	0.2234	0.2737	0.5970
5000	5.9200	4.9626	0.0193	0.0060	0.2986	0.3854	0.6423
10000	4.9874	4.0221	0.0334	0.0160	0.4028	0.5400	0.6835

other pruning-based clusters. For this best-performing case, namely DCP at 30% pruning, we also provide performance findings with varying number of clusters (see Table 3). The results show that, as in the baseline case, the cluster quality improves with increasing number of clusters. Furthermore, the DCP-based clusters are inferior to the corresponding baseline clustering up to 10000 clusters, but they provide almost the same performance for the 10000 clusters case.

4 Conclusion

In this paper, we employ the well-known C³M algorithm for clustering XML documents. Furthermore, we use index pruning techniques from the literature to reduce the size of the document vectors on which C³M operates. Our findings reveal that, for a high number of clusters, the quality of the clusters produced by the C³M algorithm does not degrade when up to 70% of the index (and, equivalently, the document vectors) is pruned.

Our future work involves repeating our experiments with larger datasets and additional evaluation metrics. Furthermore, we plan to extend the pruning strategies to exploit the structure of the XML documents in addition to content.

Acknowledgments. This work is supported by TÜBİTAK under the grant number 108E008.

References

1. Altıngövdde, I. S., Atilgan, D., Ulusoy, Ö.: XML Retrieval using Pruned Element-Index Files. In: Proc. of ECIR'10, to appear.
2. Altıngövdde, I. S., Özcan, R., Ulusoy, Ö.: A practitioner's guide for static index pruning. In: Proc. of ECIR'09. (2009) 675-679
3. Blanco, R., Barreiro, A.: Boosting static pruning of inverted files. In: Proc. of SIGIR'07, The Netherlands. (2007) 777-778
4. Büttcher, S., Clarke, C. L.: A document-centric approach to static index pruning in text retrieval systems. In: Proc. of CIKM'06. (2006) 182-189
5. Can, F., Altıngövdde, I. S., Demir, E.: Efficiency and effectiveness of query processing in cluster-based retrieval. In: Inf. Syst. 29(8) (2004) 697-717
6. Can, F., Ozkarahan E. A.: Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. In: ACM Trans. Database Systems 15 (1990) 483-517.

7. Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y. S., Soffer, A.: Static index pruning for information retrieval systems. In: Proc of SIGIR'01. (2001) 43-50
8. De Vries, C.M., Geva, S.: Document Clustering with K-tree In: Proc. of INEX 2008 Workshop. (2009) 420-431
9. Kutty, S., Tran, T., Nayak, R., Li, Y.: Clustering XML documents using frequent subtrees In: Proc. of INEX 2008 Workshop. (2009) 436-445
10. Tran, T., Kutty, S., Nayak, R.: Utilizing the Structure and Content Information for XML Document Clustering In: Proc. of INEX 2008 Workshop. (2009) 460-468
11. Zhang, S., Hagenbuchner, M., Tsoi, A.C., Sperduti, A.: Self Organizing Maps for the Clustering of Large Sets of Labeled Graphs In: Proc. of INEX 2008 Workshop. (2009) 469-481

Multi-label Wikipedia classification with textual and graph features

Boris Chidlovskii

Xerox Research Centre Europe
6, chemin de Maupertuis, F-38240 Meylan, France

Abstract. We address the problem of categorizing a large set of linked documents with important content and structure aspects, in particular, from the Wikipedia collection proposed at the INEX 2009 XML Mining challenge. We analyse the network of collection pages and turn it into valuable features for the classification. We combine the content-based and link-based features of pages to trains to build a transductive categorizer for the unlabeled pages. In the multi-label setting, we review a number of existing techniques and propose a new one. We report evaluation results obtained with a variety of learning methods on the training set of the Wikipedia corpus.

1 Introduction

The objective of the INEX 2009 XML Mining challenge is to develop machine learning methods for structured data mining and to evaluate these methods for XML document mining tasks. The challenge proposes several datasets coming from different XML collections and covering a variety of classification and clustering tasks.

In this work, we address the problem of categorizing a very large set of linked XML documents with important content and structural aspects, for example, from Wikipedia online encyclopedia. We cope with the case where there is a small number of labeled pages and a much larger number of unlabeled ones. For example, when categorizing Web pages, some pages have been labeled manually and a huge amount of unlabeled pages is easily retrieved by crawling the Web. The semi-supervised approach to learning is motivated by the high cost of labeling data and the low cost for collecting unlabeled data. Withing XML Mining challenge 2009, the Wikipedia categorization challenge has been indeed set in the semi-supervised mode, where only 20% of page labels are available at the training step.

Wikipedia is a free multilingual encyclopedia project supported by the non-profit Wikipedia foundation ¹. In April 2008, Wikipedia accounted for 10 million articles which have been written collaboratively by volunteers around the world, and almost all of its articles can be edited by anyone who can access the Wikipedia website. Launched in 2001, it is currently the largest and most popular general reference work on the Internet. Automated analysis, mining and categorization of Wikipedia pages can serve to

¹ <http://www.wikipedia.org>.

improve its internal structure as well as to enable its integration as an external resource in different applications.

Any Wikipedia page is created, revised and maintained according to certain policies and guidelines [11]. Its edition follows certain rules for organizing the content and structuring it in the form of sections, abstract, table of content, citations, links to relevant pages, etc.. In the following, we distinguish between different aspects of a Wikipedia page. First, its content is given by the set of words occurred in the page. Second, we are interested the set of links in the page referring to other pages. In some cases, we might be interested in the page XML/HTML structure, given by the set of tags, attributes and their values in the page [4]. These elements control the presentation of the page content to the viewer. Finally, we are very interested in extracting the page metadata given in the page Infobox. Unfortunately, only a small part of the Wikipedia pages include infoboxes [16].

2 INEX 2009 collection

The training corpus for the INEX 2009 XML Mining track is composed of three following files² :

- A category file that gives the set of documents considered in this track and the categories of the training documents.
- A link file that gives the links between the documents.
- A content file that corresponds to normalized tf-idf vectors computed by the organizers over this collection.

We note a small mismatch between the category, tf-idf and link data. The category data includes 54,889 pages, while the tf-idf values and links are available for 54,572 and 54,451 of them, respectively. On the other hand, the graph of 4,554,203 links corresponds to one very large connected component in the Wikipedia corpus.

Similarly, the training set with the category annotations is composed of 11,028 elements, tf-idf vectors are given for 10,968 of them and links are provided for 10,992.

Figure 1).a plots the link structure of the graph³. Two distributions of outgoing and incoming links in pages in the corpus are shown in Figure 1).b; both distributions are close to fitting the power distribution law.

Unlike the previous editions, INEX 2009 XML Mining classification challenge operates a multi-label setting. The multi-category annotations are available for 20% of data (11,028 pages). The annotations correspond to one of the 39 Wikipedia portals. Frequencies of categories occurring in the training set are reported in Figure 2).a. one leading category (`Portal:Trains`) can be easily recognized as well as 7-8 core categories. Other categories are less frequent, they form a long tail of the plot. Figure 3) represents the matrix of co-occurrences for all categories in the training set.

Due to the multi-label setting, each page may be annotated with one or more categories. Figure 2).b presents the length distribution in the category sets. On average, one page has 1.46 labels.

² The files can be downloaded from http://www-connex.lip6.fr/denoyer/inex2009/corpus_train.tar.gz.

³ The LGL package has been used for the component plotting.

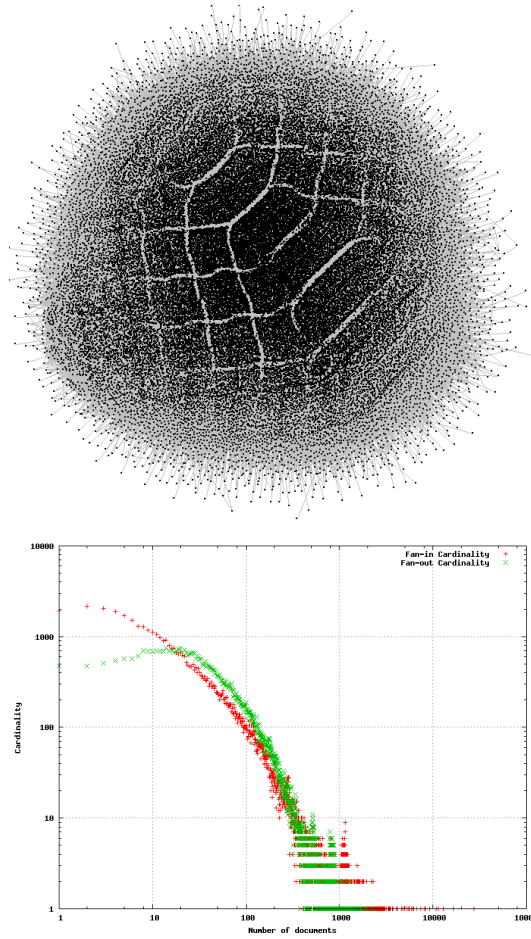


Fig. 1. INEX 2009 collection analysis. a) The link structure as one connected component. b) Fan-in and fan-out link distribution.

3 Page representation

The most obvious level of representation is the textual content. We include the contents of body and subject fields in a *bag-of-words* representation, that has proven to be well suited for classification. The idea is to construct a vector where each feature represents a word in the lexicon and the feature values express some weight.

3.1 Text representation

The tf-idf data is the textual representation of the corpora documents. The document frequency df of a word w is the number of documents it occurs in, the term frequency tf is the number of word occurrences in a document d , N is the number of documents.

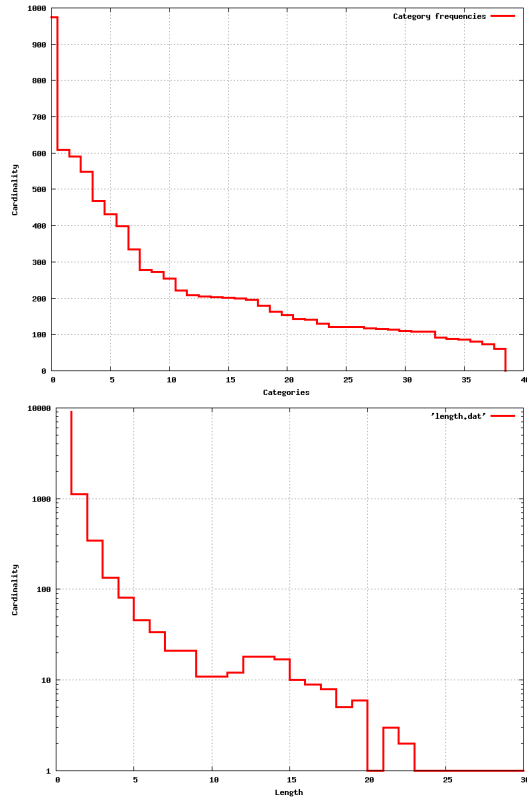


Fig. 2. INEX 2009 collection analysis. a) Category frequencies. b) Length frequencies.

For the experiments, we consider three different text representations: *binary*, *frequency* and *tf-idf* values, defined as follows:

$$\text{binary}(w,d) = \begin{cases} 1 & \text{if } w \text{ occurs in } d \\ 0 & \text{otherwise} \end{cases}$$

$$\text{frequency}(w,d) = tf$$

$$\text{tf-idf}(w,d) = tf \cdot \log(N/df).$$

The feature set based on a bag-of-words representation are high-dimensional (around 180,000) and the feature vectors are very sparse. This makes it particularly suited for the SVM classification with a linear kernel [8], because only a small number of the words actually occur in each respective document.

3.2 Graph features

Graph structures induced by links or communications between some instances have received a lot of attention in scientific literature and have proven to be useful in diverse

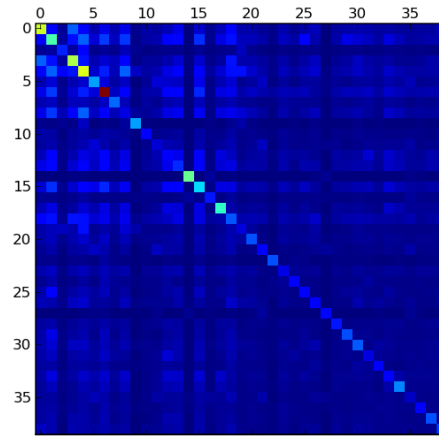


Fig. 3. INEX 2009 collection analysis. Category occurrences.

fields like sociology, biology, engineering, computer science and epidemiology [10]. With the advent of techniques to handle large graphs and the emergence of huge, real-life linked structures on the internet, structure in link-based data has become a subject of intensive research.

The structure of a large linked collection, like Wikipedia is not homogeneous. It has been shown that certain properties of nodes in a graph can serve very well to automatically detect their particular role [12]. We adopt and report below several groups of commonly used features to represent key properties of nodes in a network [2, 12].

We represent the Wikipedia linked structure as a *directed graph* $G = \langle V, E \rangle$, where V contains N nodes and E includes M edges. The first feature represents the immediate characteristics of a node in the network:

1. the number of incoming and outgoing links and their sum, $m(v) = in(v) + out(v)$.

In hypertext classification two features, that represent the authority that is assigned to nodes by its peers, have proven to be very valuable. Nodes with a high number of incoming edges from hubs are considered to be *authorities*, nodes linking to a large number of authorities are *hubs* [9].

2. *hub score* $h(v)$ is given by v -th element of the principal eigenvector of AA^T where A is the adjacency matrix corresponding to graph G ;
3. *authority score* $a(h)$ is given by v -th element of the principal eigenvector of $A^T A$.

The next group is a set of different centrality measures to model the position of the page in the network. These depend on a *undirected* version $G' = \langle V, E' \rangle$ of graph

G . We calculate the shortest paths in G' using a Matlab library for working with large graphs [7]. We obtain the distance d_{st} from node s to t and the number σ_{st} of paths from s to t . The number of paths from s to t via v is denoted as $\sigma_{st}(v)$:

4. *mean centrality*: $C_M(v) = \frac{n}{\sum_{s \in V} d_{vs}}$;
5. *degree centrality*: $\deg(v) = |s| : (v, s) \in E$;
6. *betweenness centrality*: $C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$;
7. *closeness centrality*: $C_C(v) = \frac{1}{\max_t d(v, t)}$;
8. *stress centrality*: $C_S(v) = \sum_{s \neq v \neq t} \sigma_{st}(v)$.

One more feature characterizes the connectedness in the direct neighborhood of node v :

9. *clustering coefficient*: $CC(v) = \frac{2|(s, t)|}{(\deg(v)(\deg(v)-1))}$;
 $(v, s), (v, t), (s, t) \in E'$.

The final group calculates all cliques in the graph using a Matlab implementation of [3]. It includes three following features:

10. the *number* $clq(v)$ of cliques the node v is in;
11. a *raw clique score* where each clique in $clq(v)$ of size n is given a weight 2^{n-1} ,
 $CS_R(v) = \sum_{q \in clq(v)} 2^{size(q)-1}$;
12. a *weighted clique score* where each clique is weighted by the sum of activities of its members,

$$CS_w(v) = \sum_{q \in clq(v)} 2^{size(q)-1} \sum_{w \in q} m(w).$$

All scores are scaled to a value in $[0, 1]$ range, where 1 indicates a higher importance. Once all network features for all nodes in the graph, we can proceed by combining them with the text representation following one of fusion strategies. One is the feature fusion that enriches the tf-idf representation of pages with their network features. Another is the *model fusion* which assumes training a classifier with network features only and them combine its predictions with those of the text-based classifier.

4 Multilabel classification

Single-label classification is concerned with learning from a set of examples $\mathbf{x}_i \in X$ that are associated with a single label y from a set of k disjoint labels Y . Cases $k = 2$ and $k > 2$ are known as *binary* and *multi-class* classification. In *multilabel* classification, example \mathbf{x}_i are associated with a set of labels from Y . We will present sets as binary vectors $\mathbf{y}_i = (y_1, \dots, y_k)$, where $y_i \in \{0, 1\}$.

Most techniques for multi-label classification follow either the transformation or adaptation approach, see [14] for the review on multilabel learning. Like in the multi-class setting, there is no method performing well in all cases. When working with the INEX09 XML Mining Challenge dataset, we tested a number of different multilabel

techniques, discovered their complementarity, and tried to combine them in a robust manner.

There exists a number of multilabel techniques [15, 6, 13, 14]. Below we consider some of them as base classifiers:

One-against-all (IAA) is the most common approach to multi-label classification. It transforms the problem in k binary problems and train k independent classifiers h_i , each deciding to label a given \mathbf{x} with y_i ; then some ranking or thresholding schemes are used. This technique is easy to implement, it is fast but makes a strong assumption of independence among $y \in Y$.

Length-based One-against-all (L-IAA) is a modification of the IAA guided by the length prediction [13]. It trains additionally a length predictor h_l on the $\{\mathbf{x}_i, |\mathbf{y}_i|\}$ training set where $|\mathbf{y}|$ is the numbers of 1's in \mathbf{y} . It assumes a probabilistic setting for all h_i . h_l first predicts length l for a given \mathbf{x} , then it is labeled with y_i having top l scores. The L-IAA often improves the IAA performance and is scalable. However, it is very sensitive to the performance of h_l , it still assumes independence of $y \in Y$.

Unique multi-class (UMC) takes each label set \mathbf{y} present in T as a unique label. It is easy to implement. Yet, it often results in the exponential number of unique labels, with very few examples per label; no generalization to label sets not observed in the training set.

Latent variables in Y discovers the relevant groups (topics) among y and can be replaced the groups of by topic. The technique requires apriori the number of y -topics and often incurs important loss when decoding from topics to labels.

Latent variables in (\mathbf{x}, \mathbf{y}) is aimed at discovering topics in the (X, Y) space [17]. Latent semantic indexing (LSI) is a well-known unsupervised approach for dimensionality reduction in information retrieval. However if the output information (i.e. category labels) is available, it is often beneficial to derive the indexing not only based on the inputs but also on the target values in the training data set. This is of particular importance in applications with multiple labels, in which each document can belong to several categories simultaneously. It develops the multi-label informed latent semantic indexing (MLSI) algorithm which preserves the information of inputs and meanwhile captures the correlations between the multiple outputs. The recovered "latent semantics" thus incorporate the human-annotated category information and can be used to greatly improve the prediction accuracy. The main disadvantage of this technique is a limited scalability.

5 Evaluation

When preparing our submission to the challenge, we run a set of preliminary experiments using 5-fold cross validation on the core training set which the pages having both text-based and link-based features. We evaluated the performance of each tested method using the average Micro-F1, Macro-F1 and Exact Match measures. In the experiments we tested different components of the learning system as follows:

Feature set : three possibilities include the tf-idf feature set, the graph feature set and their fusion.

Basic learning method : among different options, we considered the libsvm with linear and other standard kernels, and semi-supervised learning with the transductive SVM we tested in the 2008 evaluations [4].

Multilabel method : four possibilities include one-against-all method and its length-based extension, unique multi-class and Y -latent variables.

Table 5 reports the most important results of the preliminary tests. Beyond the three components reported above, we run a number of tests were to test other components and techniques, including the feature selection, classification fusion, etc. However, none of them improved the performance, and their results are not included in the table.

Feature set	Method	Multi-label	Avg Exact Match	Avg Micro-F1	Avg Macro-F1
Tf-idf	Linear C-SVM	1AA	56.66	58.43	54.44
	Linear C-SVM	L-1AA	56.69	58.12	54.61
	Linear C-SVM	UMC	56.81	58.54	54.60
	Linear C-SVM	Y -latent	52.17	58.63	53.32
Graph	Linear C-SVM	1AA	35.34	41.71	39.56
	Linear C-SVM	L-1AA	37.01	40.33	38.98
	Sigmoid C-SVM	UMC	37.41	40.63	38.38
Tf-idf+Graph	Linear C-SVM	1AA	57.34	60.76	56.79
	Linear C-SVM	L-1AA	57.84	61.13	57.13
	Linear C-SVM	UMC	58.10	61.24	57.36
	SSL-TSVM	L-1AA	51.73	52.25	51.27
	SSL-TSVM	UMC	51.11	52.19	51.51

Table 1. Preliminary evaluation results for different methods and feature sets.

5.1 Final submissions and evaluation

We have analysed the results of the preliminary tests in order to organize our submissions to the XML Mining classification track. Our submissions take on the most performant preliminary tests, they combine the fusion of tf-idf and graph features with the linear C-SVM and the L-1AA and UMC multilabel strategies. We applied an additional treatment to cope with the partial mismatch between category, link and tf-idf data. The test part of the category data include hundreds of pages missing either tf-idf or link data. To predict categories for these pages in the test set, we generated the classification models using an available features only.

The final track evaluation results are reported in Figure 4. The plot tracks all seven measures for all research teams which participated in the challenge. As the figure shows, our submissions (labeled 'Xerox') came out first according to the four following measure: Micro-F1 and Macro-F1, Micro-Acc and Macro-Acc. The final performance results are comparable to the results of the preliminary tests.

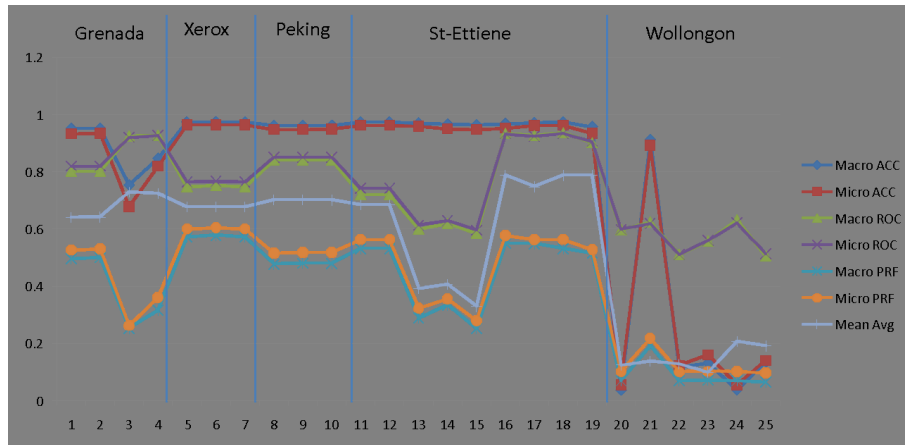


Fig. 4. Track evaluation results.

6 Conclusion

We presented our submission to the INEX 2009 XML Mining track that addresses the multilabel classification of the Wikipedia corpus. We reported a number of complementary techniques which allowed us to make a number of steady improvements over the baseline classification model. An analysis of the page network of allowed us to identify and extract a number of features valuable for the classification. Then we tested different fusion methods to combine the content-based and link-based features. We finally implemented and tested a number of alternative multilabel techniques in order to determine which one performs best on the evaluation corpus. We reported preliminary results obtained with different combinations on the training set of the Wikipedia corpus. We also included the final evaluation results which confirm the top performance of our submission according to four core measures, Micro-F1 and Macro-F1, Micro-Acc and Macro-Acc.

7 Acknowledgment

This work is partially supported by the SHAMAN Integrated Project co-financed by the European Union within the Seventh Framework Programme.

References

1. Julien Ah-Pine and Guillaume Jacquet. Clique-based clustering for improving named entity recognition systems. In *EACL*, pages 51–59, 2009.
2. Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.
3. Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

4. Boris Chidlovskii. Semi-supervised categorization of wikipedia collection by label expansion. In *INEX*, pages 412–419, 2008.
5. Paul F. Evangelista, Mark J. Embrechts, and Boleslaw K. Szymanski. Taming the curse of dimensionality in kernels and novelty detection. In *Advances in Soft Computing : Applied Soft Computing Technologies: The Challenge of Complexity*, pages 425–438. Springer Verlag, 2006.
6. Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200, New York, NY, USA, 2005. ACM.
7. David Gleich. MatlabBGL: a Matlab graph library. http://www.stanford.edu/~dgleich/programs/matlab_bgl/, 2008.
8. Thorsten Joachims. A statistical learning model of text classification for Support Vector Machines. In *Proc. 24th International ACM SIGIR Conf.*, pages 128–136. ACM Press, 2001.
9. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
10. M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.
11. D. Riehle. How and why Wikipedia works: an interview with Angela Beesley, Elisabeth Bauer, and Kizu Naoko. In *WikiSym '06: Proceedings of the 2006 international symposium on Wikis*, pages 3–8, New York, NY, USA, 2006. ACM.
12. Ryan Rowe, German Creamer, Shlomo Hershkop, and Salvatore J. Stolfo. Automated social hierarchy detection through email network analysis. In *Proc. 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, pages 109–117, New York, NY, USA, 2007. ACM.
13. Lei Tang, Suju Rajan, and Vijay K. Narayanan. Large scale multi-label classification via metalabeler. In *WWW '09: Proceedings of the 18th international conference on World Wide Web*, pages 211–220, New York, NY, USA, 2009. ACM.
14. Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.
15. US7412425. Partially supervised machine learning of data classification based on local-neighborhood laplacian eigenmaps.
16. F. Wu and D. S. Weld. Autonomously semantifying Wikipedia. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50, New York, NY, USA, 2007. ACM.
17. Kai Yu, Shipeng Yu, and Volker Tresp. Multi-label informed latent semantic indexing. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 258–265, New York, NY, USA, 2005. ACM.

Clustering with Random Indexing K-tree and XML Structure

Christopher M. De Vries, Shlomo Geva, and Lance De Vine

Faculty of Science and Technology,
Queensland University of Technology, Brisbane, Australia
`chris@de-vries.id.au` `s.geva@qut.edu.au` `l.devine@qut.edu.au`

Abstract. This paper describes the approach taken to the clustering task at INEX 2009 by a group at the Queensland University of Technology.

1 Introduction

The goal of the clustering task this year is to evaluate clustering as a means to perform collection selection. This involves minimising the spread of relevant results for adhoc queries over a clustering solution. The solution must also maximise the number of clusters while minimising the spread of relevant results. This is because the goal of collection selection is to distribute the collection over multiple machines. The more clusters that can be found, the more machines documents can be placed on without data redundancy.

The Random Indexing (RI) K-tree has been used to cluster the collection as it provides a scalable approach to clustering large collections at multiple granularities. The latest Wikipedia collection has included semantic markup mined by YAGO. This structure has been used via two simple approaches.

2 Random Indexing K-tree

The Random Indexing (RI) K-tree has been used to cluster the entire 2,666,190 XML documents in the INEX 2009 Wikipedia collection. Clusters were created as close as possible to the 100, 500, 1000, 5000 and 10000 clusters required for evaluation. The RI K-tree produces clusters in an unsupervised manner where the exact number of clusters can not be precisely controlled. It is determined by the tree order and the randomized seeding process. The algorithm produces clusters of many sizes in a single pass. The desired clustering granularity is selected by choosing a particular level in the tree.

Random Indexing (RI) is an efficient dimensionality reduction technique that projects points in a high dimensional space onto a randomly selected lower dimensional space. It is able to preserve the topology of the points. In the context of document representation, topology preserving dimensionality reduction is preserving document meaning or at least this is the conjecture which we test here. The RI projection produces dense document vectors that work well with the K-tree algorithm.

3 Run-time performance

The performance of RI K-tree has been measured when operating in main memory. The concern is with the performance of the clustering algorithm. Efficiency was not taken into account when indexing or loading the final representation into memory.

The RI operations took a total of 1860 seconds for the entity text representation. The randomly selected lower dimensional space had 1000 dimensions.

The run time of the K-tree algorithm varies between 1200 and 1500 seconds depending on the tree order selected between 15 and 50. This includes the process of reinserting all vectors to their nearest neighbour leaves upon completion of the tree building process. This produces clustering at many different granularities at once. Table 1 lists the different sized clusters found by a tree of order 40.

Level Clusters	
1	3
2	89
3	1260
4	12865
5	154934

Table 1. K-tree Clusters

4 Document Representation

Document structure has been represented by using a bag of words and a bag of tags representation derived from the semantic markup in the INEX 2009 collection. Both are vector space representations. The bag of words is made up of term frequencies contained within any entity tags in the collection. The term frequencies were weighted with BM25 where $K1 = 2$ and $b = 0.75$. The bag of tags representation is made up in an analogous manner of XML entity tag frequencies. The tag frequencies were not weighted. Both representations exploit the structure of the collection. The entity text does this by only selecting certain sections of the collection based on structure. The entity tags directly exploit structure by indexing it.

Submission were made using each of the representations separately. There is potential for improvement when combining both of the representations as the tags identify concepts such as scientist, acceptance and falsehood. The text contained within these tags can help differentiate different types of concepts.

Supervised Encoding of Graph-of-Graphs for Classification and Regression Problems

S. Zhang¹, M. Hagenbuchner¹, F. Scarselli⁴, A.C. Tsoi²

¹ University of Wollongong, Wollongong, Australia.

Email: {sz603, markus}@uow.edu.au

² Hong Kong Baptist University, Hong Kong. Email: act@hkbu.edu.hk

³ University of Siena, Siena, Italy. Email: franco@ing.unisi.it

Abstract. This paper introduces a novel approach for processing a general class of structured information, viz., a graph of graphs structure, in which each node of the graph can be described by another graph, and each node in this graph, in turn, can be described by yet another graph, up to a finite depth. This graph of graphs description may be used as an underlying model to describe a number of naturally and artificially occurring systems, e.g. nested hypertexted documents. The approach taken is a data driven method in that it learns from a set of examples how to classify the nodes in a graph of graphs. To the best of our knowledge, this is the first time that a machine learning approach is enabled to deal with such structured problem domains. Experimental results on a relatively large scale real world problem indicate that the learning is efficient. This paper presents some preliminary results which show that the classification performance is already close to those provided by the state-of-the-art ones.

1 Introduction

The emergence of neural network models capable of encoding graph structured information opened an avenue to solving machine learning problems involving graphs without the need for a pre-processing step in “squashing” the graph structure back into a vectorial form first. These methods are capable of encoding topological information which is available when dealing with structured information, and have been applied with considerable success in a number of real world problems. For example, an MLP based approach known as Recursive Neural Networks along with the training algorithm known as Backpropagation Through Structure (BPTS) ⁴ was used to solve an image classification problem by processing a set of images represented as a set of directed trees [2]. Similarly, recursive cascade correlation (RCC) [3, 4] was used to solve a regression problem by discovering structure-activity relationships of chemical molecules [5]. Another supervised machine learning method is the Graph Neural Network (GNN) which was shown to be capable of solving any practically useful learning problems involving graphs [6]. There are also unsupervised neural network methods for the encoding of structured information. For example, the Self-Organizing Map for

⁴ The term introduced by [1] is somewhat misleading since BPTS is restricted to the processing of trees. Hence, a more appropriate name would be the Backpropagation Through Trees (BPTT).

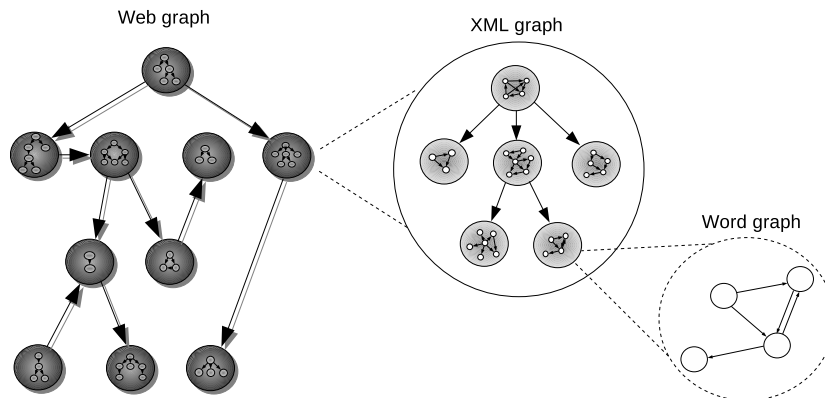


Fig. 1. An example of a graph of graphs structure: Each node in a Web graph represents a document. The document may be represented by an XML graph [7]. The nodes in the XML graph map represent paragraphs where the text in each paragraph may be represented by a word graph [11].

Structured Data (SOM-SD) is capable of clustering tree structured information very efficiently, whereas the Probability Measure Graph SOM (PMGraphSOM) is able to encode much more generic types of graphs such as labeled cyclic graphs which may have feature directed or undirected links. These methods have been very successful in solving practical problems and benchmarking with other algorithms. For example, the SOM-SD and its related algorithms have won several international INEX competitions in text mining (e.g. [7, 8]) and has set state-of-the-art performances (e.g. [9, 10]).

These existing methods are limited to processing graphs which can be represented by labeled nodes and labeled links. The label attached to nodes and links must be a real valued vector of fixed dimension. However, it is found that there are learning problems for which the label of a node is of a more complex structure, such as a tree or graph. Such learning problems require the encoding of a graph structure the nodes of which can also be graphs. This results in data featuring a *graph of graphs* (GoG) which contains different graphical elements. The situation is best explained using an example (see Figure 1): The World Wide Web consists of a collection of documents which feature a referencing method known as *hyperlinks*. A hyperlink defines a (directed) binary relationship between two documents.

The collective combination of all hyperlinks produces a *Web graph* the nodes of which represent the documents, and the links represent the hyperlinks. The nodes in such a graph can be labeled to describe properties and content of the associated document. In the case of typical Web documents, one may represent the associated document as a graph consisting of nodes representing sections of text encapsulated by HTML formatted elements, and links representing the encapsulation of the HTML formatted elements [7]. The nodes of this document graph may be labeled by a word graph consisting of nodes which may represent the word token, or a set of word tokens, and links among the nodes indicating the relationship between two sets of word tokens [11]. Now, say, in a particular paragraph, there is a hyperlinked document linked to it. Such hyperlinked

document in itself can be represented by a graph. This may be represented as a graph within a node of the parent document. This is a graph of graphs (GoG) structure. Obviously there is a possibility that the hyperlinked document also contains hyperlinks to other documents, and these documents in turn, can be represented as graphs within the nodes in the hyperlinked document graph. We note that a GoG is generally of a hybrid nature since the graphs at different levels encode different relationships, and describe different atomic elements. Hence, the GoG is an *embedded hybrid graph* whose components may differ significantly in properties. For example the hybrid graph depicted in Figure 1 features a Web graph which is a directed cyclic graph, XML graphs which are tree structured, and a word graph which may be labeled and undirected.

To the best of our knowledge, there is currently no known approach to machine learning which can directly encode such GoG structures. This paper proposes an approach to modeling such structures, and a supervised learning algorithm is proposed by generalizing an existing supervised machine learning method, viz. backpropagation through structures to encode such structures. The proposed method is recursive, in that it will process one level after the other recursively from the innermost level to the outermost level, and, hence, it will be able to encode GoG structures to any finite depth (e.g. in the case of Figure 1 the depth of the GoG structure is 3).

The structure of this paper is as follows: Section 2 introduces a formal description of the GoG structure, and proposes a data driven approach to encode such structures. Section 3 presents experimental results, and some observations on the limitations of the approach. Finally, Section 4 concludes this paper and provides an outlook for further research in this area.

2 Encoding graph of graphs

In this section a formal representation of GoG will be provided. Consider the following situation: we have a node i in a graph with a neighbourhood $\mathcal{N}_{[i]}^0$, which consists of a number of other nodes, where the superscript 0 denotes the topmost level, the parent document level. The nodes in the neighbourhood $\mathcal{N}_{[i]}^0$ all have connections via links to the node i . Each node is described by a set of features, and each link is described by yet another set of features. Each node could have an external input. If we assume that each node is described by an entity called *state* then node i is described by a state \mathbf{x}_i^0 , an n_0 -dimensional vector, The state of node i can be described by Equation 1:

$$\mathbf{x}_i^0 = \mathcal{F}_i^0(\mathbf{u}_i^0, \mathcal{C}_0(\mathbf{x}_{\mathcal{N}_{[i]}^0}^0), \mathbf{x}_i^1) \quad (1)$$

where \mathbf{u}_i^0 is the input to node i , $\mathcal{F}_i^0(\cdot, \cdot, \cdot)$ is a nonlinear vector function (which may be considered a hyperbolic tangent function, or a sigmoid function), and \mathcal{C}_0 denotes the connections from the neighbourhood $\mathcal{N}_{[i]}^0$ and the vector \mathbf{x}_i^1 into the i -th node via connecting links. \mathbf{x}_i^1 is an n_1 -dimensional vector denoting the *state*⁵ of the graph which is attached to the i -th node. In other words, here we assume an additive model, in which the (state of the) graph attached to a node i is assumed to be an additional input to the

⁵ Here we abuse the notion of state \mathbf{x}_1 for simplicity sake. The state here can be the output of the graph, or the concatenation of the individual states of the nodes in the child graph.

node i . The state \mathbf{x}_i^1 is described similarly as follows:

$$\mathbf{x}_i^1 = \mathcal{F}_i^1(\mathbf{u}_i^1, \mathcal{C}_1(\mathbf{x}_{N_{[i]}}^1), \mathbf{x}_i^2) \quad (2)$$

where \mathbf{x}_i^1 denotes the level 1 state ⁶, that is the state of the graph (or output of the graph) representing the child document of the i -th node; \mathbf{u}_i^1 denotes the input into the level 1 node; \mathcal{F}_i^1 denotes a nonlinear vector function; \mathcal{C}_1 denotes the connections into the node i in the child graph; and \mathbf{x}_i^2 denotes the state of the child graph associated with node i in level 1. From Eq. (1) and Eq. (2) the recursive nature of the approach becomes clear. The recursion stops at the maximum level of encapsulation of graphs. If we assume that there are k levels, then the k -th level will be described by the following equation:

$$\mathbf{x}_i^k = \mathcal{F}_i^k(\mathbf{u}_i^k, \mathcal{C}_k(\mathbf{x}_{N_{[i]}}^k)) \quad (3)$$

At the k -th level by assumption there will not be any inputs from the graph within the node i . Thus k denotes the terminal level; the depth of the GoG structure. Then, a mapping from the state space to the output space takes place as follows:

$$\mathbf{o}_i^0 = \mathcal{G}_i^0(\mathbf{u}_i^0, \mathcal{B}(\mathbf{x}_i^0)), \quad (4)$$

where \mathcal{B} denotes the configuration of the state vector \mathbf{x}_i^0 with the output vector \mathbf{o}_i^0 . The output can then be compared to an associated target value, and a gradient descent method can be applied to update the system with the aim to minimize the squared difference between network output and target values. It is noted that similar equations to Eq. (4) can be written to provide an output to any of the k levels.

Note that Eq. (4) is suitable for node focused applications. With node focused applications, a model is required to produce an output for *any* node in a graph. In contrast, graph focused applications require *one* output for each graph. In the literature, a graph focused behaviour of such systems is achieved by either selecting one node (i.e. the root node) to be representative for the graph as a whole [1, 2], or by producing a consolidated mapping from all states [6]. The same principles can be applied here. It is important to note that the model at any level other than level 0 are graph focused ones since the state of the graph (as a whole) which is associated to a given node i is forwarded as an input to node i . In contrast, the model at level 0 can be either a graph focused one or a node focused one depending on the requirements of the underlying learning problem.

It is trivial to observe that the GoG model accepts the common graph model as a special case. Indeed, if $k = 1$, this collapses to the standard graph model considered in [6]. Since the graph model in [6] contains the time series as a special case (a tree), and hence one may observe that the GoG model may be considered as the most general graph model formulated to represent objects so far. An MLP can be used to compute the states in each layer. Hence, the model consists of k MLPs which have forward (and backward) connections to the MLP at the next level.

⁶ Again we abuse the notion of state here, as it can represent either the output of the child graph associated with node i in level 0, or the concatenation of the individual states of the nodes in the child graph.

Once the GoG model is expressed in recursive form as shown in Eqs. (1) to (3), then it is quite clear how the unknown parameters ⁷ in the model can be trained using the standard backprop algorithm. The training algorithm can be stated as follows:

- Step 0** Initialization. The parameters in the model are initialized randomly.
- Step 1** From the deepest level k compute the state, and progressively compute the states in levels $k - 1, k - 2, \dots, 0$.
- Step 2** Compare the outputs at the topmost level 0 with those of the desired ones, and form an error function.
- Step 3** Backprop the error from the topmost level through the levels until we reach the innermost level k and update the parameters of the model in the process.

Step 1 through to step 3 are repeated for a limited number of times, or until the sum of squared errors is below a prescribed threshold, then the algorithm stops.

This algorithm while much more complex in terms of notations and concepts, nevertheless is in the same spirit as the backprop through structure algorithm. Hence we will call this algorithm backprop through structures, though in this case, it is the levels that one is concerned with.

3 Experiments

The proposed approach is applied to a real world problem involving hyperlinked documents. The experiments are carried out as part of a participation at the INEX (INitiative for the Evaluation of XML Retrieval) competition on semi-structured document mining for the purpose of classification (the INEX 2009 competition). The dataset provided by INEX is a collection of XML formatted documents from the online encyclopedia, Wikipedia. The documents are interlinked via a xref or hyperlink structure. INEX has provided a subset of Wikipedia for the classification task. The dataset consists of 54,889 documents. A target label is available for 11,028 of these documents. Hence, these 11,028 documents provide a supervising signal to the training algorithm. All remaining 43,861 are the test documents. The task is to classify the 43,861 documents for which no target is available. The documents are interlinked. However, we found that only 54,121 documents contain outgoing links. The maximum out-degree (the maximum number of outgoing links for any one document) for this dataset is 2,382, the maximum in-degree is 27,518. the dataset contains a total of 4,554,203 links. For simplicity, we removed redundant links (if a document contains several links to the same document, then we count only one such link). The removal of redundant links reduced the maximum out-degree quite significantly to 969, the maximum in-degree to 15,027, and the total number of links to 3,368,504. Each document can also be described by the features extracted.

⁷ Here we assume that the function \mathcal{F}_i^k is characterized by a set of parameters. For example, if the encoding mechanism is a multilayer perceptron, then the unknown parameters will be the strengths of connections from the inputs to hidden layer neurons. In the case if we use the outputs then the set of parameters will include the strengths of connections from the hidden layer neurons to output neurons. For simplicity, we will assume shared weight model, i.e. all \mathcal{F}_i^k in the same level k share the same set of weights.

The result is a graph whose nodes represent the XML documents, and the links represent the hyperlinks. The nodes can be labeled by a combination of the following features:

- **XML tag tree:** Each document is XML formatted. XML is a language which describes the structure of a document, and is naturally represented by a XML (parsing) tree. The tree consists of nodes which represent the XML tags, and links which represent the nesting of the tags. The nodes are labeled by an identifier which uniquely identifies the associated tag. The way to extract the XML tree from XML documents is described in [9].
- **XML tag graph:** Each node in the graph represents a unique tag within the document, and edges represent the relationship between tags. We then apply a common procedure for processing text documents called “rainbow” [12,9] on tags ⁸ and compute the information gain for each with respect to different categories. Top 100 tags with highest information gain were selected and attached to the nodes in the tag graph as node labels.
- **Concept Link graph:** Concept link graph [13] is a novel text representation scheme which encodes the contextual information of a document using a graph of concepts. Specifically, for a document d , it is represented as a weighted, undirected graph $d = \{N, E\}$ where N is the set of nodes representing the concepts, and E is the set of edges representing the strength of association among concepts. The ConceptLink graph extraction method is described in [13].
- **Term frequency vector:** INEX provided a Term-Frequency vector (presumably obtained by the well-known Bag-of-Words algorithm). The i -th element of the vector lists the number of occurrences of the i -th dictionary word. Hence, the vector encodes the textual content of a given document.
- **Rainbow classification results:** The well-known Rainbow software (which implements the Bag of Word algorithm) was used to classify the test documents over all categories. That is, for each category, we split the training dataset into two classes, one belonging to the category, while the other does not belong to the category. In this manner, Rainbow can produce a probability vector for each test document against all categories.

Thus, we considered to label the nodes in the hyperlink graph by either graphs describing the document of the associated node, by a vector describing the content of the same document, or both.

We extracted the XML structure for each of these documents, then attached the XML structure as a label to the associated node in the hyperlink graph. Thus, there are a total of 54, 575 XML structures. The maximum out-degree of any of the XML structures is 1, 006, and the total number of nodes is 42, 668, 059. The latter is equivalent to the total number of XML tags in the dataset. We removed redundant tags by consolidating successively repeated XML tags and obtained a somewhat reduced XML graph with a maximum out-degree of 533. We found that only very few XML trees have an out-degree larger than 60. Hence, we truncated the out-degree to 60 since any algorithm driven by back-propagation discards sparse information anyway.

⁸ Rainbow is a text classifier based on Naive Bayes approach.

Alternatively, we extract concept link graph from each document in the dataset, and used them as the bottom level graphs. Concept Link Graph is generated in three main steps [13]. First, we discover a set of concepts by clustering related words extracted from the training documents using self-organizing maps. Secondly, given the term clustering result, for each paragraph in a document, we map every existing word to a concept to which it belongs. Then we count the occurrence of every concept paragraph by paragraph. Given the paragraph-based occurrence statistics, we represent each document using a concept-paragraph matrix. Finally, singular value decomposition (SVD) is applied to the concept-paragraph matrix to compute a concept-concept association matrix. Formally speaking, given a matrix A as a concept-paragraph matrix of m concepts and n paragraphs, decomposing A using SVD returns $(U\Sigma V)^T$, where U and V are unitary matrices and Σ is a diagonal matrix with elements arranged in a descending order of magnitudes. Given the SVD result, Σ is interpreted as the “theme” matrix, where each of its diagonal elements represents the strength of its corresponding theme, U is the concept-to-theme relevance matrix and V is paragraph-to-theme relevance matrix. The concept link graph is thus obtained by computing the concept-to-concept association matrix $AA^T = U\Sigma^2V^T$.

The (given) term frequency vector is of dimension 186,723 which corresponds to the 186,723 dictionary words found in the dataset. There is one term frequency vector for each document, and hence, this can also be used to label the nodes in the hyperlink graph. The term frequency vector is very sparse because not every document contains all dictionary words. We were able to reduce the dimensionality of the term frequency vector to 439 by building a matrix of category and feature (the term), then counted the number of documents which belong to a category containing the feature. We retained only those features which exhibited standard deviation of larger than 10. We also used another approach to reduce the term frequency vector: We computed the information gain for each word in the dataset with respect to the different classes, then five words with the highest information gain are selected per class. We ended up with 133 unique words, thus producing a 133 dimensional feature vector.

The classification problem is defined by 39 classes. The classes are known to represent categories to which a document belongs. This produces a 39 dimensional target vector containing binary elements. Note that the target vector may contain several non-zero elements. This indicates that the corresponding document can belong to several categories. We note also that the distribution of the different categories is not balanced (See Table 1). The largest category contains 1,337 documents, the smallest category contains 191 documents, and the average number of documents per category is 414.

As a result, this dataset is described by a GoG which is similar as was depicted in Figure 1, and is consisting of two levels:

- Level 0: One graph per document, describing the structure, contents or other properties within the document.
- Level 1: One graph where documents are nodes, connected via links representing the hyperlinks between documents.

We trained the proposed machine learning method on the resulting GoG, and varied the labeling mechanisms as is illustrated in Table 2 to identify the impact of these features on the classification performance. We also varied the number of state neurons

Table 1. The number of documents belonging to the 39 different categories is unbalanced

History	Space	World War I	Weather	Philosophy	Religion	Physics
534	490	262	222	275	438	271
Video games	Saints	Tropical cyclones	Politics	Cricket	Food	Trains
224	299	302	386	241	200	1337
Chess	Chemistry	Medicine	War	Literature	Baseball	Christianity
199	191	236	963	971	286	270
Astronomy	Horror	Pornography	Geography	Nautical	Catholicism	Music
216	280	220	216	232	1240	351
Science	Anarchism	Architecture	Bible	Business	Aviation	Biography
205	349	336	380	207	595	1053
Comics	Formula One	American Civil War	Pharmacy			
522	319	329	518			

and hidden layer neurons to identify the impact of the number of internal network parameters on the classification performance. We also allowed the model to use some mechanisms based on a balanced training dataset in order to improve the results: Balance dataset (See Section 3.1) and label-to-out approach (See Section 3.2).

3.1 Balancing labeled data

The labeled data is not balanced among different categories. This is known as a *desirable problem* in machine learning due to noise tolerant abilities of such systems. In order to suitably encode a learning problem which features classes which are much smaller than other classes, then counter measures need to be taken so as to avoid that the network dismisses small classes as *noise*. In order to avoid this, it is useful to balance the distribution of train samples. Since this is a multiple categories task, the traditional methods for balancing data is not applicable. Instead of complementing the number of samples from smaller classes, we modified the error back-propagation algorithm by altering the error which is propagated back during training as follows: For each train sample, the network produced an output vector which has dimension as the number of categories available, and each element in the vector represents the output for corresponding category. Originally for each element i in the vector, the error $\varepsilon_i = o_i - t_i$ is computed. In order to balance train samples from category i , ε_i can be revised by using number of negative samples N_n and number of positive samples N_p for category i , there are two alternative ways which have the same effect on the training algorithm:

1. Method 1: if $t_i = 1$, then $\varepsilon_i = (o_i - t_i) \times N_n$; if $t_i = 0$, then $\varepsilon_i = (o_i - t_i) \times N_p$
2. Method 2: if $t_i = 1$, then $\varepsilon_i = (o_i - t_i) \div N_p$; if $t_i = 0$, then $\varepsilon_i = (o_i - t_i) \div N_n$

3.2 Label-to-out Approach

We further considered a slight modification of the proposed algorithm, namely the addition of direct links between node label and the output layer⁹. This additional layer of

⁹ There is evidence in the multilayer perceptron situation that a direct feedforward input to the output can improve the overall performance of the multilayer perceptron [14]. In the case of

Table 2. List of all input data files used for training

ID	Description
1	document-link graph. maxout=2382, maxin=27518, nodelabel=439 (reduced tfidf vector)
2	document-link graph. maxout=2382, maxin=27518, nodelabel=133 (word counts)
3	document-link graph. maxout=969, maxin=15027, nodelabel=133
4	GoGs. Level 0: tag graphs, maxout=195, nodelabel=1 (tag id); level 1: document-link graph, maxout=969, maxin=15027, nodelabel=encoding of level 0 tag graphs.
5	GoGs. Level 0: tag graphs, maxout=52, nodelabel=39 (tag information gain); level 1: document-link graph, maxout=969, maxin=15027, nodelabel=encoding of level 0 tag graphs.
6	GoGs. Level 0: tag graphs, maxout=52, nodelabel=39 (tag information gain); level 1: document-link graph, maxout=969, maxin=15027, nodelabel=encoding of level 0 tag graphs+133(word counts).
7	GoGs. Level 0: concept link graph, maxout=41, nodelabel=51; level 1: document-link graph, maxout=969, maxin=15027, nodelabel=encoding of level 0 tag graphs+39 dimensional label from classification results of rainbow for each category respectively (NaiveBayes).
8	GoGs. Level 0: concept link graph, maxout=41, nodelabel=51; level 1: document-link graph, maxout=969, maxin=15027, nodelabel=encoding of level 0 tag graphs+39 dimensional label from classification results of rainbow for each category respectively (NaiveBayes with logarithm normalization).

network weights allows the treatment of labels as independent vectors. The effect is a bias input to the output layer, one for each node in the graph. Note that this approach only affects the level 1 graph when the nodes have been produced by rainbow. By using this label-to-out approach, we anticipate that the difficulty of the training task will become simpler to encode by the network. The assumed was confirmed by experimental results. The label-to-out approach and the GoGs of concept link graphs (shown as experiment 7 in Table 3) allowed us to obtain the best results so far: MAP= 0.68.

3.3 Results

Training performance is evaluated according to two measures: the MAP (mean of the average precision) with respect to the document and the Macro F_1 score.

- MAP, the mean of the average precision with respect to each document. This measure is used to evaluate whether the system is capable of retrieving highly relevant categories first. For each document i , we obtain a list of relevant categories by sorting the scores in an ascending order of magnitudes, and then compute:

$$AvgP_i = \frac{\sum_{r=1}^N (P(r) \times rel(r))}{N_{relevant}} \quad (5)$$

MAP can be in turn computed as:

$$MAP = \frac{\sum_i AvgP_i}{n}, \quad (6)$$

a direct state to output connection as suggested here, there is yet a formal proof that this will produce improved results.

Table 3. Results by using different training configuration

	Training Configuration	MAP	PRF		ROC		ACC	
		mean	macro	micro	macro	micro	macro	micro
1	graph 6; state=20, hidden=15, outhidden=20, output=10; rprop; seed=45	0.125	0.07	0.10	0.59	0.59	0.03	0.05
2	graph 6; state=20, hidden=15, outhidden=20, output=10; weight control; rprop; seed=45	0.138	0.19	0.21	0.62	0.61	0.91	0.89
3	graph 6; state=15, hidden=10, outhidden=15, output=5; weight control; seed=3	0.129	0.06	0.10	0.51	0.51	0.11	0.12
4	graph 3; state=10, hidden=8, outhidden=6, output=39; balance method 1; seed=37	0.10	0.07	0.10	0.55	0.56	0.13	0.16
5	graph 3; state=30, hidden=20, outhidden=15, output=39; balance method 1; seed=1	0.208	0.07	0.10	0.63	0.62	0.03	0.05
6	graph 3; state=10, hidden=8, outhidden=6, output=39; balance method 2; seed=9	0.192	0.06	0.09	0.50	0.51	0.13	0.14
7	graph 8; state=6, hidden=8, outhidden=10, output=15; jacobian control; seed=91; label-to-out	0.68	0.48	0.51	0.83	0.85	0.95	0.93

where n is the total number of documents evaluated.

- Macro F-measure score, which is the evenly weighted precision and recall rates. For each category c , we computed:

$$F_c = \frac{2 \times (P_c \times R_c)}{P_c + R_c} \quad (7)$$

then we averaged the F-measure scores for all categories and obtain the Macro F-measure scores.

A range of network architectures and training parameters were tried on this training task. A selection of these are given by Table 3. The main observations are as follows:

- Using label-to-out approach has improved the performance significantly. This indicates that the additional bias is effective in simplifying the given learning task. The simplification arises out of the fact that some features can influence the network output directly without having to travel through the relatively deep network architecture (the unfolded iteration network).
- Larger network is required to produce better results. The more neurons a network features, the more parameters are available to encode a given learning problem. The need for relatively large number of parameters indicates that given learning problem is non-trivial.
- Weight control through a Jacobian control mechanism helped to produce better results. This weight control mechanism can aid the training procedure by restricting the movement of weight changes such that the size of weight adjustments remains within a limited range. This aids the stability of the weights during the training, and hence, can result in a general improvement in the quality of the training procedure.

A comparison with other approaches submitted by other groups to this classification problem is given in Table 4. In this comparison, it can be seen that the proposed GoGs

Table 4. INEX2009 XML classification task results

Submission	MAP	PRF		ROC		ACC	
	mean	macro	micro	macro	micro	macro	micro
University of Wollongong	0.681	0.479	0.513	0.829	0.849	0.947	0.925
University of Peking	0.702	0.48	0.518	0.842	0.85	0.962	0.948
Xerox Research Center	0.678	0.571	0.6	0.748	0.765	0.974	0.963
University of Saint Etienne	0.788	0.53	0.564	0.937	0.935	0.974	0.962
University of Granada	0.642	0.50	0.53	0.802	0.819	0.952	0.933

Table 5. Best classes

Category	Correct Counts	Total Counts	Percentage
Portal:Pharmacy and Pharmacology	2490	2630	0.9468
Portal:Comics	2343	2583	0.9071
Portal:War	4193	4712	0.8899
Portal:Literature	4051	4688	0.8641
Portal:Trains	5487	6352	0.8638

approach produced a very reasonable performance. We are in fact very please by these results since these are preliminary results on a system which is still under development.

Table 5 and Table 6 list the categories for which we obtained the best classification result and worst classification result respectively. It can be seen that the better results are generally be obtained for larger classes. Given that we used ways to balance the dataset, and hence, this indicates that the smaller classes in the training set do not sufficiently well cover the problem domain.

4 Conclusions

In this paper, we have deployed a new idea, the graph of graphs model in modeling the connections among the linked documents in the XML dataset. This is quite a novel idea in that it allows us to extend the idea first discussed in graph neural networks [7] which can only be used to describe a single document. A GoG model can be used to model the linked set of documents by considering each document as a graph, and the linkages among the graphs are modeled as links which connecting the documents. Using an extension of the back propagation through structure algorithm, we were able to model the set of documents given in the training dataset as GoG models. This GoG model when combined with two sets of mechanisms: a balanced training dataset, and a

Table 6. Worst classes

Category	Correct Counts	Total Counts	Percentage
Portal:Pornography	477	1041	0.4582
Portal:Science	524	1101	0.4759
Portal:Nautical	570	1182	0.4822
Portal:Chess	517	1010	0.5119
Portal:Architecture	949	1777	0.534

label-to-out approach produced very reasonable performance when compared to those obtained by other research groups.

For future work, it is possible to vary some of the parameters in the GoG model, especially the number of state neurons in the model. This state information captures the past information in the training process. A larger number of state neurons will provide a richer model for the information contained in the documents, while a small number of state neurons will compress the information contained in the documents.

Acknowledgment: This work received financial support from the Australian Research Council through Discovery Project grant DP0774168 (2007 - 2009).

References

- [1] Gori, M., K uchler, A., Soda, G.: On the implementation of frontier-to-root tree automata in recursive neural networks. In: *IEEE Transactions on Neural Networks*. Volume 10(6). (November 1999) 1305–1314
- [2] Frasconi, P., Francesconi, E., Gori, M., Marinai, S., Sheng, J., Soda, G., A., S.: Logo recognition by recursive neural networks. In Kasturi, R., K. Tomre, L., eds.: *Second International Workshop on Graphics Recognition, GREC'97*, Springer-Verlag (1997) 104–117
- [3] Sperduti, A., Majidi, D., Starita, A.: Extended cascade-correlation for syntactic and structural pattern recognition. In Perner, P., Wang, P., Rosenfeld, A., eds.: *Advances in Structural and Syntactical Pattern Recognition*. Volume 1121 of *Lecture notes in Computer Science*. Springer-Verlag (1996) 90–99
- [4] Hammer, B., Micheli, A., Sperduti, A.: Universal approximation capability of cascade correlation for structures. *Neural Computation* **17**(5) (2005) 1109–1159
- [5] Bianucci, A., Micheli, A., Sperduti, A., Starita, A.: Quantitative structure-activity relationships of benzodiazepines by recursive cascade correlation. In: *IEEE Processing of IJCNN'98- IEEE World Congress on Computational Intelligence*, Anchorage, Alaska (May 1998) 117–122
- [6] Scarselli, F., Gori, M., Tsoi, A., Hagenbuchner, M., Monfardini, G.: Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks* **volume 20**(number 1) (January 2009) 81–102
- [7] Yong, S., Hagenbuchner, M., Tsoi, A., Scarselli, F., Gori, M.: Document mining using graph neural network. In: *Lecture Notes in Computer Science*. Volume 4518., Springer-Verlag Berlin Heidelberg (2007) 458–472
- [8] KC, M., Hagenbuchner, M., Tsoi, A., Scarselli, F., Gori, M., Sperduti, S.: Xml document mining using contextual self-organizing maps for structures. In: *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg (2007)
- [9] Hagenbuchner, M., Sperduti, A., Tsoi, A.: A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks* **14**(3) (May 2003) 491–505
- [10] Zhang, S., Hagenbuchner, M., Tsoi, A., Sperduti, A. In: *Self Organizing Maps for the clustering of large sets of labeled graphs*. Springer-Verlag Berlin Heidelberg, Berlin (2009)
- [11] Mihalcea, R., Tarau, P.: TextRank: Bringing order into texts. In: *Proceedings of EMNLP-04 and the 2004 Conference on Empirical Methods in Natural Language Processing*. (July 2004)
- [12] : The rainbow software package. <http://www.cs.umass.edu/mccallum/bow/rainbow/>
- [13] Chau, R., Tsoi, A.C., Hagenbuchner, M., Lee, V.: A conceptlink graph for text structure mining. In Mans, B., ed.: *Thirty-Second Australasian Computer Science Conference (ACSC 2009)*. Volume 91 of *CRPIT.*, Wellington, New Zealand, ACS (2009) 129–137
- [14] Sontag, E.: Capabilities and training of feedforward nets. In: *Neural networks* (New Brunswick, NJ, 1990), Boston, MA, Academic Press (1990)

Clustering XML documents using Multi-feature Model

Sangeetha Kutty Richi Nayak Yuefeng Li

Faculty of Science and Technology
Queensland University of Technology
GPO Box 2434, Brisbane Qld 4001, Australia

{s.kutty, r.nayak, y2.li}@qut.edu.au

Abstract. This paper presents the approach used to cluster XML documents in the XML Mining track at INEX 2009. A Multi-feature model (MFM) is built using the structural features and the corresponding content features of the XML documents. The model is then decomposed and the decomposition factors are used in deriving the clustering solution. Due to the very large size of the corpus and the presence of multiple features, the size of the MFM was very large. Hence, applying decomposition technique on these large MFMs to derive clustering solution was computationally very expensive and hence dimensionality reduction techniques were applied.

Keywords: Multi-feature model, Clustering, XML document mining, INEX, Wikipedia, Frquent subtrees, Structure and content.

1 Introduction

The XML Mining track included two tasks namely classification and clustering. Classification task labels the XML documents in the collection into known categories using training dataset. On the other hand, clustering is an unsupervised learning and it groups the documents without any knowledge of categories. INEX 2009 clustering task consisted of two corpuses namely the entire collection and the subset collection with 2,666,190 and 54,575 documents respectively. There were two types of evaluation used, standard criteria and collection-selection. Purity, entropy, F-score and Normalised Mutual Information were used as the standard criteria to evaluate the submissions. In the collection-selection goal, the clustering solutions were evaluated to determine the quality of cluster relative to the optimal collection selection goal, given a set of queries. The required number of clusters to be submitted were 100, 500, 1000, 2500, 5000 and 10000.

In this paper, we utilise a novel approach for clustering XML documents which involves consideration of two document input features or aspects, namely structure and content, for determining the similarity between them. The corpus includes both semantic and syntactic tags. These tags representing the structure of the XML documents are used to store the content. Hence, to derive meaningful clustering

results, it is essential to utilize both these XML document features. Most of the existing approaches do not focus on utilizing these two features due to increased computational storage and processing. Therefore, this study not only combines the structural and the content features of XML documents in Multi-feature model(MFM) effectively but also utilises an approach that helps to reduce the dimensions for clustering without considerable loss in accuracy.

In this paper, we model the XML documents in MFM space but the dimensionality of these models is quite large due to the size and nature of the corpus. Hence, we apply dimensionality reduction using random projection. Then decompositions are applied on the reduced search space and clustering solutions are derived.

2 An Overview

As illustrated in Fig.1, there are three major phases in the approach that we have adopted for the INEX 2009 Document Mining Challenge corpus. The Phase-I is the pre-processing of XML documents to represent their structure. Each XML document in the INEX Wikipedia corpus is parsed and modeled as a rooted labeled ordered *document tree*. Each document tree contains nodes which represent the tag names. A frequent subtree mining algorithm is then applied to extract the common structural features from these document trees in the corpus. The content corresponding to every subtree in the XML document is then extracted.

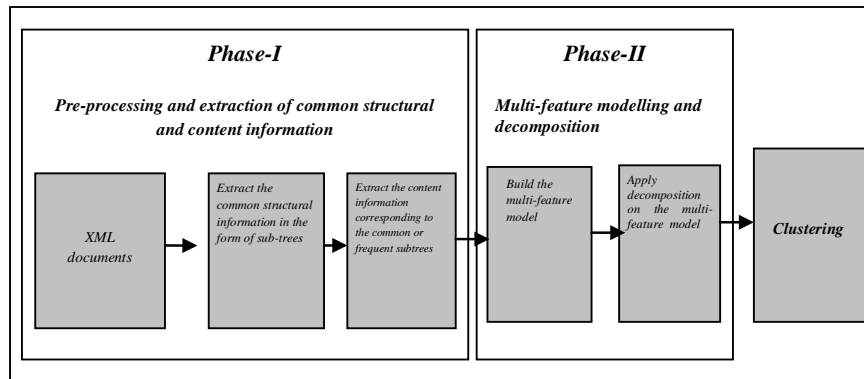


Fig. 1. Overview of the clustering method.

In Phase-II, the structure and the content for every document is represented in the three-feature model – Doc X Subtree X Term, where Term represents the terms corresponding to the Subtree in the XML document, Doc in the given corpus. Each cell in the model represents the number of occurrences of the terms for the given subtrees in a given XML document. This model is then used in the final phase to compute the similarity between the XML documents for the clustering the corpus.

UJM at INEX 2009 XML Mining Track

Christine Largeron and Christophe Moulin and Mathias Géry

Université de Lyon, F-42023, Saint-Étienne, France

CNRS UMR 5516, Laboratoire Hubert Curien

Université de Saint-Étienne Jean Monnet, F-42023, France

{christine.largeron,christophe.moulin,mathias.gery}@univ-st-etienne.fr

Abstract. This paper¹ reports our experiments carried out for the INEX XML Mining track 2009, consisting in developing categorization methods for multi-labeled XML documents. We represent XML documents as vectors of indexed terms. The purpose of our experiments is twofold: firstly we aim to compare strategies that reduce the index size using an improved feature selection criteria *CCD*. Secondly, we compare a thresholding strategy (*MCut*) we proposed with common *RCut*, *PCut* strategies. While the way we reduced the index size leads to bad results, we obtain good improvements with the *MCut* thresholding strategy.

1 Introduction

This paper describes the participation of Jean Monnet University at the INEX 2009 XML Mining Track. For the categorization task (or classification), given a set of categories, a training set of preclassified documents is provided. Using this training set, the task consists in learning the classes' descriptions in order to be able to classify a new document in the categories.

One main difference in the collection of documents provided in INEX 2009 relatively to INEX 2008 lies in the overlapping of the categories and in their dependencies [1, 2]. When each document belongs to one and only one category in INEX 2008, it can belong to several categories in INEX 2009. With the imbalance between the categories, their overlapping poses new challenges and gives opportunities for design machine learning algorithms more suited for XML documents mining.

In this article, we focus on the selection of the set of classes that will label a document for this multi-label text categorization. We explore two approaches. The first one uses a binary classifier which considers one category against the others. The algorithm returns two answers (yes or no) used to decide whether the document belongs or not to this category. In that case, the selection of a set of words characteristic of the category can be essential for improving the performance of the algorithm. Our first contribution to Inex 2009 consists in an improvement of the selection criteria that we have introduced in INEX 2008 and

¹ This work has been partly funded by the Web Intelligence project (région Rhône-Alpes, cf. <http://www.web-intelligence-rhone-alpes.org>).

which permitted to get the best results of the competition while reducing the index size [3]. The second approach uses a multi-label classifier which considers simultaneously all the categories. Given a document, the classifier returns a score (*i.e.* a numerical value), for each category. In the context of single label classification in which one and only one class must be attributed to each document the decision rule is obvious: it consists to return the class corresponding to the best score. On the contrary, in multi-label categorization, this approach raises the question of the number of classes that must be assigned to each document. In this article, we propose a thresholding strategy for selection of candidate classes and we compare it with the commonly used methods PCut and RCut [8, 5]. In the aim of introducing our notations, a brief presentation of the vector space model (VSM [6]), used to represent the documents, is given in section 2, the selection features criteria is defined in the following section. The thresholding strategy for selection of candidate classes is presented in section 4. The runs and the obtained results are detailed in sections 5 and 6.

2 Document model for categorization

2.1 Vector space model (VSM)

Vector space model, introduced by Salton and al. [6], has been widely used for representing text documents as vectors which contain terms weights. Given a collection D of documents, an index $T = \{t_1, t_2, \dots, t_{|T|}\}$, where $|T|$ denotes the cardinal of T , gives the list of terms (or features) encountered in the documents of D . A document d_i of D is represented by a vector $\mathbf{d}_i = (w_{i,1}, w_{i,2}, \dots, w_{i,|T|})$ where $w_{i,j}$ is the weight of the term t_j in the document d_i . In order to calculate this weight, TF.IDF formula can be used.

2.2 TF: term representativeness

TF (Term Frequency), the relative frequency of term t_j in a document d_i , is defined by:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_l n_{i,l}}$$

where $n_{i,j}$ is the number of occurrences of t_j in document d_i normalized by the number of terms in document d_i . The more frequent the term t_j in document d_i , the higher is the $tf_{i,j}$.

2.3 IDF: discriminatory power of a term

IDF (Inverse Document Frequency) measures the discriminatory power of the term t_j . It is defined by:

$$idf_j = \log \frac{|D|}{|\{d_i : t_j \in d_i\}|}$$

where $|D|$ is the total number of documents in the corpus and $|\{d_i : t_j \in d_i\}|$ is the number of documents in which the term t_j occurs at least one time. The less frequent the term t_j in the collection of documents, the higher is the idf_j .

The weight $w_{i,j}$ of a term t_j in a document d_i is then obtained by combining the two previous criteria:

$$w_{i,j} = tf_{i,j} \times idf_j$$

The more frequent the term t_j is in document d_i and the less frequent it is in the other documents, the higher is the weight $w_{i,j}$.

3 Criteria for features selection

3.1 Category Coverage criteria (CC)

In the context of text categorization, the number of terms belonging to the index can be exceedingly large and all these terms are not necessarily discriminant features of the categories. It is the reason why it can be useful to select a subset of T giving a more representative description of the documents belonging to each category. For this purpose, we proposed in a previous work a selection features criteria, called coverage criteria CC and based on the frequency of the documents containing the term [3]. In observing that the higher the number of documents of category c_k containing t_j , the higher is f_j^k , CC is defined by:

$$CC_j^k = \frac{df_j^k}{|c_k|} * \frac{f_j^k}{\sum_k f_j^k}$$

$$CC_j^k = \frac{(f_j^k)^2}{\sum_k f_j^k}$$

where df_j^k designed the number of documents in the category c_k in which the term t_j appears, and f_j^k , the frequency of documents belonging to c_k and including t_j :

$$df_j^k = |\{d_i \in c_k : t_j \in d_i\}|, k \in \{1, \dots, r\} \quad (1)$$

$$f_j^k = \frac{df_j^k}{|c_k|} \quad (2)$$

If the value of CC_j^k is high, then t_j is a characteristic feature of the category c_k .

3.2 Difference Category Coverage criteria (CCD)

The previous criteria considers only the coverage of the category by one term but it does not take into account the coverage of the other categories. The difference of category coverage permits to overcome this drawback. Thus, the Category Coverage Difference *CCD* is defined by :

$$CCD_j^k = (CC_j^k - CC_j^{\bar{k}})$$

where

$$CC_j^{\bar{k}} = \frac{(f_j^{\bar{k}})^2}{f_j^k + f_j^{\bar{k}}}$$

with

$$f_j^{\bar{k}} = \frac{df_j^{\bar{k}}}{|D| - |c_k|}$$

and

$$df_j^{\bar{k}} = |\{d_i \in D \wedge d_i \notin c_k : t_j \in d_i\}|, k \in \{1, \dots, r\}$$

As previously, if the value of CCD_j^k is high, then t_j is a characteristic feature of the category c_k .

The *CC* and *CCD* criteria can be used for multi-label text categorization by binary classifier. For each category and consequently each classifier, they permit to reduce the index to the set of words which are the most characteristic of this category.

4 Thresholding strategies

When a multi-label algorithm is used in multi-label text categorization, one score $\phi(\mathbf{d}_i, c_k)$ is produced by the classifier for each document-category pair (d_i, c_k) . Given these scores, the problem consists to determine the set of classes $L(d_i)$ which must be attributed to each document d_i .

To solve this problem, different approaches have been proposed, which consist in applying a threshold to the scores returned by the classifier. In RCut method, given a document d_i , the scores $(\phi(\mathbf{d}_i, c_k), k = 1, \dots, r)$ are ranked and the t top ranked classes are assigned to d_i . The value of the parameter t can be either specified by the user or learned using a training set. In PCut method, given a category c_k , the scores $(\phi(\mathbf{d}_i, c_k), i = 1, \dots, |D|)$ are ranked and the n_k top ranked documents are assigned to the class with:

$$n_k = P(c_k) * x * r$$

where $P(c_k)$ is the prior probability for a document to belong to c_k , r , is the number of categories, and x is a parameter which must be estimated using an evaluation set. A review of these methods can be found in [8, 5]

In PCut as well as in RCut, the performance of the classifier depends on the

value of the parameter (t or x). The main advantage of the thresholding method proposed in this article is that the threshold is automatically fixed.

This method, called M-Cut (for Maximum Cut) is based on the following principle, explained graphically. Given a document d_i , the scores $(\phi(\mathbf{d}_i, c_k), k = 1, \dots, r)$ are ranked in decreasing order. The sorted list obtained is noted $S = (s(l), l = 1, \dots, r)$ where $s(l) = \phi(\mathbf{d}_i, c_k)$ if $\phi(\mathbf{d}_i, c_k)$ is the l th highest value in S .

Then, a graph of the scores in their decreasing order is drawn (*i.e.* $s(l), l = 1, \dots, r$ in function of l). The value t retained as threshold is the middle of the maximum gap for S . So, the clusters assigned to d_i are those corresponding to a score $\phi(\mathbf{d}_i, c_k)$ higher than $t : L(d_i) = \{c_k \in C / \phi(\mathbf{d}_i, c_k) > t\}$.

This approach is compared with the RCut strategy in figure 1 for 2 documents, d_1 on the left and d_2 on the right. In $RCut_1$ (resp. $RCut_2$), the t parameter is set up to 1 (resp. 2). For the document d_1 , the same set of classes is affected by $RCut_1$ and $MCut$, while $RCut$ assigns one class more. In the second case, the set of affected classes is different. while d_2 belongs to 1 class (resp. 2 classes) with $RCut_1$ (resp. $RCut_2$) strategy, it is associated to 3 classes with $MCut$ strategy.

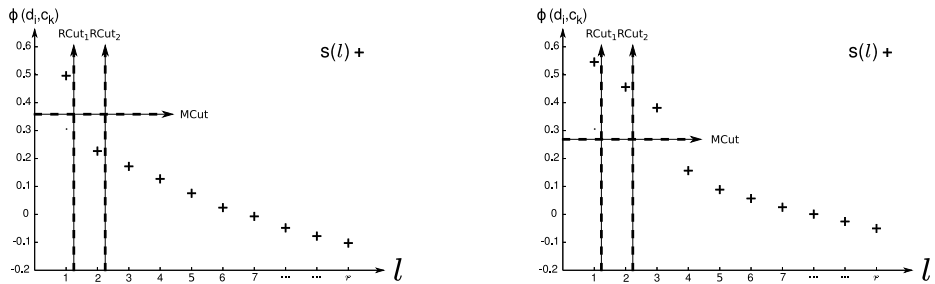


Fig. 1. Illustration comparing RCut and MCut thresholding strategies.

5 Experiments

5.1 Collection INEX XML Mining

The XML Mining collection is composed of about 54 889 XML documents of the Wikipedia XML Corpus. This subset of Wikipedia represents 39 categories, each corresponding to one subject or topic. This year, the collection is multi-label and each document belongs to at least one category. In the XML Mining Track, the training set is composed of 20% of the collection which corresponds to 11 028 documents. On the training set, the mean of the number of category by document is 1.46 and 9 809 documents belong to only one category.

5.2 Pre-processing and categorization

The first step of the categorization approach that we propose, consists in a pre-processing of the collection. It begins by the construction of the list all the terms (or features) encountered in the documents of the collection. This index of 1 136 737 terms is built with the LEMUR software². The Porter Algorithm [4] has also been applied in order to reduce different forms of a word to a common form. It still remains a large number of irrelevant terms that could degrade the categorization, e.g.: numbers (7277, -1224, 0d254c, etc.), terms with less than three characters, terms that appear less than three times, or terms that appear in almost all the documents of the training set corpus. After their deletion, the index size is reduced to 295 721 terms on all the documents and it will be noted T . Depending on the category c_k , T_k will correspond to the index only composed of terms that appear in documents of the category c_k . In order to reduce the index size, we also define $T_{k_{1000}}$ as the index of the category c_k composed of the most characteristic terms of category c_k according to the CCD criteria introduced in section 3. If CCD_{1000}^k corresponds to the best thousandth score obtained with the CCD criteria for the category c_k , $T_{k_{1000}}$ is composed of all terms t_j for which CCD_j^k is higher than CCD_{1000}^k . All the indexes definitions are summarized in the table 1.

Index	Definition
T	$= \{t_j \in d_i d_i \in D\}$
T_k	$= \{t_j \in d_i d_i \in D \wedge d_i \in c_k\}$
$T_{k_{1000}}$	$= \{t_j \in T_k \wedge CCD_j^k \geq CCD_{1000}^k\}$

Table 1. Summary of all defined indexes.

The second step is the categorization step itself. The Support Vector Machines (SVM) classifiers are used for the categorization. SVM was introduced by Vapnik for solving two classes pattern recognition problems using Structural Risk Minimization principal[7]. In our experiments, the SVM algorithm available in the Liblinear library³ has been used.

In the XML Mining Track, the final score ($score(d_i, c_k)$) assigned to a document d_i for the category c_k has to be included in $[0, 1]$ and has to be higher than 0.5 if this document belongs to the category c_k . When the SVM is used as a multi-label classifier (noted *multi-label*), it provides a score $\phi(\mathbf{d}_i, c_k)$ for each pair document - category (d_i, c_k). In that case, the final score $score(d_i, c_k)$ associated to d_i and c_k corresponds to $\phi(\mathbf{d}_i, c_k)$ normalized. So, the final result, given d_i is a set of classes ordered by relevancy. When the SVM is used as a binary classifier (noted *uni-label*), it provides, given a category c_k , two scores ($\phi_k(\mathbf{d}_i, c_k)$ and $\phi_k(\mathbf{d}_i, \bar{c}_k)$). In that case, the score $score(d_i, c_k)$ equals 1

² Lemur is available at the URL <http://www.lemurproject.org>

³ <http://www.csie.ntu.edu.tw/~cjlin/liblinear/> - L2 loss support vector machine primal

if $\phi_k(\mathbf{d}_i, c_k) > \phi_k(\mathbf{d}_i, \bar{c}_k)$ and equals 0 otherwise for a document d_i . The final result is a set of unordered classes.

5.3 Submitted runs

Run	SVM	Index	Thresholding Strategy	Set of classes
lahc_1_baseline	<i>multi – label</i>	T	-	<i>singleton</i>
lahc_2_binary	<i>uni – label</i>	T	-	<i>unordered</i>
lahc_3_binary_1k	<i>uni – label</i>	T_k	-	<i>unordered</i>
lahc_4_binary_1k_1000	<i>uni – label</i>	$T_{k_{1000}}$	-	<i>unordered</i>
lahc_5_max	<i>multi – label</i>	T	<i>max</i>	<i>ordered</i>
lahc_6_pcut	<i>multi – label</i>	T	<i>pcut</i>	<i>ordered</i>
lahc_7_rcut_1	<i>multi – label</i>	T	<i>rcut₁</i>	<i>ordered</i>
lahc_8_rcut_2	<i>multi – label</i>	T	<i>rcut₂</i>	<i>ordered</i>

Table 2. Summary of our XML Mining experiments

In the context of multi-label text categorization, our aim was to evaluate on one hand the influence of the features selection on the performance of the binary classifier (runs *lahc_2*, *lahc_3*, *lahc_4*) and on the other hand the impact of the thresholding strategies on the multi-label classifier (runs *lahc_5*, *lahc_6*, *lahc_7*, *lahc_8*). We have submitted 8 runs based on different indexes and thresholding strategies, summarized in table 2. Given the SVM score $\phi(\mathbf{d}_i, c_k)$, the first 4 runs uses the *unordered* method to compute the final score $score(d_i, c_k)$ and the last 4 runs the *ordered* one.

The first run (*lahc_1*) corresponds to the baseline. This run only assigns one category for each document. This category corresponds to the highest score provided by the multi-label SVM classifier (*multi – label*).

In order to evaluate the influence of the selection features on the performances, the three next runs consider the SVM as a binary classifier (*uni – label*) employing different indexes. The index T (resp. T_k , $T_{k_{1000}}$) is tested with the second run (*lahc_2*) (resp. *lahc_3*, *lahc_4*). The binary classifier can assign no category to a document. In that case, for *lahc_3* and *lahc_4* runs, the category c_k provided by the baseline run (*lahc_1*) is affected to the document.

The last four runs exploit the different thresholding strategies detailed in section 4. The fifth run (*lahc_5*) uses the *MCut* (resp. *PCut*, *RCut₁* and *RCut₂*) strategy. The run *lahc_6* exploits the PCut strategy with x equals to the number of documents in the test set divided by the number of categories. The run *lahc_7* (resp. *lahc_8*) applies the RCut strategy using a parameter t fixed to 1 (resp. 2).

Participant	Run	Macro ACC	Micro ACC	Macro ROC	Micro ROC	Macro PRF	Micro PRF	MAP	Mean
lhc	lahc_5_max	0,968	0,952	0,936	0,934	0,549	0,578	0,788	0,820
lhc	lahc_7_rcut_1	0,974	0,962	0,938	0,935	0,531	0,564	0,788	0,817
lhc	lahc_6_pcut	0,973	0,961	0,927	0,925	0,548	0,563	0,748	0,816
lhc	lahc_8_rcut_2	0,959	0,933	0,903	0,906	0,515	0,528	0,788	0,791
xerox	nxQ.3.merge.tfidf	0,975	0,964	0,753	0,767	0,579	0,605	0,678	0,774
xerox	netxQ.4.plus.tfidf	0,974	0,963	0,748	0,765	0,571	0,600	0,679	0,770
xerox	nxQ.4.merge	0,974	0,963	0,748	0,765	0,571	0,600	0,679	0,770
peking	3	0,963	0,948	0,842	0,850	0,480	0,519	0,702	0,767
peking	2	0,963	0,948	0,842	0,850	0,480	0,518	0,702	0,767
peking	1	0,962	0,947	0,842	0,850	0,478	0,516	0,702	0,766
granada	nb_with_links_sub	0,952	0,934	0,802	0,820	0,500	0,530	0,642	0,756
granada	nb_sub	0,951	0,933	0,803	0,820	0,496	0,527	0,641	0,755
lhc	lahc_1_baseline	0,974	0,962	0,721	0,743	0,531	0,564	0,685	0,749
granada	orgate_with_links_sub	0,848	0,819	0,928	0,927	0,316	0,360	0,725	0,700
lhc	lahc_3_binary_1k	0,967	0,950	0,619	0,629	0,334	0,355	0,407	0,642
granada	orgate_sub	0,754	0,678	0,925	0,922	0,253	0,263	0,730	0,632
lhc	lahc_2_binary	0,971	0,958	0,600	0,613	0,289	0,323	0,393	0,626
lhc	lahc_4_binary_1k_1000	0,965	0,947	0,585	0,596	0,252	0,279	0,330	0,604
wollongon	bpts2.fl.r3	0,913	0,892	0,625	0,619	0,192	0,218	0,138	0,576
wollongon	bptsext.fl.a.r3	0,131	0,160	0,558	0,561	0,072	0,103	0,100	0,264
wollongon	bptsext.fl.r3	0,038	0,055	0,632	0,623	0,071	0,102	0,208	0,253
wollongon	bpts2.fl.a.r3	0,038	0,055	0,598	0,599	0,071	0,102	0,125	0,244
wollongon	bptsext.map.r3	0,137	0,141	0,506	0,513	0,065	0,096	0,192	0,243
wollongon	bpts2.map.r3	0,115	0,123	0,511	0,510	0,070	0,101	0,129	0,238

Table 3. Summary of all XML Mining results

6 Experimental results

All the results are summarized in table 3. We will firstly discuss results of our baseline. Secondly we will detail the results concerning the selection features criteria and finally those which exploit a thresholding strategy.

Baseline results (run: 1). On table 3, our baseline results are quite good if we compare them to other participant results. As this run limits the number of affectations, it also reduces the number of errors and that is why this is our best run for the ACC criteria. As we only consider a binary score (*unordered*), ROC and MAP criteria are not very good since they take into account the order of returned categories. The PRF criteria, that combines precision and recall, is not very high. That means that we should have a correct precision, but a very low recall because this run considers only one category by document.

Features selection runs (run: 2, 3, 4). All the runs, that use thresholding strategies, permit globally to improve the baseline results. Runs 2, 3 and 4 aim to evaluate the influence of the index size using different binary classifiers for each category. As we can see on table 3, all these runs are worse than our baseline for all evaluation criteria. On average on the different evaluation criteria, run 3 is better than runs 2 and 4.

We can conclude that the index reduction is not satisfying and has to be improved. The first idea is to come back to the strategy proposed in INEX 2008 and which consists to define a global index by union of the categories' indexes $\cup_{k \in C} T_{k_{1000}}$. The second idea is to use a different number of terms depending on the category.

Thresholding strategy runs (run: 5, 6, 7, 8). The accuracy criteria (ACC) is in favour of runs which limit the number of affectations. Indeed, if we use a model that assigns no category to the documents, it will obtain a Macro ACC of 0,963. It is the reason why, our baseline run (run 1) and the *rcut*₁ strategy, which affect only one category, obtain the best accuracy over all our runs. In run 5, 6 and 8 several categories can be assigned to one document, and for this reason, we observe a decrease of the accuracy. The run 8 is globally worse than the others because two classes are systematically affected to each document while the average number of categories by document, estimated on the training set, is around 1.46.

On average, run 5 is the best of our runs. It is the best for the PRF criteria and it provides the same results as runs 7 and 8 for the MAP criteria since there is only the thresholding strategy that changes. Concerning the ROC criteria, it is slightly worse (Macro: 0.936, Micro: 0.934) than the run 7 (Macro: 0.938, Micro: 0.935).

7 Conclusion

In this article, we focused on the selection of the set of classes that will label a document for the multi-label text categorization. We propose a thresholding strategy, called *MCut*. The results obtained on the Inex XML Mining collection

are encouraging. This method is compared to the commonly used approaches *RCut* and *PCut*. *Rcut*₁ and *RCut*₂ give also quite good results but they have the drawback to impose a predefined number of categories by document. Contrary to *RCut*, the number of categories for each document could be different with the *PCut* strategy. However, this method is not suitable if we want to know the category of a new single document. So, *MCut* seems to be a good choice because it does not make hypothesis on categories distributions and does not impose the number of category per document.

References

1. L. Denoyer and P. Gallinari. Report on the xml mining track at inex 2007 categorization and clustering of xml documents. *SIGIR Forum*, 42(1):22–28, 2008.
2. Ludovic Denoyer and Patrick Gallinari. Overview of the inex 2008 xml mining track. In *Advances in Focused Retrieval, Proceedings of Initiative for the Evaluation of XML Retrieval, Lecture Notes in Computer Science*, volume 5631, pages 401–411, 2009.
3. Mathias Géry, Christine Langeron, and Christophe Moulin. Ujm at inex 2008 xml mining track. In *Advances in Focused Retrieval, Proceedings of the International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX2008*, volume 5631 of *Lecture Notes in Computer Science*, pages 446–452, 2009.
4. M. F. Porter. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316, 1997.
5. Arturo Montejo Ráez and Luis Alfonso Ureña López. Selection strategies for multi-label text categorization. In *Advances in Natural Language Processing, Proceedings of International Conference on NLP, FinTAL*, volume 4139 of *Lecture Notes in Computer Science*, pages 585–592, 2006.
6. G. Salton and M.J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1983.
7. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.
8. Yiming Yang. A study of thresholding strategies for text categorization. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 137–145, 2001.

BUAP: Performance of K-Star at the INEX'09 Clustering Task

David Pinto¹, Mireya Tovar¹, Darnes Vilariño¹, Beatriz Beltrán¹,
Héctor Jiménez-Salazar², Basilia Campos¹
{dpinto, mtovar, darnes, bbeltran}@cs.buap.mx, hgimenezs@gmail.com

¹Faculty of Computer Science
B. Autonomous University of Puebla, Mexico

²Department of Information Technologies
Autonomous Metropolitan University, Mexico

Abstract. The aim of this paper is to use unsupervised classification techniques in order to group the documents of a given huge collection into clusters. We approached this challenge by using a simple iterative clustering algorithm (K-Star) in a recursive clustering process over subsets of the complete collection.

The obtained results are good with respect to different baselines presented in the INEX 2009 clustering task.

1 Introduction

The INEX 2009 clustering task was presented with the purpose of being an evaluation forum for providing a platform to measure the performance of clustering methods over a real-world and high-volume Wikipedia collection.

Clustering analysis refers to the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait, often proximity, according to some defined distance measure [1,2,3].

Clustering methods are usually classified with respect to their underlying algorithmic approaches: hierarchical, iterative (or partitional) and density-based are some possible categories belonging to this taxonomy. In Figure 1 we can see the taxonomy presented in [4].

Hierarchical algorithms find successive clusters using previously established ones, whereas partitional algorithms determine all clusters at once. Hierarchical algorithms can be agglomerative (“bottom-up”) or divisive (“top-down”); agglomerative algorithms begin with each element as a separate cluster and merge the obtained clusters into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. Iterative algorithms start with some initial clusters (their number either being unknown in advance or given a priori) and intend to successively improve the existing cluster set by changing their “representatives” (“centers of gravity” or “centroids”), like in *K*-Means [3] or by iterative node-exchanging (like in [5]).

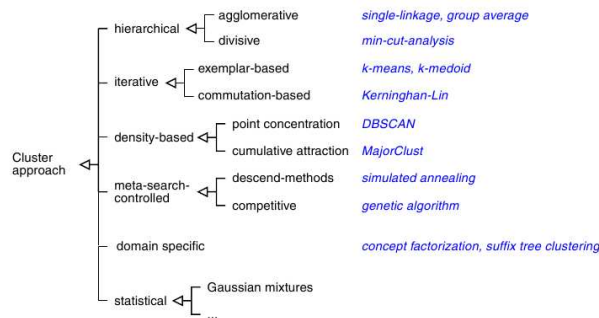


Fig. 1. A taxonomy of clustering methods as presented in [4] (Reproduced with permission of the author).

In this paper we report the obtained results when the iterative K-Star method was applied to the INEX2009.SUBSET collection. Therefore, the complete description of this clustering method is given in the following section.

2 The K-Star clustering method

K-Star [9] is a clustering method which automatically reveals the number of clusters, unknown in advance. As the most of the clustering methods, it requires a similarity matrix of the documents to be clustered (corpus). The algorithm follows as shown in Figure 2.

1. It looks for the maximum value in the similarity matrix $\varphi(d_i, d_j)$, and constructs a cluster (C_i) made up of the two documents this similarity value refers to. It marks these documents (d_i and d_j) as assigned.
2. For each unassigned document (d_k)
 - If $\varphi(d_k, d_i) > \tau$, where τ is a given threshold, then add d_k to cluster C_i and mark d_k as assigned.
3. Return to Step 1

Fig. 2. The K-Star Clustering Method

In this work, we have used a canonic threshold τ defined as the average of the values in the similarity matrix which were calculated as described in the following section.

3 The similarity matrix

We assume that the complete document clustering task may be carried out by executing at least the following three steps: (1) document representation; (2) calculus of a similarity matrix which represents the similarity degree among all the documents of the collection; and (3) clustering of the documents. In particular, the construction of the similarity matrix was carried out by means of the TF-IDF measure which is described into detail as follows.

The Term Frequency and Inverse Document Frequency (*tf-idf*) is a statistical measure of weight often used in natural language processing to measure how important a word is to a document in a corpus, using a vectorial representation. The importance of each word increases proportionally to the number of times a word appears in the document (frequency) but is offset by the frequency of the word in the corpus. In this document, we will refer to the *tf-idf* as the complete similarity process of using the *tf-idf* weight and a special similarity measure proposed by Salton [8] for the Vector Space Model, which is based on the use of the cosine among vectors representing the documents.

The *tf* component of the formula is calculated by the normalized frequency of the term, whereas the *idf* is obtained by dividing the number of documents in the corpus by the number of documents which contain the term, and then taking the logarithm of that quotient. Given a corpus D and a document d_j ($d_j \in D$), the *tf-idf* value for a term t_i in d_j is obtained by the product between the normalized frequency of the term t_i in the document d_j (tf_{ij}) and the inverse document frequency of the term in the corpus ($idf(t_i)$) as follows:

$$tf_{ij} = \frac{tf(t_i, d_j)}{\sum_{k=1}^{|d_j|} tf(t_k, d_j)} \quad (1)$$

$$idf(t_i) = \log \left(\frac{|D|}{|d : t_i \in d, d \in D|} \right) \quad (2)$$

$$tf-idf = tf_{ij} * idf(t_i) \quad (3)$$

Each document can be represented by a vector where each entry corresponds to the *tf-idf* value obtained by each vocabulary term of the given document. Thus, given two documents in vectorial representation, d_i and d_j , it is possible to calculate the cosine of the angle between these two vectors as follows:

$$\text{Cos}_\theta(\vec{d}_i, \vec{d}_j) = \frac{\vec{d}_i \cdot \vec{d}_j}{\|\vec{d}_i\| \|\vec{d}_j\|}$$

The complete description of the implemented approach is given in the following section.

4 Description of the approach

In order to be able to cluster high volumes of data, we have approached a simple clustering technique based on partitioning of the complete document collection. The process followed is presented in Figure 3, whereas a scheme of the same process is given in Figure 4.

1. Eliminate all those terms whose frequency is lower than 3
2. Represent each document according to TF-IDF
3. Let D be the complete document collection
4. While $Loop \leq MAX_ITERATIONS$
 - Split D into subsets D_i made of 100 documents
 - For each subset D_i
 - Calculate de similarity matrix M_i of D_i using the cosine measure
 - Apply the K-Star clustering method to M_i in order to discover k clusters ($C_{Loop,k}$)
 - End For
 - If ($Loop > 1$)
 - Let $C_{Final,j} = C_{Loop,j} \cup C_{(Loop-1),j'}$, where $C_{Loop,j} \cap C_{(Loop-1),j'} \geq 1$
 - End If
 - Select a representative document R_j for each cluster $C_{Loop,j}$ obtained
 - Let D be the set of documents R_j , i.e., only those that represent each cluster obtained
 - $Loop = Loop + 1$
5. End While
6. Output the set of clusters C_{Final}

Fig. 3. Algorithm used for clustering the INEX2009_SUBSET with K-Star

The obtained results are presented and described in the following section.

5 Experimental results

The clustering task of INEX 2009 evaluated unsupervised machine learning solutions against the ground truth categories by using standard evaluation criteria such as Purity, Entropy and F-score.

Even if the complete description of the dataset used in the clustering task of INEX 2009 will be given in the track overview paper, we may describe general features of this corpus.

The INEX XML Wikipedia collection used in our experiments is a subset of the complet corpus. This subset contains 54,575 documents pre-processed in order to provide various representations of the documents. The aim of this pre-processing was to enable the participation of different teams with minimal

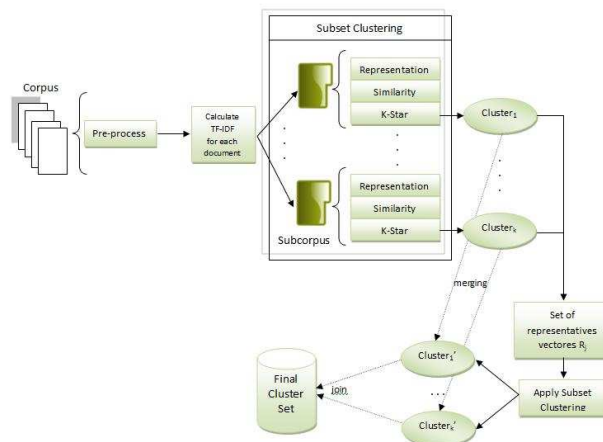


Fig. 4. Two step approach of BUAP Team at the INEX 2009 clustering task

overheads in data-preparation the collection. It was provided, for instance, a bag-of-words representation of terms and frequent phrases in a document, frequencies of various XML structures in the form of tags, trees, links, named entities, etc.

We approached three different representations of data: unigram stems, bigrams and bigram stems which were executed in order to observe the K-Star clustering method performance.

In Tables 1 and 2 we may see the obtained results of the three BUAP approaches with respect to two baselines, one random assignment (random-54575) and one ground truth over 73,944 and 12,803 YAGO categories, respectively. At least one approach shows a high Micro and Macro Purity

Table 1. Purity of the INEX09_SUBSET clustering evaluation using 73,944 YAGO categories (all YAGO categories)

<i>Run ID</i>	<i>Description</i>	<i>Clusters</i>	<i>Micro Purity</i>	<i>Macro Purity</i>
24	random-54575	54575	1.0	1.0
11	ground truth for 73,944 YAGO categories	73944	0.9999	1.0
67	KStar method Bigrams	35569	0.6812	0.8968
65	KStar method Unigram Stems	7019	0.1894	0.6399
66	KStar method Stems Bigrams	6961	0.1883	0.6350

Tables 3 and 4 present the K-Star performance with respect to entropy. In general the three approaches show low entropy value.

Table 2. Purity of the INEX09_SUBSET clustering evaluation using 12,803 YAGO categories (containing ≥ 5 documents)

<i>Run ID</i>	<i>Description</i>	<i>Clusters</i>	<i>Micro Purity</i>	<i>Macro Purity</i>
12	ground truth for 12,804 YAGO categories	12804	1.0	1.0
24	random-54575	54575	1.0	1.0
67	KStar method Bigrams	35569	0.6964	0.9006
65	KStar method Unigram Stems	7019	0.2076	0.6410
66	KStar method Stems Bigrams	6961	0.2074	0.6407

Table 3. Entropy of the INEX09_SUBSET clustering evaluation using 73,944 YAGO categories (all YAGO categories)

<i>Run ID</i>	<i>Description</i>	<i>Clusters</i>	<i>Micro Entropy</i>	<i>Macro Entropy</i>
66	KStar method Stems Bigrams	6961	0.1883	0.6350
65	KStar method Unigram Stems	7019	0.1894	0.6399
11	ground truth for 73,944 YAGO categories	73944	0.9999	1.0
67	KStar method Bigrams	35569	0.6812	0.8968
24	random-54575	54575	1.0	1.0

Table 4. Entropy of the INEX09_SUBSET clustering evaluation using 12,803 YAGO categories (containing ≥ 5 documents)

<i>Run ID</i>	<i>Description</i>	<i>Clusters</i>	<i>Micro Entropy</i>	<i>Macro Entropy</i>
66	KStar method Stems Bigrams	6961	0.2074	0.6407
65	KStar method Unigram Stems	7019	0.2076	0.6410
12	ground truth for 12,804 YAGO categories	12804	1.0	1.0
67	KStar method Bigrams	35569	0.6964	0.9006
24	random-54575	54575	1.0	1.0

6 Conclusions

In this paper a recursive method based on the iterative K-Star clustering method was proposed. The aim was to allow high scalability of the clustering algorithm. The obtained results are preliminary shown in this paper, but they must be analysed into detail and compared with the runs submitted by other teams in order to validate the performance of K-Star.

We observed, however, that the implemented approach is easy of being implemented and obtained good results in the INEX 2009 clustering task.

7 Acknowledgments

This project has been partially supported by CONACYT.

References

1. MacKay, D.J.C.: Information Theory, Inference and Learning Algorithms. Cambridge University Press (2003)
2. Mirkin, B.G.: Mathematical Classification and Clustering. Springer (1996)
3. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press (1967) 281–297
4. Meyer zu Eissen, S.: On information need and categorizing search. PhD dissertation, University of Paderborn, Germany (2007)
5. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Systems Technical Journal **49**(2) (1970) 291–308
6. Stein, B., Nigemman, O.: On the nature of structure and its identification. In: Proc. of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science. Volume 1665 of Lecture Notes in Computer Science., Springer-Verlag (1999) 122–134
7. Manning, D.C., Schütze, H.: Foundations of statistical natural language processing. MIT Press (1999)
8. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Communications of the ACM **18**(11) (1975) 613–620
9. Shin, K., Han, S.Y.: Fast clustering algorithm for information organization. In: Proc. of the CICLing 2003 Conference. Volume 2588 of Lecture Notes in Computer Science., Springer-Verlag (2003) 619–622

Link-based text classification using Bayesian networks

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

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

Abstract. In this paper we propose a new methodology for link-based document classification based on probabilistic classifiers and Bayesian networks. We also report the results obtained of its application to the XML Document Mining Track of INEX'09.

1 Introduction

1.1 Our participation

This is the third year that researchers from the University of Granada (specifically from the Uncertainty Treatment in Artificial Intelligence research group) participate on the XML Document Mining track of the INEX workshop. As in previous editions, we restrict our solutions to the application of probabilistic methods to these problems. To be more precise, we are looking to solve to the problem of link-based document classification within the field of Bayesian networks [10] (a special case of Probabilistic Graphical Models).

1.2 Problem overview

This year, the proposed problem is rather similar to the one considered in the previous edition of the workshop [3]. A training corpus, composed of labeled XML files is provided, and an unlabeled test corpus is left to the participants, in order to be estimated its labeling. Also, a link file is shown, which gives specific relations among documents (which may be in the training corpus or not). Thus, the problem can be seen as a graph labeling problem, where each node has textual (XML) content.

The main difference between this INEX track in 2008 and 2009 is the fact that the training corpus is made of multilabeled documents, that is to say, a document can belong to one or more categories. The rest of the rules are essentially the same, although the document collection and the set of categories are also different.

As we did before, we can assume that the XML markup (the “internal structure” of the collection) is not very helpful for categorization. In fact, we did

not find it very useful for the task in previous editions [5] (by making several transformations from XML to flat text documents). Moreover, the organizers have provided an indexed file of term vectors representing the documents, where XML marks have been removed.

Like our previous participation [6], we will use explicitly the “external structure” of the collection, i.e. the link file (the graph among documents). Then, we provided a “graphical proof” that the category of the documents linked by one tend to be similar to the category of the own document. Several experiments in the same direction showed us the same fact for the 2009 corpus, although we do not reproduce them here. Apart from those experiments, the names of the categories (which are explicitly given in the training set), tend to show categories which are probably coming from a hierarchy (for example `Portal:Religion`, `Portal:Christianity` and `Portal:Catholicism`). The two known facts about the relations are summarized here:

- In this linked corpus, due to its nature, a “hyperlink regularity” is supposed to arise (see [15] for more details).
- There is some categories strongly related a priori, because the probable existence of a (unknown) hierarchy.

Although last year we proposed a method that captures some “fixed” relations among categories, given this different problem setting and its higher dimensionality, this year we pretend to learn those relations automatically from data, leading to a more flexible approach.

2 Base classifiers

Two base classifiers will be used to label the graph nodes based only on their content. These two will be the baseline, and will be combined with the Bayesian network learnt from data with our new methodology. We will briefly describe them, in order to make the paper more readable.

Both of the classifiers are probabilistic, i.e. given a document d_j , they compute the values $p(c_i|d_j)$ for each category, and assign it as a degree of confidence. The advantage of probability is that it is very well founded, and several different probabilistic approaches can be combined together, because they are dealing with the same measures.

Note that here we solve here the multilabel problem defining a binary classifier for each label, following the more “classical” approach to this task [12].

2.1 Multinomial naive Bayes

The model is the same used by McCallum et al. [9], adapting it to the case of many binary problems. The naive Bayes, in its multinomial version is a very fast and well performing method. In this model, we firstly assume that the length of the document is independent of the category. We also assume that the term

occurrences are independent one each other, given the category (this is the core of the naive Bayes method).

In the multinomial version of this classifier, we see a document d_j as being drawn from a multinomial distribution of words with as many independent trials as the length $|d_j|$ of d_j .

So, given a category c_i , we express the probability¹ $p_i(c_i|d_j)$ as

$$p_i(c_i|d_j) = \frac{p_i(d_j|c_i) p_i(c_i)}{p_i(d_j)}. \quad (1)$$

We can rewrite $p(d_j)$ using the law of total probability,

$$p_i(d_j) = p_i(d_j|c_i) p_i(c_i) + p_i(d_j|\bar{c}_i) (1 - p_i(c_i)). \quad (2)$$

The values $p_i(c_i|d_j)$ can be easily computed in terms of the prior probability $p_i(c_i)$ and the probabilities $p_i(d_j|c_i)$ and $p_i(d_j|\bar{c}_i)$.

Besides, prior probabilities are estimated from document counts:

$$\hat{p}_i(c_i) = \frac{N_{i,doc}}{N_{doc}} \quad (3)$$

where N_{doc} is the number of documents in the training set and $N_{i,doc}$ is the number of documents in the training set which belong to category c_i .

On the other hand, we can estimate $p_i(d_j|c_i)$ and $p_i(d_j|\bar{c}_i)$ as follows (as a multinomial distribution over the words):

$$p_i(d_j|c_i) = p_i(|d_j|) \frac{|d_j|!}{\prod_{t_k \in d_j} n_{jk}!} \prod_{t_k \in d_j} p_i(t_k|c_i)^{n_{jk}},$$

and

$$p_i(d_j|\bar{c}_i) = p_i(|d_j|) \frac{|d_j|!}{\prod_{t_k \in d_j} n_{jk}!} \prod_{t_k \in d_j} p_i(t_k|\bar{c}_i)^{n_{jk}},$$

where n_{jk} is the frequency of the term t_k in the document d_j .

Substituting and simplifying in equations 1 and 2 we obtain:

$$p_i(c_i|d_j) = \frac{p_i(c_i) \prod_{t_k \in d_j} p_i(t_k|c_i)^{n_{jk}}}{p_i(c_i) \prod_{t_k \in d_j} p_i(t_k|c_i)^{n_{jk}} + (1 - p_i(c_i)) \prod_{t_k \in d_j} p_i(t_k|\bar{c}_i)^{n_{jk}}}.$$

¹ With the notation $p_i(c_i|d_j)$ we are emphasizing that the probability distribution is computed over a binary variable C_i , with values in $\{c_i, \bar{c}_i\}$. So, we have a different probability distribution over each category.

And finally, individual term probabilities $p_i(t_k|c_i)$ and $p_i(t_k|\bar{c}_i)$ are given by the following formulae (using Laplace smoothing):

$$\hat{p}_i(t_k|c_i) = \frac{N_{ik} + 1}{N_{i\bullet} + M}, \quad \hat{p}_i(t_k|\bar{c}_i) = \frac{N_{\bullet k} - N_{ik} + 1}{N - N_{i\bullet} + M}, \quad (4)$$

where N_{ik} is the number of times the term t_k appears in documents of class c_i , $N_{i\bullet}$ is the total number of words in documents of class c_i ($N_{i\bullet} = \sum_{t_k} N_{ik}$), $N_{\bullet k}$ is the numbers of times that the term t_k appears in the training documents ($N_{\bullet k} = \sum_{c_i} N_{ik}$), N is the total number of words in the training documents, and M is the size of the vocabulary (the number of distinct words in the documents of the training set).

2.2 Bayesian OR gate

The Bayesian OR gate classifier was presented in the INEX 2007 Workshop by this group [5]. This classifier has the assumption that the relationship among the terms and each category follows a so-called *noisy-OR gate* probability distribution. Following the Bayesian networks notation, we express this model as one node for the category C_i (binary variable C_i , ranging in $\{c_i, \bar{c}_i\}$), one node for each term T_k (binary variable T_k , with values in $\{t_k, \bar{t}_k\}$), and arcs going from each term node to the category nodes they appear in (i.e. they are the parents $Pa(C_i)$, of the category node).

In the naive Bayes model (a generative one), we are defining $p(d_j|c_i)$, whereas in the Bayesian OR gate (a discriminative model), we are computing directly $p_i(c_i|d_j)$. Instead of modeling a “general” distribution of probability, $p(c_i|d_j)$ is considered to be following a “canonical model” [10] (the noisy OR gate) which makes computations and parameter storage a feasible task.

We can define the probability distribution for this noisy OR gate in the following way:

$$p_i(c_i|pa(C_i)) = 1 - \prod_{T_k \in R(pa(C_i))} (1 - w(T_k, C_i))$$

$$p_i(\bar{c}_i|pa(C_i)) = 1 - p_i(c_i|pa(C_i)),$$

where $R(pa(C_i)) = \{T_k \in Pa(C_i) | t_k \in pa(C_i)\}$, i.e. $R(pa(C_i))$ is the subset of parents of C_i which are instantiated to its t_k value in the configuration $pa(C_i)$. $w(T_k, C_i)$ is a weight representing the probability that the occurrence of the “cause” T_k alone (T_k being instantiated to t_k and all the other parents T_h instantiated to \bar{t}_h) makes the “effect” true (i.e., forces class c_i to occur).

Then, given a certain document d_j , we can compute the posterior probability $p_i(c_i|d_j)$ instantiating to the value t_k the terms that appear in the document (i.e. $p_i(t_k|d_j) = 1$), and to the value \bar{t}_k those terms that do not appear in d_j (i.e. $p_i(t_h|d_j) = 0$). The results is [4]:

$$\begin{aligned}
p_i(c_i|d_j) &= 1 - \prod_{T_k \in Pa(C_i)} (1 - w(T_k, C_i) p_i(t_k|d_j)) \\
&= \prod_{T_k \in Pa(C_i) \cap d_j} (1 - w(T_k, C_i)).
\end{aligned}$$

Finally, we have to give a definition for the weights $w(T_k, C_i)$, which is almost the same appearing in [5]:

$$w(T_k, C_i) = \frac{N_{ik}}{nt_i N_{\bullet k}} \prod_{h \neq k} \frac{(N_{i\bullet} - N_{ih})N}{(N - N_{\bullet h})N_{i\bullet}}. \quad (5)$$

In this formula, $N_{ik}, N_{\bullet k}, N_{i\bullet}$ and N mean the same than in previous definitions made in the multinomial naive Bayes explanation, and nt_i is the number of different terms occurring in documents of the class C_i . The factor nt_i is introduced here to relax the independence assumption among terms, but some other valid definitions for the weights (which do not use this factor) can be found in [4] and [5].

Finally, in order to make the probabilities independent on the length of the document, and make scores comparable, we introduce the following normalization, which is somewhat similar to the *RCut* thresholding strategy [14], and we return, as the final probability $p(c_i|d_j)$:

$$p(c_i|d_j) = \frac{p_i(c_i|d_j)}{\max_{c_k} \{p_k(c_k|d_j)\}}$$

Some experiments [4] have shown that the Bayesian OR gate classifier tends to outperform the multinomial naive Bayes classifier, although the number of parameters needed and the complexity are more or less the same.

3 The Bayesian network model

This section describes a new methodology that models a link-based categorization environment using Bayesian networks. In this development, we will only use data from incoming links, because we carried several experiments on the corpus, and found them much more informative than outgoing ones. Anyway, information from outgoing links (or even considering undirected links) could also be used in this model.

3.1 Modeling link structure between documents

In this problem, we will consider two binary variables for every category i : one is C_i (with states $\{c_i, \bar{c}_i\}$) which models the probability of a document being (or not) of class C_i , and the variable LC_i , (with states $\{lc_i, \bar{lc}_i\}$, which represents if there is a link, or not, from documents of category i (as we stated before, we also

could represent the existence of outgoing links to a document of category i , or both interactions). We assume there is a global probability distribution among all these variables, and we will model it with a Bayesian network.

To learn a model from the data, we will use the training documents, each one as an instance whose categories (values for variables C_i) are perfectly known, and the links from other documents. If a document is linked by another training document of category j , we will set $LC_j = lc_j$, setting it to \bar{lc}_j otherwise. Note that a training document could be linked by test documents (whose categories are unknown). In that case, all of those test documents are ignored.

So, we could learn a Bayesian network from training data (see next section) and, for each test document d_j , we could compute $p(c_i|e_j)$, where e_j represents all the evidence given by the information of documents that link this.

Thus, the question is the following: for a certain document d_j , given $p(c_i|d_j)$ and $p(c_i|e_j)$, how could we combine them in an easy way? We want to compute the posterior probability $p(c_i|d_j, e_j)$, the probability of a category given the terms composing the document and the evidence due to link information.

Using Bayes' rule, and assuming that the content and the link information are independent given the category, we get:

$$\begin{aligned} p(c_i|d_j, e_j) &= \frac{p(d_j, e_j|c_i) p(c_i)}{p(d_j, e_j)} = \frac{p(d_j|c_i) p(e_j|c_i) p(c_i)}{p(d_j, e_j)} \\ &= \frac{p(c_i|d_j) p(d_j) p(e_j|c_i) p(c_i)}{p(c_i) p(d_j, e_j)} = \frac{p(c_i|d_j) p(d_j) p(c_i|e_j) p(e_j)}{p(c_i) p(d_j, e_j)} \\ &= \left(\frac{p(d_j) p(e_j)}{p(d_j, e_j)} \right) \left(\frac{p(c_i|d_j) p(c_i|e_j)}{p(c_i)} \right). \end{aligned}$$

The first term of the product is a factor which does not depend on the category. So, we can write the probability as:

$$p(c_i|d_j, e_j) \propto \frac{p(c_i|d_j) p(c_i|e_j)}{p(c_i)}$$

And we can rewrite the first expression in this final form:

$$p(c_i|d_j, e_j) = \frac{p(c_i|d_j) p(c_i|e_j) / p(c_i)}{p(c_i|d_j) p(c_i|e_j) / p(c_i) + p(\bar{c}_i|d_j) p(\bar{c}_i|e_j) / p(\bar{c}_i)} \quad (6)$$

We must make some final comments about this equation to make it more clear:

- As we said before, the posterior probability $p(c_i|d_j)$ is the one obtained from a binary probabilistic classifier (one of the two presented before, or any other), which is going to be combined with the information obtained from the link evidence.

- The prior probability used here, $p(c_i)$, is the one computed with propagation over the Bayesian network learnt with link information.
- Because the variables C_i are binary, it is clear that $p(\bar{c}_i|e_j) = 1 - p(c_i|e_j)$, $p(\bar{c}_i) = 1 - p(c_i)$ and $p(\bar{c}_i|d_j) = 1 - p(c_i|d_j)$.

3.2 Learning link structure

Given the previous variable setting, from the training documents, their labels and the link file, we can obtain a training set for the Bayesian network learning problem, composed of vectors of binary variables C_i and LC_i (one for each training document).

We have used WEKA package [13] to learn a generic Bayesian network (not a classifier) using a hill climbing algorithm (with the classical operators of addition, deletion and reversal of arcs) [1], with the BDeu metric [8]. In order to reduce the search space, we have limited the number of parents of each node to a maximum of 3.

After the network has been learnt, we have converted it to the Elvira [7] format. Elvira is a software developed by some Spanish researchers which has implemented many algorithms for Bayesian networks. In this case, we have used it to carry out the inference procedure. This is done as follows:

1. For each test document d_j , we set in the Bayesian network the LC_i variables to lc_i or \bar{lc}_i , depending whether d_j is linked by at least one document of category i , or not, respectively. This is the evidence coming from the links (represented before by e_j).
2. For each category variable, C_i , we compute the posterior probability $p(c_i|e_j)$. This procedure is what is called *evidence propagation*.

Due to the size of the problem, instead of exact inference, we have used an approximate inference algorithm [2], firstly to compute prior probabilities of each category in the network, $p(c_i)$, and secondly, to compute the probabilities of each category given the link evidence e_j , for each document d_j in the test set, $p(c_i|e_j)$. The algorithm used is called Importance Sampling algorithm, and is faster than other exact approaches.

4 Results

Four files were sent to the organization to participate in the Workshop. Two of them, the baselines, were flat-text classifiers (that is to say, no link structure was used to label the documents, only its content). The other two were the Bayesian network model (BN) over those baselines (the link structure was added using equation 6).

4.1 Preliminary results

The results of the models we sent for this track are displayed in Table 4.1, where M means the “macro” version of the measure, and μ means the “micro” one. The performance measures computed are Accuracy (ACC), Area under Roc curve (ROC), F1 measure (PRF) and Mean average precision by document (MAP).

	MACC	μ ACC	MROC	μ ROC	MPRF	μ PRF	MAP
N. Bayes	0.95142	0.93284	0.80260	0.81992	0.49613	0.52670	0.64097
N. Bayes + BN	0.95235	0.93386	0.80209	0.81974	0.50015	0.53029	0.64235
OR gate	0.75420	0.67806	0.92526	0.92163	0.25310	0.26268	0.72955
OR gate + BN	0.84768	0.81891	0.92810	0.92739	0.31611	0.36036	0.72508

Table 1. Preliminary results.

In both cases, the Bayesian network version of the classifier outperforms the “flat” version, though the results on the OR gate are surprisingly poor in ACC and PRF. This fact is due to the nature of the classifier, and to the kind of evaluation: For the OR gate, the fact that $p(c_i|d_j) > 0.5$ holds does not mean necessarily that d_j should be labeled C_i , whereas in the naive Bayes does (this was the criterion used by the evaluation procedure to assign categories to the test documents).

In fact, for the OR gate classifier is not known, a priori, what is the appropriate threshold τ_i that assigns d_j to class C_i if $p(c_i|d_j) > \tau_i$. This is not a major problem to compute, for example, averaged break-even point measures [12], where no hard categorization is needed. In this case, the threshold 0.5 has been adopted, and we need to re-adapt the model to this setting in order to perform well.

In the following section we can see how we estimated a set of thresholds (using only training data) and how we scaled the probability values, in order to match the evaluation criteria, dramatically improving the results.

4.2 Scaled version of the Bayesian OR gate results

To make this version of the OR gate results, we have followed this procedure: using only training data, a classifier has been built (both in its flat and Bayesian network versions), and evaluated using cross validation (with five folds). In each fold, for each category, we have searched for the threshold of probability that gives the higher $F1$ measure per class and, afterwards, all thresholds have been averaged over the set of cross validation folds.

This is what is called in the literature the *Scut* thresholding strategy [14]. Thus, we obtain, for each category a threshold τ_i between 0 and 1 (different for each of the two models). We should then to transform the results to a scale where each category threshold is mapped to 0.5.

So, the probabilities of the or gates are rescaled using a lineal continuous function f_i which verifies $f_i(0) = 0$, $f_i(1) = 1$ and $f_i(\tau_i) = 0.5$. The function is:

$$f_i(x) = \begin{cases} \frac{0.5x}{\tau_i} & \text{if } x \in [0, \tau_i] \\ 1 - \frac{0.5}{1-\tau_i}(1-x) & \text{if } x \in (\tau_i, 1] \end{cases}$$

Then, the new probability values are computed, using the old values $p(c_i|d_j)$, as $\hat{p}(c_i|d_j) = f_i(p(c_i|d_j))$. Once again, we would like to recall that these new results are only “scaled” versions of the old ones, with thresholds being computed only using the training set. The new results are displayed in Table 4.2.

	MAcc	μ ACC	MROC	μ ROC	MPRF	μ PRF	MAP
OR gate	0.92932	0.92612	0.92526	0.92163	0.45966	0.50407	0.72955
OR gate + BN	0.96607	0.95588	0.92810	0.92739	0.51729	0.55116	0.72508

Table 2. Results using thresholds.

Note that, using the scaling procedure, ROC and MAP values remains equal, whereas PRF and ACC, on the contrary, are improved a lot, and gives results which are much more better.

5 Conclusions and future works

Given the previous results, we can state the two following conclusions:

- The use of the Bayesian network structure for links can improve a lot a basic “flat-text” classifier.
- Our results are fairly well situated in a middle-high point among all participants.

The first statement is clear, particularly on the case of the OR gate classifier, where some measures, like micro PRF are improved near 10%. Accuracy is improved 3-4%, while ROC stands more or less equal. Only MAP is slightly decreased (less than 1%). The changes on the naive Bayes classifier are more irrelevant, but they are all positive too.

The second statement can be easily proved watching at the official table of results. Our best model (OR + Bayesian network) performs in a medium position for ACC, slightly better for PRF, fairly well for MAP (where only 4 models beat us) and very well for ROC measures (the third best performing model in each of the two versions of ROC, among all the participants).

The results could probably be improved with the usage of a better probabilistic base classifier. For example, a logistic regression or some probabilistic version of a SVM classifier (like the one proposed by Platt [11]), which are likely to have better results than our base models (although they can be much more inefficient). We expect to carry out more experiments with different basic classifiers for the final version of this paper.

References

1. W.L. Buntine. A guide to the literature on learning probabilistic networks from data, *IEEE Transactions on Knowledge and Data Engineering* 8:195–210, 1996.
2. A. Cano, S. Moral, A. Salmerón. Algorithms for approximate probability propagation in Bayesian networks, in *Advances in Bayesian Networks, Studies in Fuzziness and Soft Computing* Vol. 146, 77-99, Springer-Verlag, 2004.
3. L. Denoyer, P. Gallinari. Overview of the INEX 2008 XML Mining Track, *Lecture Notes in Computer Science* 5631:401–411, 2009.
4. L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, A.E. Romero. OR gate Bayesian networks for text classification: a discriminative alternative approach to multinomial naive Bayes, *Actas del XIV Congreso Español sobre Tecnologías y Lógica Fuzzy*, 385–390, 2008.
5. L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, A.E. Romero. Probabilistic methods for structured document classification at INEX'07, *Lecture Notes in Computer Science* 4862:195–206, 2008.
6. L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, A.E. Romero. Probabilistic methods for link-based classification at INEX'08, *Lecture Notes in Computer Science* 5631:453–459, 2009.
7. Elvira Consortium. Elvira: An environment for probabilistic graphical models, *Proceedings of the First European Workshop on Probabilistic Graphical Models*, 222–230, 2002. Elvira software available at <http://leo.ugr.es/~elvira>.
8. D. Heckerman, D. Geiger, D.M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data, *Machine Learning* 20(3):197–243, 1995.
9. A. McCallum, K. Nigam. A Comparison of event models for Naive Bayes text classification, *AAAI/ICML Workshop on Learning for Text Categorization*, 137–142, AAAI Press, 1998.
10. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan and Kaufmann, 1988.
11. J. Platt. Probabilistic outputs for Support Vector Machines and comparisons to regularized likelihood methods, in *Advances in Large Margin Classifiers*, 61–74, MIT Press, 1999.
12. F. Sebastiani. Machine Learning in automated text categorization, *ACM Computing Surveys* 34:1–47, 2002.
13. I.H. Witten, E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*, Morgan Kaufmann, 2005.
14. Y. Yang. A study of thresholding strategies for text categorization, *Proceedings of the 24th annual international ACM SIGIR conference on Research and Development in Information Retrieval*, 137–145, 2001.
15. Y. Yang, S. Slattery. A study of approaches to hypertext categorization, *Journal of Intelligent Information Systems* 18:219–241, 2002.

Extended VSM for XML Document Classification using Frequent Subtrees

Jianwu Yang¹ and Songlin Wang¹,

¹ Institute of Computer Sci. & Tech. , Peking University,
Beijing 100871, China
{yangjianwu, wangsonglin}@icst.pku.edu.cn

Abstract. Structured link vector model (SLVM) is a representation proposed for modeling XML documents, which was extended from the conventional vector space model (VSM) by incorporating document structures. In this paper, we describe the classification approach for XML documents based on SLVM in the Document Mining Challenge of INEX 2009, where the closed frequent subtrees as structural units are used for content extraction from the XML document and the Chi-square test is used for feature selection.

Keywords: XML Document, Classification, Vector Space Model (VSM), Structured Link Vector Model (SLVM), Frequent Subtree.

1. Introduction

XML is the W3C recommended markup language for semi-structured data. Its structural flexibility makes it an attractive choice for representing data in application domains. With the rapid growth of XML documents, these arise many issues concerning the management of these documents effectively. Even though the tasks of interest are still clustering, classification and retrieval, conventional document analysis tools developed for unstructured documents [1] fail to take the full advantage of the structural properties of XML documents.

To contrast with ordinary unstructured documents, XML documents represent their syntactic structure via (1) the use of XML elements, each marked by a user-specified tag, and (2) the associated schema specified in either DTD or XML Schema format. In addition, XML documents can be cross-linked by adding IDREF attributes to their elements to indicate the linkage. Thus, techniques designed for XML document analysis normally take into account the information embedded in both the element tags as well as their associated contents for better performance.

2. Structured Link Vector Model (SLVM): An Overview

Structured Link Vector Model (SLVM), which forms the basis of this paper, was originally proposed in [2] for representing XML documents. It was extended from the conventional vector space model (VSM) [3] by incorporating document structures (represented as term-by-element matrices), referencing links (extracted based on IDREF attributes), as well as element similarity (represented as an element similarity matrix). The SLVM has been used in the Document Mining Challenge of INEX 2009 [4].

2.1. Basic representation

Vector Space Model (VSM) [3] has long been used to represent unstructured documents as document feature vectors which contain term occurrence statistics. This bag of terms approach assumes that the term occurrences are *independent* of each other.

Definition 2.1 Assume that there are n distinct terms in a given set of documents D . Let doc_x denote the x^{th} document and d_x denote the **document feature vector** such that

$$d_x = [d_{x(1)}, d_{x(2)}, \dots, d_{x(n)}]^T$$
$$d_{x(i)} = TF(w_i, doc_x) IDF(w_i)$$

where $TF(w_i, doc_x)$ is the frequency of the term w_i in doc_x , $IDF(w_i) = \log(|D|/DF(w_i))$ is the inverse document frequency of w_i for discounting the importance of the frequently appearing terms, $|D|$ is the total number of the documents, and $DF(w_i)$ is the number of documents containing the term w_i .

Applying VSM directly to represent XML documents is not desirable as the document syntactic structure tagged by their XML elements will be ignored. For example, VSM considers two documents with an identical term appearing in, say, their “title” fields to be equivalent to the case with the term appearing in the “title” field of one document and in the “author” field of another. As the “author” field is semantically unrelated to the “title” field, the latter case should be considered as a piece of less supportive evidence for the documents to be similar when compared with the former case. Using merely VSM, these two cases cannot be differentiated.

Structured Link Vector Model (SLVM), proposed in [2], can be considered as an extended version of vector space model for representing XML documents. Intuitively speaking, SLVM represents an XML document as an array of VSMs, each being specific to an XML element (specified by the `<element>` tag in a DTD).

Definition 2.2 SLVM represents an XML document doc_x using a **document feature matrix** $\Delta_x \in R^{n \times m}$, given as

$$\Delta_x = [\Delta_{x(1)}, \Delta_{x(2)}, \dots, \Delta_{x(m)}]$$

where m is the number of distinct XML elements, $\Delta_{x(i)} \in R^n$ is the TFIDF feature vector representing the i^{th} XML element (e_i), given as $\Delta_{x(i,j)} = TF(w_j, doc_x \cdot e_i) \cdot IDF(w_j)$ for all $j=1$ to n , and $TF(w_j, doc_x \cdot e_i)$ is the frequency of the term w_j in the element e_i of doc_x .

Definition 2.3 The normalized document feature matrix is defined as

$$\tilde{\Delta}_{x(i,j)} = \Delta_{x(i,j)} / \sum_k \Delta_{x(i,k)}$$

where the factor caused by the varying size of the element content is discounted via normalization.

Example 2.1 Figure 1 shows a simple XML document. Its corresponding document feature vector d_x , document feature matrix Δ_x , and normalized document feature matrix $\tilde{\Delta}_x$ are shown in Figure 2-4 respectively. Here, we assume all the terms share the same *IDF* value equal to one.

```

<article>
  <title>Ontology Enabled Web Search</name>
  <author>John</author>
  <conference>Web Intelligence</conference>
</article>
```

Fig. 1. An XML document.

$$d_x = \begin{matrix} & \text{thisDocument} \\ & \begin{bmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\ & \begin{matrix} \text{Ontology} \\ \text{Enabled} \\ \text{Web} \\ \text{Search} \\ \text{John} \\ \text{Intelligence} \end{matrix} \end{matrix}$$

Fig. 2. The document feature vector for the example shown in Figure 1.

$$\Delta_x = \begin{matrix} & \text{title} & \text{author} & \text{Confe} \\ & & & \text{rence} \\ & \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ & \begin{matrix} \text{Ontology} \\ \text{Enabled} \\ \text{Web} \\ \text{Search} \\ \text{John} \\ \text{Intelligence} \end{matrix} \end{matrix}$$

Fig. 3. The document feature matrix for the example in Figure 1.

$$\tilde{\Delta}_x = \begin{array}{ccc|l} \text{title} & \text{author} & \text{Confe} & \\ & & \text{rence} & \\ \hline 0.5 & 0 & 0 & \text{Ontology} \\ 0.5 & 0 & 0 & \text{Enabled} \\ 0.5 & 0 & \sqrt{2}/2 & \text{Web} \\ 0.5 & 0 & 0 & \text{Search} \\ 0 & 1 & 0 & \text{John} \\ 0 & 0 & \sqrt{2}/2 & \text{Intelligence} \end{array}$$

Fig. 4. The normalized document feature matrix for the example in Figure 1.

2.2. Similarity measures

Using VSM, similarity between two documents doc_x and doc_y is typically computed as the cosine value between their corresponding document feature vectors, given as

$$\text{sim}(doc_x, doc_y) = \frac{d_x \cdot d_y}{\|d_x\| \|d_y\|} = \tilde{d}_x \cdot \tilde{d}_y^T = \sum_{i=1}^k \tilde{d}_{x(i)} \tilde{d}_{y(i)} \quad (1)$$

where n is the total number of terms and $\tilde{d}_x = d_x / \|d_x\|$ denotes normalized d_x . So, the similarity measure can also be interpreted as the inner product of the normalized document feature vectors.

For SLVM, with the objective to model semantic relationships between XML elements, the corresponding document similarity can be defined with an element similarity matrix introduced.

Definition 2.4 The *SLVM-based document similarity* between two XML documents doc_x and doc_y is defined as

$$\text{sim}(doc_x, doc_y) = \sum_{j=1}^m \sum_{i=1}^m M_{e(i,j)} \cdot (\tilde{\Delta}_{x(i)}^T \bullet \tilde{\Delta}_{y(j)}) \quad (2)$$

where M_e is a matrix of dimension $m \times m$ and named as the *element similarity matrix*.

The matrix M_e in Eq. (2) captures both the similarity between a pair of XML elements as well as the contribution of the pair to the overall document similarity (*i.e.*, the diagonal elements of M_e are not necessarily equal to one). An entry in M_e being small means that the two corresponding XML elements should be unrelated and same words appearing in the two elements of two different documents will not contribute much to the overall similarity of them. If M_e is diagonal, this implies that all the XML

elements are not correlated at all with each other, which obviously is not the optimal choice. To obtain an optimal M_e for a specific type of XML data, we proposed in [5] to learn the matrix using pair-wise similar training data in an iterative manner.

2.3. SVM for XML Documents Classification

SVM was introduced by Vapnik in 1995 for solving two-class pattern recognition problems using the Structural Risk Minimization principle [6]. Given a training set containing two kinds of data (one for positive examples, the other for negative examples), which is linearly separable in vector space, this method finds the decision hyper-plane that best separated positive and negative data points in the training set. The problem searching the best decision hyper-plane can be solved using quadratic programming techniques. SVM can also extend its applicability to linearly nonseparable data sets by either adopting soft margin hyper-planes, or by mapping the original data vectors into a higher dimensional space in which the data points are linearly separable. Joachims [7] first applied SVM to text categorization, and compared its performance with other classification methods using the Reuters-21578 corpus. His results show that SVM outperformed all the other methods tested in his experiments.

SVM success in practice is drawn by its solid mathematical foundations which convey the following two salient properties:

- **Margin maximization:** The classification boundary functions of SVM maximize the margin, which in machine learning theory, corresponds to maximizing the *generalization* performance given a set of training data.
- **Nonlinear transformation of the feature space using the kernel trick:** SVM handle a nonlinear classification efficiently using the kernel trick which implicitly transforms the input space into another high dimensional feature space.

The kernel $k(x_i, x_j)$ could be regarded as the similarity function between two data points. For linear boundary, the kernel function is $x_i \cdot x_j$, a scalar product of two data points. The nonlinear transformation of the feature space is performed by replacing $k(x_i, x_j)$ with an advanced kernel, such as polynomial kernel $(x^T x_i + 1)^p$ or RBF kernel $\exp(-\frac{1}{2\delta^2} \|x - x_i\|^2)$.

In SLVM, the similarity between two XML documents is defined as definition 2.4, so we consider the kernel $k(x_i, x_j)$ for XML documents classification based on SLVM as:

$$k(x_i, x_j) = \text{sim}(doc_x, doc_y) = \sum_{j=1}^m \sum_{i=1}^m M_{e(i,j)} \cdot (\tilde{\Delta}_{x(i)}^T \bullet \tilde{\Delta}_{y(j)}) \quad (3)$$

3. Frequent Subtree

The form of SLVM studied in [2, 3, 5] is only a simplified one where only the leaf-node elements in the DTD are incorporated without considering their positions in the document DOM tree and their consecutive occurrence patterns.

In this paper, we utilize the frequent subtrees as structural units to extract the content information from the XML documents. A series of concepts of the subtree are defined same as in the paper [8, 9].

Let D denote a database where each transaction $s \in D$ is a labeled rooted unordered tree. For a given pattern t , which is a rooted unordered tree, we say t occurs in a transaction s if there exists at least one subtree of s that is isomorphic to t . The occurrence $\delta_t(s)$ of t in s is the number of distinct subtrees of s that are isomorphic to t . Let $\sigma_t(s) = 1$ if $\delta_t(s) > 0$, and 0 otherwise. We say s supports pattern t if $\sigma_t(s)$ is 1 and we define the support of a pattern t as $supp(t) = \sum_{s \in D} \sigma_t(s)$. A pattern t is called frequent if its support is greater than or equal to a minimum support (minsup) specified by a user.

We define a frequent tree t to be maximal if none of t 's proper supertrees is frequent, and closed if none of t 's proper supertrees has the same support that t has.

In this paper, we utilize *CMTreeminer* [9] to mining closed frequent subtrees from XML document collection, and the closed frequent subtrees as structural units are utilized to extract the content information from the XML documents.

4. Implementation and Evaluation Details

As outlined above, we classify the XML documents based on SLVM in the Document Mining Challenge of INEX 2009, where the closed frequent subtrees as structural units are used to extract content from the XML document.

4.1 Phase 1: Pre-processing of Structure

In the XML document collection of the INEX 2009, a group of "Template Element" is mapped to the tag "<sec>". In order to reduce the complexity of structure, the element "<sec>" is normalized by replacing its sub-element "<st>", whose meaning is "section title", as its attribute.

Each common path is replaced as a node, when any sub-path of it does not appear in the collection.

4.2 Phase 2: Mining closed frequent subtrees

We utilize *CMTreeminer* [9] to mining closed frequent subtrees from XML document collection.

There are a great deal *closed frequent subtrees* for the XML document collection, so we use the Chi-square test to select a part of *subtrees* as useful structural units.

4.3 Phase 3: Document Representation

Each document is represented as a matrix based on SLVM, where the selected *closed frequent subtrees* are regarded as structural unit.

In order to deal with exception that a document do not include any the selected *closed frequent subtrees*, we add the vector of the document based on VSM into the matrix as a column.

In addition, the interconnectivity between the documents based on link should also be considered. We add the vector of all link target document's title based on VSM into the matrix as a column.

4.4 Phase 4: Training, Tuning and Testing

The SVM algorithm in SVMTorch [10] is used for training and testing, with the formula (3) as kernel and the M_e is set as diagonal. For the multi-class document, the document is set as positive example for its each class.

In order to obtain optimal threshold, we split a part of training documents for tuning SVM classification threshold.

5. Experiment Result

In the experiments, all the algorithms were implemented by us in C++, except the SVM algorithm in SVMTorch [10]. All experiments were run on a PC with a 3.0 GHz Intel CPU and 2GB RAM.

In the XML Mining Track, the collection is composed of XML documents of the Wikipedia XML, the training set is composed of 10,969 XML documents, and the test set is composed of 43,606 XML documents.

We utilize *CMTreeminer* [9] to mining *closed frequent subtrees* from XML document collection, and select the top 10 *closed frequent subtrees* using the Chi-square test for each classification.

The table 1 summarizes the experiment result. The baseline is the approach based on original SLVM [2, 4]. The "No_Link" approach do not use the link information, and the "With_link" approach use the link information. The "Term_Selection" approach is that the term be selected by the Chi-square test.

Table 1. The experiment result.

Approaches	Macro F1	Micro F1	Mean Average precision by document
Baseline	0.241853	0.295418	0.486702
No_Link	0.479748	0.518635	0.7018610
With_Link	0.505916	0.546976	0.712323
Term_Selection	0.483367	0.525132	0.699250

6. Conclusion and Future Works

In this paper, we applied SLVM to XML documents classification, where the closed frequent subtrees as structural units are used for content extraction from the XML document and the Chi-square test is used for feature selection.

For future work, we are interested to study how to use link information, and combine the vector similarity method with graph theory methods.

Acknowledgment

The work reported in this paper was supported by the National Natural Science Foundation of China Grant 60642001 and 60875033.

References

1. Berry, M.: Survey of Text Mining: Clustering, Classification, and Retrieval, Springer (2003).
2. Yang, J., Chen, X.: A semi-structured document model for text mining. In: Journal of Computer Science and Technology, 17(5) pp 603-610 (2002)
3. Salton G, and McGill MJ, Introduction to Modern information Retrieval. McGraw-Hill, 1983.
4. Yang, J., Zhang, F.: XML Document Classification using Extended VSM, in Pre-Proceedings of the Sixth Workshop of Initiative for the Evaluation of XML Retrieval. 2007. Dagstuhl, Germany.
5. Yang, J., Cheung, W. K., Chen, X.O.: Learning Element Similarity Matrix for Semi-structured Document Analysis. Knowledge and Information Systems, 19: 53-78 (2009)
6. Vapnic, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
7. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In Proceedings of the 1998 European conference on Machine Learning (ECML), pages: 137-142 (1998)
8. Chi, Y., Nijssen, S., Muntz, R.R., Kok, J.N.: Frequent Subtree Mining -An Overview. Fundamenta Informaticae, 2005
9. Chi, Y., Yang Y., Xia Y., Muntz, R. R., CMTreeMiner: Mining Both Closed and Maximal Frequent Subtrees. The Eighth Pacific Asia Conference on Knowledge Discovery and Data Mining, 2004
10. Collobert, R. and Bengio, S.: SVM Torch: support vector machines for large-scale regression problems, Journal of Machine Learning Research, Vol.1, pp. 143-160 (2001)

Author Index

Aji, Ablimit198	Geva, Shlomo . 16, 209, 290, 326, 343, 365
Altingovde, Ismail Sengor 349	Giguët, Emmanuel 136
Atilgan, Duygu 349	Grün, Christian 192
Balog, Krisztian 238	Gurevych, Iryna 314
Baudrillard, Alexandre 136	Géry, Mathias 60, 381
Beigbeder, Michel 75	Hagenbuchner, Markus 302, 367
Beltran, Beatriz391	Hatano, Kenji 67
Bhirud, Dinesh133	He, Jiyin 238
Bron, Marc 238	Hoffart, Johannes 314
Broschart, Andreas 188	Holupirek, Alexander 192
Bruza, Peter 13	Huang, Darren 290
Buffoni, David 51	Huete, Juan F. 398
Bär, Daniel 314	
Campos, Luis M. de398	Ibekwe-SanJuan, Fidelia 99
Chau, Rowena 302	Imafouo, Amelie 75
Chevallet, Jean-Pierre 85	Iofciu, Tereza 233
Chidlovskii, Boris 355	Itakura, Kelly Y. 249
Clarke, Charles L. A. 249	
Crouch, Carolyn 133	Jia, Xiang-Fei 209
Crouch, Donald 133	
Demartini, Gianluca 233	Kamps, Jaap 16, 260
Deng, Zhi-Hong 53	Kaptein, Rianne 260
Denoyer, Ludovic 339	Karthik, Venkatesh 273
De Vine, Lance 365	Kazai, Gabriella 120
De Vries, Chris 343, 365	Kc, Milly 302
Doucet, Antoine 120	Keyaki, Atsushi 67
Du, Xiaoyong 108	Koolen, Marijn 120, 260
Déjean, Hervé 143	Kutty, Sangeetha 343, 379
	Kühne, Gerold 222
Fernández-Luna, Juan M. 398	Landoni, Monica 120
	Largeron, Christine 60, 381
Gallinari, Patrick 51, 339	Larson, Ray 153
Ganguly, Debasis 94	Lee, Vincent 302
Gao, Ning 53	Lehtonen, Miro 16
Gath, Sebastian 192	Li, Qiushi 108
Geng, Yina 108	Li, Rongmei 163

Li, Yuefeng	379	Thom, James A.	16, 106
Lucas, Nadine	136	Tovar, Mireya	391
		Trotman, Andrew ...	16, 209, 290, 326
		Tsoi, Ah Chung	302, 367
Masegosa, Andrés R.	398		
Mercier, Annabelle	75	Ulusoy, Özgür	349
Meunier, Jean-Luc	143	Usunier, Nicolas	51
Mitra, Mandar	94		
Miyazaki, Jun	67	Vilariño, Darnes	391
Moriceau, Véronique	334	Vries, Arjen de	233
Moulin, Christophe	381		
Mukherjee, Saswati	273	Wang, Qiuyue	108
Mulhem, Philippe	85	Wang, Shan	108
Murugesan, Meenakshi Sundaram	273	Wang, Songlin	408
		Weerkamp, Wouter	238
Nayak, Richi	343, 379	Winter, Judith	222
Nordlie, Ragnar	170, 282	Witten, Ian	15
Pal, Sukomal	94	Xiang, Yong-Qing	53
Paris, Cecile	14		
Pharo, Nils	170, 282	Yang, Jianwu	408
Pinto, David	391	Yu, Hang	53
Polumetla, Chaitanya	133		
Poluri, Pavan	133	Zesch, Torsten	314
Preminger, Michael	170	Zhang, ShuJia	367
Qin, Zuoyan	108		
Rajagopal, Srikant	273		
Ramanathan, Madhu	273		
Rijke, Maarten de	238		
Romero, Alfonso E.	398		
Salazar, Héctor Jiménez	391		
SanJuan, Eric	99, 334		
Scarselli, Franco	367		
Schenkel, Ralf	16, 175, 188, 198		
Scholl, Marc H.	192		
Sudhaker, Varun	133		
Tang, Eric	326		
Tannier, Xavier	334		
Theobald, Martin	175, 198		

ISBN 978-90-814485-2-9



9 789081 448529

90000 >