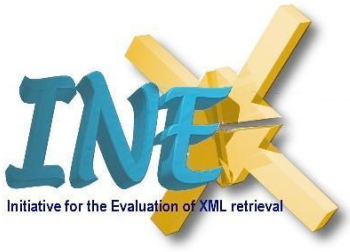


INEX 2008 Workshop Pre-proceedings

Shlomo Geva, Jaap Kamps, Andrew Trotman
(editors)

December 15-18, 2008
Schloss Dagstuhl, Leibniz Center for Informatics
http://www.inex.otago.ac.nz/

Preface

Welcome to the 7th workshop of the Initiative for the Evaluation of XML Retrieval (INEX)!

Now, in its seventh 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 2008 was an exciting year for INEX, and brought a lot of changes. In total eight research tracks were included, which studied different aspects of focused information access:

Ad hoc Track The main track of INEX 2008 will investigate the effectiveness of XML-IR and Passage Retrieval for three ad hoc retrieval tasks (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 Efficiency and Question Answering tracks were new for 2008. The consolidation of the existing tracks, and the expansion to new areas offered by the two new tracks, allows INEX to grow in reach.

The aim of the INEX 2008 workshop is to bring together researchers in the field of XML IR who participated in the INEX 2008 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.

This is also the seventh INEX Workshop at the *Schloss Dagstuhl – Leibniz Center for Informatics*, providing the unique setting where informal interaction and discussion occurs naturally and frequently. It is clear that this has been essential to the growth of INEX over the years, we feel honored and priviliged for *Dagstuhl* housing the INEX 2008 Workshop.

Finally, INEX is run for but especially by the participants. It is a result of tracks and tasks suggested by participants, topics created by particants, systems build by participants, and relevance judgments provided by participants. So the main thank you goes each of these individuals!

December 2008

Shlomo Geva
Jaap Kamps
Andrew Trotman
Chairs of INEX 2008

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 (General, Queensland University of Technology)
Jaap Kamps (General, University of Amsterdam)
Andrew Trotman (General, University of Otago)
Ludovic Denoyer (Document Collection, University Paris 6)
Ralf Schenkel (Document Exploration, Max-Planck-Institut fur Informatik)
Martin Theobald (Document Exploration, Stanford University)

Book

Antoine Doucet (University of Caen)
Gabriella Kazai (Microsoft Research Limited)
Monica Landoni (University of Strathclyde)

Efficiency

Martin Theobald (Stanford University)

Entity Ranking

Arjen de Vries (CWI)
Gianluca Demartini (L3S)
Tereza Iofciu (L3S)
Jianhan Zhu (University College London)

Interactive (iTrack)

Nisa Fachry (University of Amsterdam)
Ragnar Nordlie (Oslo University College)
Nils Pharo (Oslo University College)

Question Answering (QA)

Valentin Jijkoun (University of Amsterdam)
Maarten de Rijke (University of Amsterdam)

Link-the-Wiki

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

XML-Mining

Ludovic Denoyer (University Paris 6)
Patrick Gallinari (University Paris 6)

Dagstuhl

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

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

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

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

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

The *Schloss Dagstuhl – Leibniz Center for Informatics* is the world's premier venue for informatics. It enables the international elite, promising young researchers and practicians alike to gather together to discuss their views and research findings.

http://www.dagstuhl.de/

Table of Contents

Front matter.

Adhoc.

Entity Ranking.

Interactive.

Link the Wiki.

XML Mining.

Back matter.

Overview of the INEX 2008 Ad Hoc Track

Jaap Kamps[1], Shlomo Geva[2], Andrew Trotman[3],
Alan Woodley[2], and Marijn Koolen[1]

[1] University of Amsterdam, Amsterdam, The Netherlands
{kamps,m.h.a.koolen}@uva.nl
[2] Queensland University of Technology, Brisbane, Australia
{s.geva,a.woodley}@qut.edu.au
[3] University of Otago, Dunedin, New Zealand
andrew@cs.otago.ac.nz

Abstract. This paper gives an overview of the INEX 2008 Ad Hoc Track. The main goals of the Ad Hoc Track were two-fold. The first goal was to investigate the value of the internal document structure (as provided by the XML mark-up) for retrieving relevant information. This is a continuation of INEX 2007 and, for this reason, the retrieval results are liberalized to arbitrary passages and measures were chosen to fairly compare systems retrieving elements, ranges of elements, and arbitrary passages. The second goal was to compare focused retrieval to article retrieval more directly than in earlier years. For this reason, standard document retrieval rankings have been derived from all runs, and evaluated with standard measures. In addition, a set of queries targeting Wikipedia have been derived from a proxy log, and the runs are also evaluated against the clicked Wikipedia pages. The INEX 2008 Ad Hoc Track featured three tasks: 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 results for the three 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. Finally, we look at the ability of focused retrieval techniques to rank articles, using standard document retrieval techniques, both against the judged topics as well as against queries and clicks from a proxy log.

1 Introduction

This paper gives an overview of the INEX 2008 Ad Hoc Track. There are two main research question underlying the Ad Hoc Track. The first 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 in identify where the relevant information is within a document? This question, first studied at INEX 2007, has attracted a lot of attention in recent years.

Trotman and Geva [11] argued that, since INEX relevance assessments are not bound to XML element boundaries, retrieval systems should also not be bound to XML element boundaries. Their implicit assumption is that a system returning passages is at least as effective as a system returning XML elements. This assumption is based on the observation that elements are of a lower granularity than passages and so all elements can be described as passages. The reverse, however is not true and only some passages can be described as elements. Huang et al. [4] implement a fixed window passage retrieval system and show that a comparable element retrieval ranking can be derived. In a similar study, Itakura and Clarke [5] show that although ranking elements based on passage-evidence is comparable, a direct estimation of the relevance of elements is superior. Finally, Kamps and Koolen [6] study the relation between the passages highlighted by the assessors and the XML structure of the collection directly, showing reasonable correspondence between the document structure and the relevant information.

Up to now, element and passage retrieval approaches could only be compared when mapping passages to elements. This may significantly affect the comparison, since the mapping is non-trivial and, of course, turns the passage retrieval approaches effectively into element retrieval approaches. 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 now 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 [2] and later requests to liberalize to arbitrary passages of text [11].

The second main question is to compare focused retrieval directly to traditional article retrieval. Throughout the history of INEX, participating groups have found that article retrieval—a system retrieving the whole article by default—resulted in fairly competitive performance [e.g., 7, 10]. Note that every focused retrieval system also generates an underlying article ranking, simply by the order is which results from different articles are ranked. This is most clear in the Relevant in Context and Best in Context tasks, where the article ranking is an explicit part of the task description. To study the importance of the underlying article ranking quality, we derived article level judgments by treating every article with some highlighted text as relevant, derived article rankings from every submission on a first-come, first-served basis, and evaluated with standard measures. This will also shed light on the value of element or passage level evidence for document retrieval [1]. In addition to this, we also include queries derived from a proxy log in the topic set, and can derive judgments from the later clicks in the same proxy log, treating all clicked articles as relevant for the query at

hand. All submissions are also evaluated against these clicked Wikipedia pages, giving some insight in the differences between an IR test collection and real-world searching of Wikipedia.

The INEX 2008 Ad Hoc Track featured three tasks:

1. 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.
2. 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.
3. 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 three tasks, giving results for the top 10 participating groups and discussing the 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 2008 ad hoc retrieval tasks and measures. Section 3 details the collection, topics, and assessments of the INEX 2008 Ad Hoc Track. In Section 4, we report the results for the Focused Task (Section 4.2); the Relevant in Context Task (Section 4.3); and the Best in Context Task (Section 4.4). Section 5 details particular types of runs (such as CO versus CAS, and element versus passage), 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, both in terms of the judged topics treating any article with highlighted text as relevant, and in terms of clicked Wikipedia pages for queries derived from a proxy log. 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

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 2008, we allowed the submission of three types of results: XML elements; ranges of XML elements; and file-offset-length (FOL) text passages.

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 so that (with the .xml extension removed). The next example identifies 9996.xml as the target document from the Wikipedia collection.

```
<file>9996</file>
```

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.

```
<path>/article[1]/body[1]/section[1]/p[1]</path>
```

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, then, is identified unambiguously using the combination of file name and element path, as shown in the next example.

```
<result>
  <file>9996</file>
  <path>/article[1]/body[1]/section[1]/p[1]</path>
  <rsv>0.9999</rsv>
</result>
```

Ranges of Elements To support ranges of elements, elemental passages are given in the same format.[1] As a passage need not start and end in the same element, each is given separately. The following example is equivalent to the element result example above since it starts and ends on an element boundary.

```
<result>
  <file>9996</file>
  <passage start="/article[1]/body[1]/section[1]/p[1]"
     end="/article[1]/body[1]/section[1]/p[1]"/>
  <rsv>0.9999</rsv>
</result>
```

Note that this format is very convenient for specifying ranges of elements, e.g., the following example retrieves the first three sections.

```
<result>
  <file>9996</file>
  <passage start="/article[1]/body[1]/section[1]"
     end="/article[1]/body[1]/section[3]"/>
  <rsv>0.9999</rsv>
</result>
```

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

[1] At INEX 2007, and in earlier qrels, an extended format allowing for optional character-offsets was used that allowed these passages to start or end in the middle of element or text-nodes. This format is superseded with the clean file-offset-length (FOL) passage format.

the collection is provided to facilitate the use of passage retrieval systems. File offsets start counting a 0 (zero).

The following example is effectively equivalent to the example element result above.

```
<result>
  <file>9996</file>
  <fol offset="461" length="202"/>
  <rsv>0.9999</rsv>
</result>
```

The paragraph starts at the 462th character (so 461 characters beyond the first character), and has a length of 202 characters.

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 2008 measures are solely based on the retrieval of highlighted text. We simplify all INEX tasks to highlighted text retrieval and assume that systems 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.

Focused Task Recall is measured as the fraction of all highlighted text that has been retrieved. Precision is measured as the fraction of retrieved text that was highlighted. 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 what happens in the first retrieved results, the INEX 2008 official 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 [9], 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 (at INEX 2007, F_1 was used). 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 (at INEX 2007, $n = 1,000$ was used). 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).

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

The document collection was the Wikipedia XML Corpus based on the English Wikipedia in early 2006 [3]. The Wikipedia collection contains 659,338 Wikipedia articles. On average an article contains 161 XML nodes, where the average depth of a node in the XML tree of the document is 6.72.

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. For details see Denoyer and Gallinari [3].

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 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. Figure 1 presents an example of an ad hoc topic. Based on the submitted candidate topics, 135 topics were selected for use in the INEX 2008 Ad Hoc Track as topic numbers 544–678.

```
<topic id="544" ct_no="6">
  <title>meaning of life</title>
  <castitle>
    //article[about(., philosophy)]//section[about(., meaning of life)]
  </castitle>
  <description>What is the meaning of life?</description>
  <narrative>
    I got bored of my life and started wondering what the meaning of
    life is. An element is relevant if it discusses the meaning of life
    from different perspectives, as long as it is serious. For example,
    Socrates discussing meaning of life is relevant, but something like
    "42" from H2G2 or "the meaning of life is cheese" from a comedy is
    irrelevant. An element must be self contained. An  element that is a
    list of links is considered irrelevant because it is not
    self-contained in the sense that I don't know in which context the
    links are given.
  </narrative>
</topic>
```

Fig. 1. INEX 2008 Ad Hoc Track topic 544.

In addition, 150 queries were derived from a proxy-log for use in the INEX 2008 Ad Hoc Track as topic numbers 679–828. For these topics, as well as the candidate topics without a ⟨`castitle`⟩ field, a default CAS-query was added based on the CO-query: `//*[about(., "`*CO-query*`")]`.

3.3 Judgments

Topics were assessed by participants following precise instructions. The assessors used the new 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 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.

The relevance judgments were frozen on October 22, 2008. At this time 70 topics had been fully assessed. Moreover, 11 topics were judged by two separate assessors, each without the knowledge of the other. All results in this paper refer to the 70 topics with the judgments of the first assigned assessor, which is typically the topic author.

– The 70 assessed topics were: 544–547, 550–553, 555–557, 559, 561, 562–563, 565, 570, 574, 576–582, 585–587, 592, 595–598, 600–603, 607, 609–611, 613, 616–617, 624, 626, 628, 629, 634–637, 641–644, 646–647, 649–650, 656–657, 659, 666–669, 673, 675, and 677.

In addition, there are clicked Wikipedia pages available in the proxy log for 125 topics:

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

	total		# per topic				
	topics	number	min	max	median	mean	st.dev
judged articles	70	42,105	502	618	603	601.5	15.3
articles with relevance	70	4,850	2	375	49	69.3	68.7
highlighted passages	70	7,510	3	906	59	107.3	131.0
highlighted characters	70	11,337,505	1,419	1,113,578	99,569	161,964.4	132,544.9
Unique articles with clicks	125	225	1	10	1	1.8	1.5
Total clicked articles	125	532	1	24	3	4.3	3.8

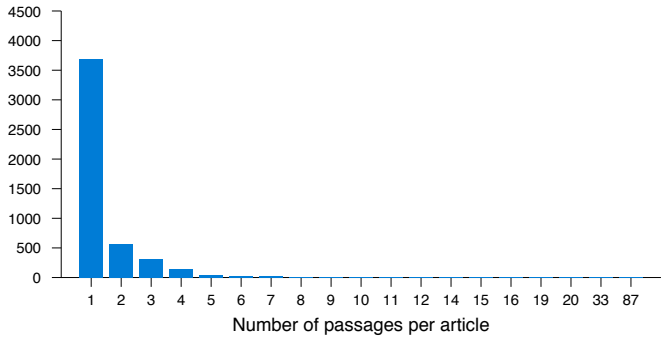


Fig. 2. Distribution of passages over articles.

– The 125 topics with clicked articles are numbered: 679–682, 684–685, 687–693, 695–704, 706–708, 711–727, 729–732, 734–751, 753–776, 778, 780–782, 784, 786–787, 789–790, 792–793, 795–796, 799–804, 806–807, 809–810, 812–813, 816–819, 821–824, and 826–828.

Table 1 presents statistics of the number of judged and relevant articles, and passages. In total 42,105 articles were judged. Relevant passages were found in 4,850 articles. The mean number of relevant articles per topic is 69, but the distribution is skewed with a median of 49. There were 7,510 highlighted passages. The mean was 107 passages and the median was 59 passages per topic.[2]

Table 1 also includes some statistics of the number of clicked articles in the proxy log. There are in total 225 clicked articles (unique per topic) over in total 125 topics, with a mean of 1.8 and a median of 1 clicked article per topic. We filtered the log for queries issued by multiple persons, and can also count the total number of clicks. Here, we see a total of 532 clicks (on the same 225 articles before), with a mean of 4.3 and a median of 3 clicks per topic. It is clear that the topics and clicked articles from the log are very different in character from the ad hoc topics.

[2] Recall from above that for the Focused Task the main effectiveness measures is precision at 1% recall. Given that the average topic has 107 relevant passages in 69 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 best entry point judgement.

	# topics	number	min	max	median	mean	st.dev
best entry point offset	70	4,850	1	87,982	14	1,746.2	4,826.5
first relevant character offset	70	4,850	1	87,982	20	1,821.4	4,862.9
fraction highlighted text	70	4,850	0.0005	1.000	0.580	0.549	0.425

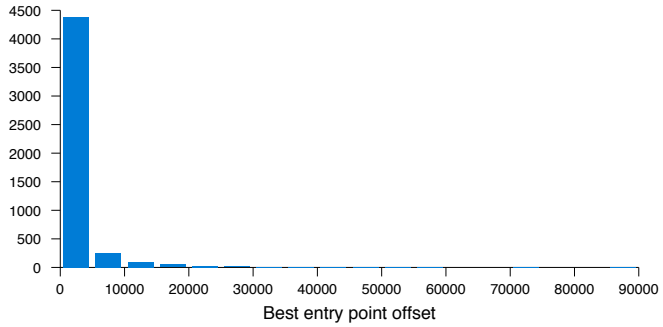


Fig. 3. Distribution of best entry point offsets.

Figure 2 presents the number of articles with the given number of passages. The vast majority of relevant articles (3,696 out of 4,850) had only a single highlighted passage, and the number of passages quickly tapers off.

Assessors where 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 1,746 but the distribution is very skewed with a median BEP offset of only 14. Figure 3 shows the distribution of the character offsets of the 4,850 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 1,821 but the median offset is only 20. This suggests a relation between the BEP offset and the FRC offset. Figure 4 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.

Finally, the statistics on the fraction of highlighted text in Table 2 show that amount of relevant text varies from almost nothing to almost everything. The mean fraction is 0.55, and the median is 0.58, indicating that typically over half the article is 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.

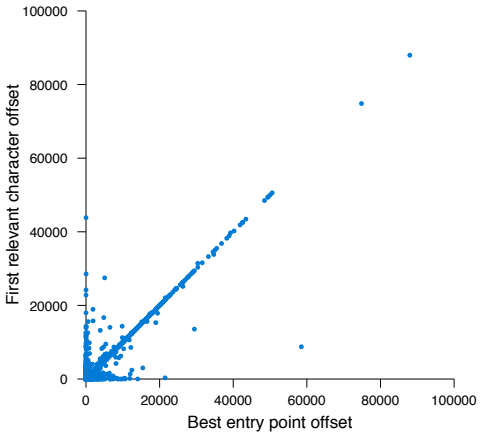


Fig. 4. Scatter plot of best entry point offsets versus the first relevant character.

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)

3.4 Questionnaires

At INEX 2008, 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

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)

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 2008 Ad Hoc Track.

4 Ad Hoc Retrieval Results

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

4.1 Participation

A total of 163 runs were submitted by 23 participating groups. Table 5 lists the participants and the number of runs they submitted, also broken down over the tasks (Focused, Relevant in Context, or Best in Context); the used query (Content-Only or Content-And-Structure); and the used result type (Element, Passage or FOL). Unfortunately, no less than 27 runs turned out to be invalid and will only be evaluated with respect to their "article retrieval" value in Section 6.

Participants were allowed to submit up to three element result-type runs per task and three passage result-type runs per task (for all three tasks). This

Table 5. Participants in the Ad Hoc Track.

Id	Participant	Focused	Relevant in Context	Best in Context	CO query	CAS query	Element results	Passage results	FOL results	# valid runs	# submitted runs
4	University of Otago	0	6	0	6	0	3	3	0	6	6
5	Queensland University of Technology	6	6	6	15	3	9	9	0	18	18
6	University of Amsterdam	6	6	3	9	6	13	0	2	15	15
9	University of Helsinki	3	0	0	3	0	3	0	0	3	3
10	Max-Planck-Institut Informatik	3	1	1	5	0	5	0	0	5	5
12	University of Granada	3	3	3	9	0	9	0	0	9	9
14	University of California, Berkeley	2	0	1	3	0	3	0	0	3	3
16	University of Frankfurt	1	3	3	0	7	7	0	0	7	9
22	ENSM-SE	2	0	0	2	0	0	0	2	2	9
25	Renmin University of China	3	0	1	2	2	4	0	0	4	4
29	INDIAN STATISTICAL INSTITUTE	3	0	0	3	0	3	0	0	3	3
37	Katholieke Universiteit Leuven	6	0	0	3	3	6	0	0	6	6
40	IRIT	0	0	2	1	1	2	0	0	2	6
42	University of Toronto	2	0	0	0	2	2	0	0	2	3
48	LIG	3	2	0	5	0	5	0	0	5	5
55	Doshisha University	0	0	1	0	1	1	0	0	1	3
56	JustSystems Corporation	3	3	3	6	3	9	0	0	9	9
60	Saint Etienne University	3	0	0	3	0	3	0	0	3	9
61	Universit Libre de Bruxelles	0	0	0	0	0	0	0	0	0	2
68	University Pierre et Marie Curie - LIP6	2	0	0	2	0	2	0	0	2	2
72	University of Minnesota Duluth	2	2	2	6	0	6	0	0	6	6
78	University of Waterloo	3	3	4	10	0	8	2	0	10	13
92	University of Lyon3	5	5	5	15	0	15	0	0	15	15
	Total runs	61	40	35	108	28	118	14	4	136	163

totaled to 18 runs per participant.[3] The submissions are spread well over the ad hoc retrieval tasks with 61 submissions for Focused, 40 submissions for Relevant in Context, and 35 submissions for Best in Context.

4.2 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 6 shows the best run of the top 10 participating groups. The first column gives the

[3] As it turns out, two groups submitted more runs than allowed: *University of Lyon3* submitted 6 extra element runs, and *University of Amsterdam* submitted 4 extra element runs. At this moment, we have not decided on any repercussions other than mentioning them in this footnote.

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

Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p78-FOERStep	0.7657	0.6873	0.5700	0.4879	0.2071
p10-TOPXCOarti	0.6804	0.6795	0.5807	0.5265	0.2967
p48-LIGMLFOCRI	0.7114	0.6665	0.5210	0.4216	0.1441
p92-manualQEin*	0.6664	0.6664	0.6139	0.5540	0.3065
p60-JMUexpe142	0.6918	0.6640	0.5800	0.4986	0.2342
p9-UHelRun394	0.7109	0.6619	0.5532	0.5028	0.2251
p14-T2FBCOPARA	0.7319	0.6395	0.4906	0.4026	0.1392
p25-weightedfi	0.6553	0.6346	0.5490	0.5222	0.2647
p5-GPX1COFOCe	0.6818	0.6344	0.5693	0.5178	0.2587
p29-LMnofb020	0.6830	0.6337	0.5537	0.5100	0.2847

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]).

University of Waterloo Element 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 using Okapi BM25. In addition, scores were boosted by doubling the tf values of the first 10 words of an element.

Max-Planck-Institut für Informatik Element retrieval run using the CO query. Description: The TopX system retrieving only article elements, using a linear combination of a BM25 content score with a BM25 proximity score that also takes document structure into accout.

LIG Grenoble An element retrieval run using the CO query. Description: Based on a language Model using a Dirichlet smoothing, and equally weighting element score and its context score, where the context score are based on the collection-links in Wikipedia.

University of Lyon3 A *manual* element retrieval run using the CO query. Description: Using indri search engine in Lemur with manually expanded queries from CO, description and narrative fields. The run is retrieving only articles.

Saint Etienne University An element retrieval run using the CO query. Description: A probabilistic model used to evaluate a weight for each tag: "the probability that tags distinguishes terms which are the most relevant", i.e. based on the fact that the tag contains relevant or non relevant passages. The resulting tag weights are incorporated into an element-level run with BM25 weighting.

Based on the information from these and other participants:

– All ten runs use the CO query. The fourth run, *p92-manualQEin*, uses a manually expanded query using words from the description and narrative

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

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p78-RICBest	0.4052	0.3414	0.2739	0.2184	0.2263
p5-GPX1CORICe	0.3737	0.3430	0.2658	0.2135	0.2097
p92-manualQEin*	0.4138	0.3564	0.2659	0.2069	0.2092
p10-TOPXCOallA	0.3650	0.3049	0.2352	0.1908	0.1933
p4-WHOLEDOC	0.3696	0.3242	0.2476	0.1944	0.1917
p6-inex08artB	0.3481	0.2991	0.2200	0.1726	0.1752
p72-UMDRic2	0.3828	0.3341	0.2342	0.1882	0.1714
p12-p8u3exp511	0.2933	0.2710	0.2154	0.1612	0.1575
p56-VSMRIP05	0.3281	0.2638	0.2097	0.1608	0.1495
p48-LIGMLRIC4O	0.3595	0.3069	0.2303	0.1708	0.1486

fields. The tenth run, *p29-LMnofb020*, is an automatic run using the title and description fields. All other runs use only the CO query in the title field.
- All runs retrieve elements as results.
- The systems at rank second (*p10-TOPXCOarti*), fourth (*p92-manualQEin*), and tenth (*p29-LMnofb020*), are retrieving only full articles.

4.3 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 7 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).

University of Waterloo Element 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 using Okapi BM25, and grouped the results by articles and ranked the articles by their best scoring element.

Queensland University of Technology Element retrieval run using the CO query. Description: GPX run using a `//*[about(.,keywords)]` query, serving non-overlapping elements grouped per article, with the articles ordered by their best scoring element.

University of Lyon3 A manual element retrieval run using the CO query. Description: the same as the Focused run above. In fact it is literally the same article ranking as the Focused run. Recall that the run is retrieving only whole articles.

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

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p78-BICER	0.3811	0.3233	0.2498	0.1979	0.2207
p92-manualQEin*	0.4087	0.3645	0.2784	0.2203	0.2171
p25-weightedfi	0.3454	0.3003	0.2481	0.2004	0.2009
p5-GPX1COBICp	0.3655	0.3367	0.2572	0.2019	0.1970
p6-submitinex	0.3447	0.2870	0.2203	0.1681	0.1693
p10-TOPXCOallB	0.2392	0.2321	0.1875	0.1528	0.1689
p12-p8u3exp501	0.2490	0.2303	0.1935	0.1487	0.1457
p72-UMDBIC1	0.3163	0.2710	0.1857	0.1451	0.1438
p56-VSMRIP08	0.2269	0.2010	0.1736	0.1397	0.1311
p40-xfirmcos07	0.2402	0.1869	0.1347	0.1083	0.0951

Max-Planck-Institut für Informatik Element retrieval run using the CO query. Description: An element retrieval run using the new BM25 scoring function (i.e., considering each element as "document" and then computing a standard BM25 model), selecting non-overlapping elements based on score, and grouping them per article with the articles ranked by their highest scoring element.

University of Otago Element retrieval run using the CO query. Description: BM25 is used to select and rank the top 1,500 documents and whole documents are selected as the passage. That is, the run is retrieving only whole articles.

Based on the information from these and other participants:

- The runs ranked sixth (*p6-inex08artB*) and ninth (*p56-VSMRIP05*) are using the CAS query. The run ranked third, *p92-manualQEin*, is using a manually expanded query based on keywords in the description and narrative. All other runs use only the CO query in the topic's title field.
- All runs retrieve elements as results.
- Solid article ranking seems a prerequisite for good overall performance, with third best run, *p92-manualQEin*, the fifth best run, *p4-WHOLEDOC*, and the ninth best run, *p56-VSMRIP05*, retrieving only full articles.

4.4 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 8 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).

University of Waterloo Element 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 using Okapi BM25, and kept only the best scoring element per article.

University of Lyon3 A manual element retrieval run using the CO query. Description: the same as the Focused and Relevant in Context runs above. In fact all three runs have literally the same article ranking. This run is retrieving the start of the whole article as best entry point, in other words an article retrieval run.

Renmin University of China Element retrieval run using the CO query. Description: using language model to compute RSV at leaf level combined with aggregation at retrieval time, assuming independence.

Queensland University of Technology Run retrieving ranges of elements using the CO query. The run is always returning a whole article, setting the BEP at the very start of the article. Description: GPX run using a `//*[about(.,keywords)]` query, ranking articles by their best scoring element, but transformed to return the complete article as a passages. This is effectively an article level GPX run.

University of Amsterdam Run retrieving FOL passages using the CO query. Description: language model with local indegree prior, setting the BEP always at the start of the article. Since the offset is always zero, this is similar to an article retrieval run.

Based on the information from these and other participants:

- 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 of all the retrieved articles at rank two (*p92-manualQEin*), rank four (*p5-GPX1COBICp*), and rank five (*p6-submitinex*).
- The fourth ranked run, *p5-GPX1COBICp*, uses ranges of elements, albeit a degenerate case where always the full article is selected. The fifth run, *p6-submitinex*, uses fol passages, albeit again a degenerate case where the BEP is always the zero offset.
- With the exception of the runs ranked nine (*p56-VSMRIP08*) and ten (*p40-xfirmcos07*), which used the CAS query, all the other best runs per group use the CO query.

4.5 Significance Tests

We tested whether higher ranked systems were significantly better than lower ranked system, using a t-test (one-tailed) at 95%. Table 9 shows, for each task, whether it is significantly better (indicated by "⋆") than lower ranked runs. For

Table 9. Statistical significance (t-test, one-tailed, 95%).

(a) Focused Task

	1	2	3	4	5	6	7	8	9	10
p78	-	-	-	-	-	-	-	-	-	-
p10		-	-	-	-	-	-	-	-	-
p48			-	-	-	-	-	-	-	-
p92				-	-	-	-	-	-	-
p60					-	-	-	-	-	-
p9						-	-	-	-	-
p14							-	-	-	-
p25								-	-	-
p5									-	-
p29										

(b) Relevant in Context Task

	1	2	3	4	5	6	7	8	9	10
p78	⋆	-	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
p5		-	⋆	-	⋆	⋆	⋆	⋆	⋆	⋆
p92			-	-	⋆	⋆	⋆	⋆	⋆	⋆
p10				-	-	-	⋆	⋆	⋆	⋆
p4					-	-	⋆	⋆	⋆	⋆
p6						-	-	-	⋆	⋆
p72							-	-	⋆	⋆
p12								-	-	⋆
p56									-	⋆
p48										

(c) Best in Context Task

	1	2	3	4	5	6	7	8	9	10
p78	-	-	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
p92		-	-	⋆	⋆	⋆	⋆	⋆	⋆	⋆
p25			-	⋆	⋆	⋆	⋆	⋆	⋆	⋆
p5				⋆	-	⋆	⋆	⋆	⋆	⋆
p6					-	-	-	⋆	⋆	⋆
p10						-	-	⋆	⋆	⋆
p12							-	-	⋆	⋆
p72								-	⋆	⋆
p56									⋆	⋆
p40										

example, For the Focused Task, we see that the early precision (at 1% recall) is a rather unstable measure and none of the runs are significantly different. 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 better than ranks 4 and 6 through 10, the third ranked system better than ranks 6 through 10, and the fourth and fifth ranked systems better than ranks 8 through 10. For the Best in Context Task, we see that the top run is significantly better than ranks 4 through 10, the second and third runs significantly better than than ranks 5 to 10. The fourth ranked system is better than the systems ranked 5 and 7 to 10, and the fifth ranked system better than ranks 9 and 10.

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 18 submissions using ranges of elements of FOL-passage results, from in total 5 participating groups. We will look at the relative effectiveness of element and passage runs.

As we saw above, in Section 4, for all three tasks the best scoring runs used elements as the unit of retrieval. Table 10 shows the best runs using ranges of elements or FOL passages for the three ad hoc tasks. All these runs use the CO query. As it turns out, the best focused run using passages ranks outside the top scoring runs in Table 6; the best relevant in context run using passages is ranked fifth among the top scoring runs in Table 7; and the best best in context run using passages is ranked fourth among the top scoring runs in Table 8. This outcome is consistent with earlier results using passage-based element retrieval,

Table 10. Ad Hoc Track: Runs with ranges of elements or FOL passages.

(a) Focused Task

Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p5-GPX2COFOCp	0.6308	0.6301	0.5379	0.4699	0.2502
p22-EMSEFocuse*	0.6745	0.5703	0.4481	0.3845	0.1545

(b) Relevant in Context Task

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p4-WHOLEDOCPA	0.3696	0.3242	0.2476	0.1944	0.1917
p5-GPX1CORICp	0.3544	0.3209	0.2413	0.1860	0.1891

(c) Best in Context Task

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p5-GPX1COBICp	0.3655	0.3367	0.2572	0.2019	0.1970
p6-submitinex	0.3447	0.2870	0.2203	0.1681	0.1693
p78-BICPRplus	0.2585	0.2206	0.1642	0.1248	0.1237

where passage retrieval approaches showed comparable but not superior behavior to element retrieval approaches [4, 5].

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-GPX2COFOCp* is an article retrieving run using ranges of elements; and *p22-EMSEFocuse* is a manual query run using FOL passages. For Relevant in Context, both *p4-WHOLEDOCPA* and *p5-GPX1CORICp* are article retrieving runs using ranges of elements. For Best in Context, *p5-GPX1COBICp* is an article runs using ranges of elements; *p6-submitinex* is an article run using FOL passages; and *p78-BICPRplus* is an element retrieving run using ranges of elements. So, all but two of the runs retrieve only articles. 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 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 11 show the distribution of target elements. In total 86 topics had a non-trivial CAS query.[4] These CAS topics

[4] Note that some of the wild-card topics (using the "*" target) in Table 11 had non-trivial about-predicates and hence have not been regarded as trivial CAS queries.

Table 11. CAS query target elements over all 135 topics.

Target Element	Frequency
*	51
section	39
article	30
p	11
figure	3
body	1

are numbered 544–550, 553–556, 564, 567, 568, 572, 574, 576–578, 580, 583, 584, 586–591, 597–605, 607, 608, 610, 615–625, 627, 629–633, 635–640, 646, 651–655, 658, 659, 661–670, 673, and 675–678. As it turned out, 39 of these CAS topics were assessed. The results presented here are restricted to only these 39 CAS topics.

Table 12 lists the top 10 participants measured using just the 39 CAS topics and for the Focused Task (a), the Relevant in Context Task (b), and the Best in Context Task (c). For the Focused Task the CAS runs score lower than the CO query runs. For the Relevant in Context Task, the best CAS run would have ranked fifth among the CO runs. For the Best in Context Task, the best CAS run would rank seventh among the CO runs. Overall, we see the that teams submitting runs with both types of queries have higher scoring CO runs, with participant 6 as a notable exception for Relevant in Context.

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. We look first at the article rankings in terms of the Ad Hoc Track judgements—treating every article that contains highlighted text as relevant. Then, we look at the article rankings in terms of the clicked pages for the topics derived from the proxy log—treating every clicked articles as relevant.

6.1 Article retrieval: Relevance Judgments

We will first look at the topics judged during INEX 2008, the same topics as in earlier sections, but now using the judgments to derive standard document-level relevance by regarding an article as relevant if some part of it is highlighted by the assessor. Throughout this section, 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.

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

Table 12. Ad Hoc Track CAS Topics: CO runs (left-hand side) versus CAS runs (right-hand side).

(a) Focused Task

Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p60-JMUexpe136	0.7321	0.7245	0.6416	0.5861	0.2926
p48-LIGMLFOCRI	0.7496	0.7209	0.5307	0.4440	0.1569
p78-FOER	0.7263	0.7064	0.6070	0.5470	0.2222
p5-GPX1COFOCe	0.7167	0.6970	0.6416	0.5607	0.2613
p29-LMnofb020	0.7193	0.6759	0.5919	0.5553	0.2938
p10-TOPXCOallF	0.7482	0.6657	0.5514	0.4872	0.1923
p25-weightedfi	0.6665	0.6634	0.5915	0.5588	0.2662
p6-inex08artB	0.6689	0.6571	0.5397	0.4941	0.2098
p9-UHelRun394	0.7024	0.6515	0.5555	0.5189	0.2246
p72-UMDFocused	0.7206	0.6386	0.4913	0.3812	0.1111

Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p6-inex08artB	0.6514	0.6379	0.5947	0.5072	0.2255
p56-VSMRIP02	0.7515	0.6314	0.4760	0.3656	0.1396
p5-GPX3COSFOC	0.6232	0.6220	0.5521	0.4617	0.2134
p25-RUCLLP08	0.5969	0.5969	0.5815	0.5439	0.2486
p37-kulcaselem	0.6817	0.5611	0.3525	0.2720	0.1256
p42-B2U0visith	0.6030	0.5360	0.4823	0.4442	0.1736
p16-001RunofUn	0.3110	0.2268	0.1673	0.1206	0.0364

(b) Relevant in Context Task

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p78-RICBest	0.4760	0.3769	0.2975	0.2261	0.2476
p5-GPX1CORICe	0.3946	0.3518	0.2660	0.2156	0.2161
p4-WHOLEDOC	0.3936	0.3492	0.2491	0.1991	0.2114
p10-TOPXCOallA	0.3892	0.3170	0.2339	0.1897	0.1963
p92-manualQEin*	0.3767	0.3370	0.2474	0.1943	0.1920
p6-inex08artB	0.3711	0.3114	0.2274	0.1778	0.1892
p72-UMDRic2	0.3908	0.3412	0.2280	0.1858	0.1740
p12-p8u3exp511	0.3178	0.2855	0.2227	0.1622	0.1673
p48-LIGMLRIC4O	0.3769	0.3384	0.2451	0.1827	0.1581
p56-VSMRIP04	0.2264	0.2005	0.1664	0.1345	0.1263

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p6-inex08artB	0.3799	0.3292	0.2379	0.1831	0.1932
p5-GPX3COSRIC	0.3482	0.3232	0.2381	0.1918	0.1762
p56-VSMRIP05	0.3401	0.2796	0.2133	0.1610	0.1498
p16-009RunofUn	0.0153	0.0156	0.0123	0.0095	0.0023

(c) Best in Context Task

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p78-BICER	0.3883	0.3361	0.2514	0.1931	0.2162
p25-weightedfi	0.3342	0.3016	0.2360	0.1934	0.1992
p5-GPX1COBICp	0.3663	0.3358	0.2494	0.1911	0.1977
p92-manualQEin*	0.3677	0.3357	0.2558	0.2052	0.1939
p10-TOPXCOallB	0.2424	0.2397	0.1769	0.1447	0.1723
p6-submitinex	0.3454	0.3037	0.2248	0.1698	0.1707
p12-p8u3exp501	0.2536	0.2373	0.1924	0.1412	0.1440
p72-UMDBIC1	0.3171	0.2725	0.1746	0.1366	0.1363
p56-VSMRIP09	0.1562	0.1537	0.1377	0.1122	0.1034
p40-xfirmbicco	0.1594	0.1521	0.1357	0.1122	0.0656

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p5-GPX3COSBIC	0.3109	0.2883	0.2235	0.1780	0.1659
p56-VSMRIP08	0.2123	0.1911	0.1481	0.1214	0.1227
p40-xfirmcos07	0.2381	0.1794	0.1348	0.1078	0.0908
p55-KikoriBest	0.1817	0.1721	0.1422	0.1123	0.0803
p16-006RunofUn	0.0307	0.0347	0.0307	0.0261	0.0128

task requirements—most notably non-overlapping results for all tasks, and having scattered results from the same article in relevant in context—are now also considered, and we work with all 163 runs submitted to the Ad Hoc Track.

Table 13 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

Table 13. Top 10 Participants in the Ad Hoc Track: Article retrieval.

Participant	P5	P10	1/rank	map	bpref
p78-BICER	0.6200	0.5257	0.8711	0.3753	0.3693
p92-manualQEin*	0.6371	0.5843	0.8322	0.3601	0.3917
p10-TOPXCOarti	0.5914	0.5386	0.8635	0.3489	0.3620
p5-GPX1COBICe	0.5686	0.5214	0.7868	0.3390	0.3580
p37-kulcoeleme	0.5229	0.4500	0.7468	0.3240	0.3335
p25-weightedfi	0.4914	0.4600	0.7192	0.3224	0.3350
p60-JMUexpe136	0.5429	0.4814	0.7843	0.3173	0.3380
p29-VSMfbElts0	0.5486	0.4800	0.7955	0.3163	0.3385
p9-UHelRun293	0.5714	0.4957	0.7766	0.3113	0.3317
p4-SWKL200	0.5657	0.4943	0.7950	0.3086	0.3292

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). Recall from the above that second ranked run (*p92-manualQEin*) is a manual article retrieval run submitted to all three tasks. Also the run ranked three (*p10-TOPXCOarti*) and the run ranked seven (*p60-JMUexpe136*) retrieve exclusively articles. The relative effectiveness of these article retrieval runs in terms of their article ranking is no surprise. Furthermore, we see submissions from all three ad hoc tasks. Most notably runs from the Best in Context task at ranks 1, 2, 4, and 6; runs from the Focused task at ranks 2, 3, 5, 7, 8, and 9; and runs from the Relevant in Context task at ranks 2 and 10.

If we break-down all runs over the original tasks, shown on the left-hand side of Table 14), we can compare the ranking to Section 4 above. We see some runs that are familiar from the earlier tables: three Focused runs correspond to Table 6, five Relevant in Context runs correspond to Table 7, and seven Best in Context runs correspond to Table 8. More formally, we looked at how the two system rankings correlate using kendall's tau.

- Over all 61 Focused task submissions the system rank correlation is 0.526 between iP[0.01] and map, and 0.574 between MAiP and map.
- Over all 35 Relevant in Context submissions the system rank correlation between MAgP and map is 0.792.
- Over all 40 Best in Context submissions the system rank correlation is 0.787

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 with the Focused task runs is much lower than with the Relevant in Context and Best in Context tasks. This makes sense, since the ranking of articles is an important part of the two "in context" tasks.

6.2 Article retrieval: Clicked pages

In addition to the topics created and assessed by INEX participants, we also included 150 queries derived from a proxy log, and can also construct pseudo-relevance judgments by regarding every clicked Wikipedia article as relevant.

Table 14. Top 10 Participants in the Ad Hoc Track: Article retrieval per task over judged topics (left) and clicked pages (right).

(a) Focused Task

Participant	P5	P10	1/rank	map	bpref	Participant	P5	P10	1/rank	map	bpref
p92-manualQEin*	0.6371	0.5843	0.8322	0.3601	0.3917	p5-Terrier	0.1594	0.0877	0.5904	0.5184	0.8266
p10-TOPXCOarti	0.5914	0.5386	0.8635	0.3489	0.3620	p6-inex08artB	0.1623	0.0870	0.5821	0.5140	0.8150
p5-GPX1COFOCp	0.5686	0.5214	0.7868	0.3390	0.3580	p92-autoindri0	0.1565	0.0884	0.5601	0.4853	0.8211
p37-kulcoeleme	0.5229	0.4500	0.7468	0.3240	0.3335	p60-JMUexpe142	0.1536	0.0862	0.5624	0.4853	0.8250
p78-FOER	0.5714	0.4986	0.7995	0.3230	0.3273	p48-LIGMLFOCRI	0.1449	0.0833	0.5191	0.4596	0.7153
p60-JMUexpe136	0.5429	0.4814	0.7843	0.3173	0.3380	p10-TOPXCOarti	0.1522	0.0841	0.5164	0.4538	0.8167
p25-weightedfi	0.4914	0.4600	0.7192	0.3164	0.3319	p78-FOER	0.1304	0.0819	0.4979	0.4404	0.8136
p29-VSMfbElts0	0.5486	0.4800	0.7955	0.3163	0.3385	p40-xfirmcos07	0.1217	0.0717	0.4301	0.3748	0.7184
p9-UHelRun293	0.5714	0.4957	0.7766	0.3113	0.3317	p55-KikoriFocu	0.1261	0.0732	0.4334	0.3727	0.7785
p6-inex08artB	0.5486	0.4757	0.7851	0.2990	0.3104	p22-EMSEFocuse*	0.1203	0.0783	0.4233	0.3704	0.8105

(b) Relevant in Context Task

Participant	P5	P10	1/rank	map	bpref	Participant	P5	P10	1/rank	map	bpref
p92-manualQEin*	0.6371	0.5843	0.8322	0.3601	0.3917	p5-Terrier	0.1594	0.0877	0.5904	0.5184	0.8266
p5-GPX1CORICp	0.5686	0.5214	0.7868	0.3390	0.3580	p6-inex08artB	0.1623	0.0870	0.5821	0.5140	0.8150
p78-RICBest	0.5800	0.4943	0.8161	0.3371	0.3418	p60-JMUexpe150	0.1536	0.0862	0.5624	0.4853	0.8167
p60-JMUexpe150	0.5857	0.4886	0.8266	0.3107	0.3181	p92-autoindri0	0.1565	0.0884	0.5601	0.4853	0.8211
p10-TOPXCOallA	0.5257	0.4757	0.8226	0.3094	0.3275	p48-LIGMLRIC4O	0.1464	0.0841	0.5238	0.4647	0.7081
p4-SWKL200	0.5657	0.4943	0.7950	0.3086	0.3292	p78-RICBest	0.1348	0.0812	0.4979	0.4422	0.8126
p6-inex08artB	0.5486	0.4757	0.7851	0.2989	0.3104	p10-TOPXCOallA	0.1333	0.0775	0.5139	0.4397	0.7863
p56-VSMRIP05	0.5486	0.4514	0.7752	0.2869	0.3041	p72-UMDRic2	0.1275	0.0717	0.4560	0.4088	0.7526
p72-UMDRic2	0.5971	0.5157	0.8508	0.2715	0.3042	p4-SWKL200	0.1159	0.0732	0.4168	0.3701	0.8007
p22-EMSERICStr*	0.5029	0.4529	0.7079	0.2712	0.3054	p55-KikoriRele	0.1232	0.0710	0.4125	0.3501	0.7712

(c) Best in Context Task

Participant	P5	P10	1/rank	map	bpref	Participant	P5	P10	1/rank	map	bpref
p78-BICER	0.6200	0.5257	0.8711	0.3753	0.3693	p5-Terrier	0.1594	0.0877	0.5904	0.5184	0.8266
p92-manualQEin*	0.6371	0.5843	0.8322	0.3601	0.3917	p6-submitinex	0.1594	0.0862	0.5673	0.4976	0.8164
p5-GPX1COBICe	0.5686	0.5214	0.7868	0.3390	0.3580	p92-autoindri0	0.1565	0.0884	0.5601	0.4853	0.8211
p10-TOPXCOallB	0.5257	0.4757	0.8226	0.3261	0.3340	p60-JMUexpe151	0.1536	0.0855	0.5624	0.4844	0.8214
p25-weightedfi	0.4914	0.4600	0.7192	0.3224	0.3350	p78-BICPRplus	0.1522	0.0841	0.5432	0.4673	0.7799
p60-JMUexpe151	0.5857	0.4886	0.8266	0.3086	0.3178	p10-TOPXCOallB	0.1333	0.0775	0.5139	0.4398	0.8205
p6-submitinex	0.5457	0.4729	0.7793	0.2965	0.3081	p72-UMDBIC1	0.1275	0.0710	0.4482	0.4011	0.7398
p56-VSMRIP08	0.5486	0.4514	0.7752	0.2869	0.3041	p40-xfirmcos07	0.1217	0.0717	0.4301	0.3748	0.7160
p72-UMDBIC2	0.5886	0.5129	0.8440	0.2738	0.3016	p55-KikoriBest	0.1261	0.0732	0.4334	0.3727	0.7785
p12-p8u3exp501	0.4771	0.4343	0.6997	0.2709	0.3058	p56-VSMRIP08	0.1130	0.0659	0.3943	0.3445	0.7258

Table 15 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). Compared to the judged topics, we immediately see much lower scores for the early precision measures (precision at

Table 15. Top 10 Participants in the Ad Hoc Track: Clicked articles.

Participant	P5	P10	1/rank	map	bpref
p5-Terrier	0.1594	0.0877	0.5904	0.5184	0.8266
p6-inex08artB	0.1623	0.0870	0.5821	0.5140	0.8150
p60-JMUexpe150	0.1536	0.0862	0.5624	0.4853	0.8167
p92-autoindri0	0.1565	0.0884	0.5601	0.4853	0.8211
p78-BICPRplus	0.1522	0.0841	0.5432	0.4673	0.7799
p48-LIGMLRIC4O	0.1464	0.0841	0.5238	0.4647	0.7081
p10-TOPXCOarti	0.1522	0.0841	0.5164	0.4538	0.8167
p72-UMDRic2	0.1275	0.0717	0.4560	0.4088	0.7526
p40-xfirmcos07	0.1217	0.0717	0.4301	0.3748	0.7184
p55-KikoriFocu	0.1261	0.0732	0.4334	0.3727	0.7785

5 and 10, and reciprocal ranks), while at the same time higher scores for the overall measures (map and bpref). This is a result of the very low numbers of relevant documents, 1.8 on average, that make it impossible to get a grips on recall aspects. The runs ranked first (*p5-Terrier*), fourth (*p92-autoindri0*), and seventh (*p10-TOPXCOarti*) retrieve exclusively full articles. Again, it is no great surprise that these runs do well for the task of article retrieval.

The resulting ranking is quite different from the article ranking based on the judged ad hoc topics in Table 13. They have only one run in common, although they agree on five of the ten participants. Looking, more formally, at the system rank correlations between the two types of article retrieval we see the following.

- Over all 163 submissions, the system rank correlation is 0.359.
- Over the 76 Focused task submissions, the correlation is 0.363.
- Over the 49 Relevant in task submissions, the correlation is 0.374.
- Over the 38 Best in Context task submissions, the correlation is 0.388.

Hence the judged topics above and the topics derived from the proxy log vary considerable. A large part of the explanation is the dramatic difference between the numbers of relevant articles, with 69.3 on average for the judged topics and 1.8 on average for the proxy log topics.

7 Discussion and Conclusions

In this paper we provided an overview of the INEX 2008 Ad Hoc Track that contained three tasks: 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 three tasks, and analysed the relative effectiveness of element and passage runs, and of keyword (CO) queries and structured queries (CAS). We also look at effectiveness in term of article retrieval, both using the judged topics and using queries and clicks derived from a proxy log.

When examining the relative effectiveness of CO and CAS we found that for all tasks the best scoring runs used the CO query. This is in contrast with earlier results showing that structural hints can help promote initial precision [8]. Part of the explanation may be in the low number of CAS submissions (28) in comparison with the number of CO submissions (108). Only 39 of the 70 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. This may have diminished the value of the CAS query in comparison with earlier years.

Given the efforts put into the fair comparison of element and passage retrieval approaches, the number of passage and FOL submissions was disappointing. Twenty-two submissions used ranges of elements or FOL passage results, whereas 118 submissions used element results. In addition, many of the passage or FOL submissions used exclusively full articles as results. Although we received too few non-element runs to draw clear conclusions, we saw that the passage based approaches were competitive, but not superior to element based approaches. This outcome is consistent with earlier results using passage-based element retrieval [4, 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 still 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 especially for the two "in context" tasks. The systems with the best performance for the ad hoc tasks, also tend to have the best article rankings. When we look at a different topic set derived from a proxy log, and a shallow set of clicked pages rather than a full-blown IR test collection, we see notable differences. Given the low number of relevant articles (1.8 on average) compared to the ad hoc judgments (69.3 on average), the clicked pages focus exclusively on precision aspects. This leads to a different system ranking, although there is still some agreement on the best groups. The differences between these two sets of topics require further analysis.

Finally, the Ad Hoc Track had two main research questions. The first 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. We found that the best performing system used predominantly element results, although the number of non-element retrieval runs submitted is too low to draw any definite conclusions. The second main research question was to compare focused retrieval directly to traditional article retrieval. We found that the best scoring Ad Hoc Track submissions also tend to have the best article ranking, and that the best article rankings were generated using element-level evidence. For both 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] J. P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310. Springer-Verlag, New York NY, 1994.

[2] 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.

[3] L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 40:64–69, 2006.

[4] 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.

[5] 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.

[6] J. Kamps and M. Koolen. On the relation between relevant passages and XML document structure. In *Proceedings of the SIGIR 2007 Workshop on Focused Retrieval*, pages 28–32. University of Otago, Dunedin New Zealand, 2007.

[7] J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. The importance of morphological normalization for XML retrieval. In *Proceedings of the First INEX Workshop*, pages 41–48. ERCIM, 2003.

[8] 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.

[9] 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.

[10] J. A. Thom and J. Pehcevski. How well does best in context reflect ad hoc XML retrieval. In *Pre-Proceedings of INEX 2007*, pages 124–125, 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	151	p4-SWKL200	RiC	CO	Pas	
4	152	p4-WHOLEDOC	RiC	CO	Ele	Article-only
4	153	p4-WHOLEDOCPA	RiC	CO	Pas	Article-only
5	122	p5-Terrier	BiC	CO	Pas	Article-only
5	123	p5-Terrier	Foc	CO	Pas	Article-only
5	124	p5-Terrier	RiC	CO	Pas	Article-only
5	133	p5-GPX2COFOCp	Foc	CO	Pas	Article-only
5	138	p5-GPX1COBICe	BiC	CO	Ele	
5	139	p5-GPX1COFOCe	Foc	CO	Ele	
5	140	p5-GPX1CORICe	RiC	CO	Ele	
5	141	p5-GPX3COSBIC	BiC	CAS	Ele	
5	142	p5-GPX3COSFOC	Foc	CAS	Ele	
5	143	p5-GPX3COSRIC	RiC	CAS	Ele	
5	144	p5-GPX1COBICp	BiC	CO	Pas	Article-only
5	145	p5-GPX1COFOCp	Foc	CO	Pas	Article-only
5	146	p5-GPX1CORICp	RiC	CO	Pas	Article-only
6	255	p6-submitinex	BiC	CO	FOL	Article-only
6	264	p6-inex08artB	RiC	CAS	Ele	
6	265	p6-inex08artB	RiC	CO	Ele	
6	269	p6-inex08artB	RiC	CO	Ele	
6	270	p6-inex08artB	Foc	CAS	Ele	
6	271	p6-inex08artB	Foc	CO	Ele	
6	274	p6-inex08artB	Foc	CO	Ele	
6	276	p6-inex08artB	Foc	CO	Ele	
9	174	p9-UHelRun293	Foc	CO	Ele	
9	176	p9-UHelRun394	Foc	CO	Ele	
10	91	p10-TOPXCOallF	Foc	CO	Ele	
10	92	p10-TOPXCOallB	BiC	CO	Ele	
10	93	p10-TOPXCOallA	RiC	CO	Ele	
10	207	p10-TOPXCOarti	Foc	?	Ele	Article-only
12	97	p12-p8u3exp501	BiC	CO	Ele	
12	100	p12-p8u3exp511	RiC	CO	Ele	
14	205	p14-T2FBCOPARA	Foc	CO	Ele	
16	233	p16-009RunofUn	RiC	CAS	Ele	
16	234	p16-006RunofUn	BiC	CAS	Ele	
16	244	p16-001RunofUn	Foc	CAS	Ele	
22	62	p22-EMSEFocuse	Foc	CO	Ele	Manual Invalid
22	66	p22-EMSEFocuse	Foc	CO	FOL	Manual
22	68	p22-EMSERICStr	RiC	CO	Ele	Manual Invalid
25	30	p25-RUCLLP08	Foc	CAS	Ele	
25	278	p25-weightedfi	Foc	CO	Ele	

Continued on Next Page...

Group	Run	Label	Task	Query	Results	Notes
25	282	p25-weightedfi	BiC	CO	Ele	
29	238	p29-VSMfbElts0	Foc	CO	Ele	
29	253	p29-LMnofb020	Foc	CO	Ele	Article-only
37	227	p37-kulcaselem	Foc	CAS	Ele	
37	230	p37-kulcoeleme	Foc	CO	Ele	
40	54	p40-xfirmbicco	BiC	CO	Ele	
40	296	p40-xfirmcos07	BiC	CAS	Ele	
40	297	p40-xfirmcos07	Foc	CAS	Ele	Invalid
42	299	p42-B2U0visith	Foc	CAS	Ele	
48	59	p48-LIGMLFOCRI	Foc	CO	Ele	
48	72	p48-LIGMLRIC4O	RiC	CO	Ele	
55	279	p55-KikoriFocu	Foc	CAS	Ele	Invalid
55	280	p55-KikoriRele	RiC	CAS	Ele	Invalid
55	281	p55-KikoriBest	BiC	CAS	Ele	
56	190	p56-VSMRIP02	Foc	CAS	Ele	
56	197	p56-VSMRIP04	RiC	CO	Ele	Article-only
56	199	p56-VSMRIP05	RiC	CAS	Ele	Article-only
56	202	p56-VSMRIP08	BiC	CAS	Ele	
56	224	p56-VSMRIP09	BiC	CO	Ele	
60	11	p60-JMUexpe136	Foc	CO	Ele	Article-only
60	53	p60-JMUexpe142	Foc	CO	Ele	
60	81	p60-JMUexpe150	RiC	CO	Ele	Invalid
60	82	p60-JMUexpe151	BiC	CO	Ele	Invalid
72	106	p72-UMDFocused	Foc	CO	Ele	
72	154	p72-UMDBIC1	BiC	CO	Ele	
72	155	p72-UMDBIC2	BiC	CO	Ele	
72	277	p72-UMDRic2	RiC	CO	Ele	
78	156	p78-FOER	Foc	CO	Ele	
78	157	p78-FOERStep	Foc	CO	Ele	
78	160	p78-BICER	BiC	CO	Ele	
78	163	p78-BICPRplus	BiC	CO	Pas	
78	164	p78-RICBest	RiC	CO	Ele	
92	177	p92-autoindri0	BiC	CO	Ele	Article-only
92	178	p92-autoindri0	Foc	CO	Ele	Article-only
92	179	p92-autoindri0	RiC	CO	Ele	Article-only
92	183	p92-manualQEin	BiC	CO	Ele	Manual Article-only
92	184	p92-manualQEin	Foc	CO	Ele	Manual Article-only
92	185	p92-manualQEin	RiC	CO	Ele	Manual Article-only

Exploiting User Navigation to Improve Focused Retrieval

M. S. Ali, Mariano P. Consens, Bassam Helou, and Shahan Khatchadourian

University of Toronto
{sali, consens, bassam, shahan}@cs.toronto.edu

Abstract. A common approach for developing XML element retrieval systems is to adapt text retrieval systems to retrieve elements from documents. Two key challenges in this approach are to effectively score structural queries and to control overlap in the output across different search tasks. In this paper, we continue our research into the use of navigation models for element scoring as a way to represent the user's preferences for the structure of retrieved elements. Our goal is to improve search systems using structural scoring by boosting the score of desirable elements and to post-process results to control XML overlap. This year we participated in the Ad-hoc Focused, Entity Ranking and the Efficiency Tracks, where we focused our attention primarily on the effectiveness of small navigation models. Our experiments involved three modifications to our previous work; (i) using separate summaries for boosting and post-processing, (ii) introducing summaries that are generated from user study data, and (iii) confining our results to using small models. Our results suggest that smaller models can be effective but more work needs to be done to understand the cases where different navigation models may be appropriate.

1 Introduction

At INEX 2008, the University of Toronto investigated the effectiveness of using XML summaries [8] in structural scoring for XML retrieval. An XML summary is a graph-based model that is found by partitioning elements in the collection. By weighting the graph, it represents a navigation model of users traversing elements in their search for relevant information to satisfy their information need. Our use of navigation models was originally developed for use with the performance evaluation measure structural relevance (SR) [5, 2]. SR is a measure of the expected relevance value of an element in a ranked list, given the probability of whether the user will see the element one or more times while seeking relevant information in the higher-ranked results [5]. SR has been shown to be a stable measure that effectively evaluates element, passage, document and tree retrieval systems [2]. Its effectiveness has been validated using navigation models based on either collection statistics or user assessments.

Our search engine uses the Lucene text retrieval system as its basis. Our main adaptation of it is that we index the collection based on XML elements

as documents. Specifically, for INEX 2008, we considered indexes for `article`, `section` and `p` elements. This allowed us to consider the scoring of elements in much the same way as documents would be scored in classical text retrieval. To structurally score elements, we employed boosts corresponding to the label path of the candidate element. In focused retrieval, a significant problem of using Lucene in this way is that it does not prevent overlap (which is known to degrade the quality of results) and so a post-processor is used to control overlap. A key feature of our approach is that both score boosting and post-processing use navigation models in structural scoring.

Our approach to structural scoring involved: (i) using separate and independent navigation models for boosting and post-processing, (ii) introducing navigation models that are generated from user study data, and (iii) focusing on the effectiveness of very small models. In this paper, we show how navigation models have been integrated into structural scoring by using concepts from structural relevance. In particular, we show experimentally how post-processing can be used to not only control overlap, but also to improve the system effectiveness. We do this by presenting three different approaches to post-processing: (a) INEX overlap control where the lowest ranking element in a pair of overlapping elements is removed from the results, (b) Navigation overlap control where the element which was most highly weighted using a pre-selected navigation model is removed from the results, and (c) Ranked list control where the set of elements from the same document that had the highest structural relevance [2] would be included in the system output.

Existing approaches to XML retrieval have relied on rote return structures and ad-hoc tuning parameters for structural scoring of elements. A naive approach assumes that XML documents are structured as articles, and so only logical elements such as articles, sections and paragraphs are returned in the search results. Another approach is to allow users to specify structure, such as using NEXI which is a notation for expressing XML queries that includes structural constraints and hints [15]. NEXI can be used in conjunction with XPATH to retrieve strict XML structural paths according to what the user specifies in the query. Other approaches to structural retrieval, like XRANK [9] or Clarke's Re-ranking Algorithm [7], use element weighting schemes to iteratively score and re-rank results to improve the final system output. In this work, we rely on a probabilistic model of navigation developed for the evaluation measure structural relevance. Other models exist, most notably the model that underlies the PRUM evaluation measure [12].

We first investigated the effectiveness of using XML summaries as a way to introduce structural scoring into XML retrieval at INEX 2007 in the Thorough Ad Hoc Track in element retrieval [3]. Our initial approach allowed complex modeling of user navigation using large models that were derived solely from collection statistics in XML summaries (such as element length, element depth and element label paths). In this work, we greatly extend this work, and, in particular, focus on the effectiveness of using smaller models, and weighting schemes that are based on user assessments.

The paper is structured as follows. In Section 2, we review the preliminary concepts of the information seeking behaviour of users and structural relevance. In Section 3, we present the navigation models that we used in INEX 2008. In Section 4, we present the XML retrieval system that was used in this work. In Section 5, we present our results for INEX 2008. Finally, in Section 6 we discuss our findings and future work.

2 Preliminary Concepts

In structural scoring in XML retrieval, the key challenge is to differentiate among candidate elements those that meet the user's preference for elements whose structure supports how they fulfill their information need or those that minimize the redundancy that a user will experience while seeking for relevant information from search results. The goal of a focused system is to retrieve non-overlapping elements (or passages) that contain relevant information. For INEX 2008, only focused results, which contain no overlapping elements, are considered. We show how the more general concept of redundancy in structural relevance can be used as an effective way to address problems with overlap in XML retrieval [10] by providing a means to structurally score XML elements based on user preferences and XML element structural characteristics.

In this work, the redundancy between elements is measured using structural relevance and found using different navigation models of how users experience redundancy while browsing for relevant information in retrieved documents. In Section 2.1, the information seeking behaviour of users fulfilling an information need is presented. This is followed by Section 2.2 where structural relevance and its underlying probabilistic model of redundancy based on user navigation is presented.

2.1 Information Seeking Behaviour

In element retrieval, we assume that users *consult* the system output going from one rank to the next, *visiting* the element in context [2]. The user stops consulting the output when their information need has been satisfied. A visited element is considered to have been *seen* by the user. After seeing an element, the user may visit additional elements by *browsing* out of the element into the rest of its parent XML document. This process of visiting, browsing and seeing content (in elements) within documents to seek relevant information is called *user navigation*. During this process the user may encounter content in already seen elements, and, thus, redundant content. If the user tolerates redundant content, then the relevant content that is seen a number of times remains relevant to the user. If the user does not tolerate redundant content, then relevant content is considered non-relevant if already seen. In this work, we assume that the user does not tolerate (redundancy) seeing content more than once. In structural relevance, overlap is a special-case of redundancy where overlapping elements are on the same XML branch in the same document instance [5].

2.2 Structural Relevance

Structural relevance (SR) [2] is a measure of the relevance of the results given that the user may find some of the results in the output redundant. Here, we present structural relevance for elements (although, more generally, these concepts hold for trees and passages). We define the redundancy of an element to a user as whether the content of the element has been previously seen by the user from the search results. We measure it by finding the probability of whether the content has been seen more than once while the user was seeking to fulfill their information need, given the search results and how the user navigates between elements within the collection to find relevant information.

SR depends on the user's browsing history, which is the set of elements that a user has viewed. We calculate SR as the expected relevance value of a ranked list given that the user does *not* find the results redundant:

$$SR(R) = \sum_{i=1}^{k} rel(e_i) \cdot (1 - p(e_i; R[e_{i-1}])) \tag{1}$$

where the system output $R = \{e_1, e_2, \ldots, e_k\}$ is a ranked list of k elements, the browsing history $R[e_{i-1}] = \bigcup_{j=1}^{i-1} e_j$ is the set of elements that are ranked higher than the i-th element $e_i \in R$, $rel(e_i) \in [0, 1]$ is the relevance value of element e_i, and the redundancy $p(e_i; R[e_{i-1}])$ is the probability that the element e_i is seen more than once by the user.

In SR, redundancy of an element is defined as the probability that a user has seen an element given that they may have navigated to it from a previously seen element. The following definition formalizes the probabilistic model of user navigation that underlies redundancy in SR.

User Navigation. The probability that the content in an element is seen by a user while visiting a different element is

$$p(e; f) = P(e \text{ seen} | \text{Visit to } f) \tag{2}$$

If e and f are from two different documents then e is not seen and $p(e; f) = 0$; if $e = f$ then e is seen with certainty and $p(e; f) = 1$; otherwise, e and f are from the same document, so e is seen with some probability $0 \leq p(e; f) \leq 1$.

The redundancy of the element e in the ranked list R when the user does not tolerate seeing redundant content is $p(e_i; R[e_{i-1}]) = \prod_{j=1}^{i-1} 1 - p(e_i; e_j)$. In [2], it was shown how User Navigation (above) can be simplified to $p(e; f) = \pi_{(e)}$, the steady-state probability that the user will see e, if the user can navigate to every element in a document from any other element in the document (where $e \neq f$, e and f are from the same document, and e has not been previously seen by the user).

We model user navigation using a navigation graph that represents documents in the collection. Each node in the graph corresponds to a partition of the elements in the collection. Every element in the collection is included in a partition in the navigation graph (which, as an aside, is precisely an XML summary). To calculate the probabilities for user navigation, weights are ascribed to

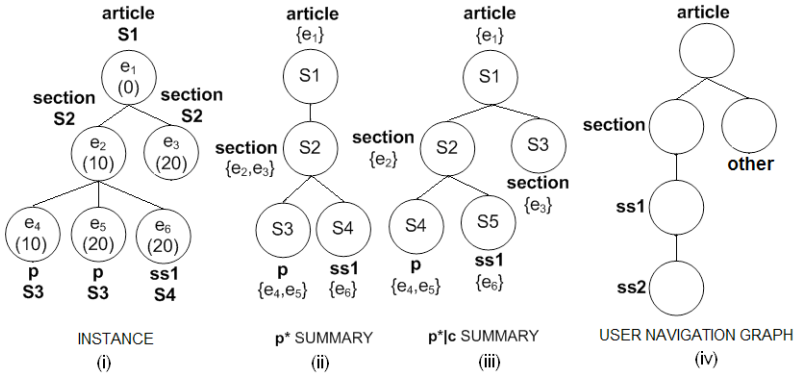


Fig. 1. Examples of (i) Document instance, (ii) p^* summary, (iii) $p^*|c$ summary, and (iv) user navigation graph.

the edges (or paths) of the navigation graph. These weights are then normalized across each row, and the resultant transition matrix represents the user navigation probabilities in Equation 2. To find the steady-state probabilities $\pi_{(e)}$, the matrix is iteratively multiplied with itself until all rows are equal [13] (pp. 200–213).

In this section, we presented information seeking behaviour and structural relevance. In the next section, we present different scenarios for creating the XML summary of the collection, and weighting the resultant navigation graph to determine the steady-state probabilities that we can use for structural scoring of elements.

3 Proposed Navigation Models for INEX 2008

Navigation models represent how users navigate between elements while seeking relevant information from structured search results. In structural scoring, we use these models to both boost candidate retrieval elements and for post-processing ranked lists to control overlap. In this section, we provide the background to understand what is entailed in a navigation model and the navigation models considered in INEX 2008.

XML structural summaries (referred to as summaries) provide a way to represent navigation models based on the structure of documents in the collection. We refer to these models as summary navigation models. Summaries are graphs representing relationships between sets of document elements with a common structure (paths, subtrees, etc.). For instance, AxPRE summaries [8] define a broad range of the different summaries available in the literature. They are created using an axis path regular expression language that is capable of describing a plethora of partitioning schemes. For example, a p^* summary partitions XML elements based on their incoming paths, since p^* is the axis path regular ex-

pression describing paths of parent (p) axis traversals. Similarly, a $p^*|c$ summary is the axis path regular expression describing paths of parent (p) with a single child (c) axis traversals. Figure 1 shows an example Wikipedia article instance in (i), its p^* summary in (ii), and its $p^*|c$ summary in (iii). The elements in the summary partitions are called the *extent* of the summary partition.

Summary navigation models are weighted via collection statistics of child nodes that are used to bi-directionally weight the edges (as opposed to paths) in the navigation graph. Examples of different weighting statistics include content length, extent size, and element depth. Content length weights are the number of characters of content in the elements in the extent of the child nodes. Extent size is the number of elements in the extent of a child node. Finally, depth weights are the same as content but damped (divided) by the path depth of the elements in the extent of the child nodes. Using the methodology for finding $pi_{(e)}$ described in the previous section, and 2343 randomly selected Wikipedia articles summarized using a p^* summary and whose partitions were mapped to the user navigation graph shown in Figure 1(iv), we get Table 1C which shows summary navigation models for Wikipedia based on path, content and depth weights.

Navigation models can also be generated from user assessments. We refer to these models as user navigation models. In [6], assessments from the INEX 2006 Interactive Track user study were used to produce user navigation models. The INEX 2006 user study consisted of 83 participants for 12 assessment topics with user activity recorded for 818 documents from the INEX 2006 Wikipedia collection [11]. Figure 1(iv) shows the five types of XML elements that participants visited in the 2006 user study; namely, `ARTICLE`, `SEC`, `SS1`, `SS2`, and `OTHER`. These correspond to elements whose label paths are the root */article* (`ARTICLE`), a section path */article/body/section* (`SEC`), a subsection path `SEC`*/section* (`SS1`), a sub-subsection path `SS1`*/section* (`SS2`), and all other elements' paths (`OTHER`). We call Figure 1(iv) the user navigation graph.

Table 2 tabulates the observed visits and mean time spent in visits for element assessments by participants in the INEX 2006 user study. For instance, participants visited `SS2` elements and then navigated to element `ARTICLE` 4 times. The mean time spent in `SS2` before navigating to element `ARTICLE` was on average 12.3 seconds. This led to an overall time, which we refer to as an episode, of 12.3 x 4 = 49.2 seconds. The most visited element was `SEC`, and the largest mean time spent occurred in navigations to `SEC` elements from `ARTICLE`. These assessments of user navigation can be used to weight the paths in the user navigation graph in Figure 1(iv). The resultant navigation probabilities $p(e; f)$ for the user navigation model, based on normalizing the number of visits in Table 2, are shown in Table 1A. Similarly, we can generate user navigation models (based on the same user navigation graph) for the observed time-spent and episodes. The user navigation models developed in [6] are shown in Table 1B.

Additionally, we investigated the use of trivial navigation models that were composed of two nodes; a main node that would contain one type of element and the other node which would include all other elements in the collection. The main node has a high steady-state probability (we used 0.999) and the

A. Normalized Weights for Visits

	Destination				
Source	ARTICLE	SEC	SS1	SS2	OTHER
ARTICLE	0.0	0.87	0.11	0.01	0.01
SEC	0.40	0.54	0.06	0.0	0.0
SS1	0.31	0.34	0.0	0.0	0.01
SS2	0.21	0.11	0.68	0.0	0.0
OTHER	0.58	0.0	0.08	0.0	0.33

B. User Navigation Models

	ARTICLE	SEC	SS1	SS2	OTHER
Visit	0.281	0.606	0.105	0.002	0.006
Episode	0.410	0.531	0.050	0.001	0.009
Time spent	0.318	0.209	0.129	0.028	0.317

C. Summary Navigation Models

	ARTICLE	SEC	SS1	SS2	OTHER
Path	0.361	0.537	0.087	0.014	0.001
Content	0.103	0.434	0.089	0.013	0.361
Depth	0.309	0.435	0.067	0.008	0.181

Table 1. Visit user model transition matrix.

	Destination				
Source	ARTICLE	SEC	SS1	SS2	OTHER
ARTICLE	0 (0)	138 (100.4)	18 (48.7)	1 (22)	2 (76)
SEC	278 (57.0)	372 (14.7)	41 (11.3)	0 (0)	0 (0)
SS1	46 (13.1)	50 (10.2)	50 (9.52)	0 (0)	1 (48)
SS2	4 (12.3)	2 (264.5)	13 (5.3)	0 (0)	0 (0)
OTHER	7 (27.7)	0 (0)	1 (4)	0 (0)	4 (26)

Table 2. Number of visits (mean time spent)

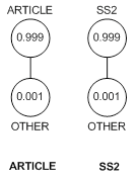


Fig. 2. Trivial navigation models used in INEX 2008

other node would have a correspondingly small steady-state probability (0.001). We proposed two trivial models; namely the article model and the ss2 model. These have the effect of an exclusion filter in that the elements in the main node will be less preferred in the results than other nodes. The proposed trivial navigation models are shown in Figure 2. For INEX 2008, five navigation models were considered: summary navigation models path and depth (Table 1C), user navigation model visit (Table 1B), and, the article and ss2 trivial navigation models (Figure 2).

In this section, we have presented how different navigation models are generated, and presented the 5 navigation models that we used in INEX 2008. In the next section, we present a description of our search system, how we implemented boosts, and a description of the different post-processors that we used in INEX 2008.

4 System Description

This section provides details on how we implemented our search engine, which is based on Apache Lucene.

4.1 Lucene

The p^* structural summary of the Wikipedia XML collection, originally generated using code from DescribeX [4], consisted of 55486 summary nodes (with aliasing on tags containing the substrings `link`, `emph`, `template`, `list`, `item`, or `indentation`). The extents in the structural summary were then mapped to the navigation graph shown in Figure 1(iv) to produce the summary navigation models. As the collection was summarized, modified Apache Lucene [1] code was used to index the tokens. The posting list also included character offsets. Tokens not excluded from the stop word filter had punctuation symbols removed and the tokens maintained their case. The structural summary was generated at the same time as each document was indexed, and the payload information for each token occurrence included the summary partition in which the token appears.

To accommodate the structural hints in the INEX topics, separate indexes were built for each tag identified by the structural hint present within the set of INEX topics which included “article”, “section”, and “p”. For example, building

an index for the "p" tag would index the first "p" element and its children, including nested "p" elements, until its respective closing tag. Thus, a file with multiple non-overlapping indexed elements will create multiple documents within the index, and these non-overlapping elements are easily identified since the index stores the character offsets as previously mentioned. This results in having element-level documents which allows the calculation of idf scores for terms within elements. Table 3 shows the index sizes (which includes term frequencies, file paths and the payload information).

Tag	Size
article	6.07GB
section	4.84GB
p	4.63GB

Table 3. Index sizes using tag-level documents

Lucene's query parser was adapted to accept Top-X [14] NEXI queries with structural hints. The queries were encoded using boolean operators to represent tokens that were mandatory, optional, or to be excluded. Double quotes indicating adjacent tokens were removed since token positions were not indexed. Prior to running a query, the query was examined for any structural hints and the required indexes were searched as a single merged index using Lucene's regular application interface. If no structural hints were identified, the complete set of element indexes were used in the search.

In the Content Only (CO) sub-task, queries are specified using keywords and content-related conditions. Structure can be included in the query as hints to reduce the number of returned elements. CO queries with structural hints are called Content Only + Structure (CO+S) queries. For CO+S queries, it is left to the discretion of the search engine to interpret it as either strict or vague. Our system used the element labels in structural hints to include or exclude specific search indexes while processing the query. Queries were not explicitly interpreted as either strict or vague. The score of elements was composed of two main factors; (i) content relevance, and (ii) a score boost based on the label path of the smallest element (Section 4.2) that enclosed the content. The highest scoring elements from Lucene were then post-processed (Section 4.3) to ensure that the final results returned were focused.

4.2 Boosting Strategies

The collection was indexed at the element-level for `article`, `section`, and `p`. In our experiments, we included runs with score boosting per term occurrence and using the average of the term scores as a modifier to Lucene's original document score. The boost used was the stationary probability $\pi_{(e)}$ of the partition in the summary of the element in which the term occurs. The baseline payload score

per occurrence was set to 1 and the boosted term score was the baseline plus the stationary probability. The scenarios reported in this paper include runs with either no boosting, boosted using the path summary navigation model (Table 1B), or boosted using the visit user navigation model (Table 1C).

4.3 Post-processing Algorithms

The purpose of post-processing is to control overlap and produce focused runs from Lucene's results. We present three different approaches to post-processing. The first approach, called INEX Overlap Control and shown in Figure 3, removes all parent-child overlap from the ranked list by comparing each pair of elements e_i and e_j in R, and removing the lower-ranked element if they are on the same branch.

The second approach, called Navigation Overlap Control and shown in Figure 4, involves removing parent-child overlap where overlapped elements were ranked closely to one another (in the results reported here, the window size was set to 10 rank positions). Instead of removing the lowest ranked overlapped element, the element with the highest steady-state probability was removed from the ranked list. This is akin to removing the element most likely to be visited by an element within the same document using any path.

The third approach, called Ranked List Overlap Control and shown in Figure 5, was developed in INEX 2007 and involves computing SR for scenarios where redundant elements (i.e., from the same document) are systematically removed or reordered in the ranked list until the highest scoring scenario is found. We assume that all redundant elements are relevant and place the restriction that a document cannot be excluded from the ranked list by removing all of its elements that were present in the original result list.

So, the post-processing of overlap used either; (i) a simple heuristic to remove the lowest ranked elements that were overlapped; (ii) a more complex heuristic to remove the most redundant overlapped elements; or (iii) an algorithm to find the most structurally relevant non-overlapped set of elements. To determine redundancy and structural relevance in post-processing, we used four navigation models; namely, a trivial navigation model based on article elements being redundant, a second trivial navigation model based on sub-subsection elements being redundant, a depth summary navigation model, and a path user navigation model.

5 Results INEX 2008

In this section, we present our results from our participation in INEX 2008 in the Ad-Hoc Focused Track (Section 5.1), the Efficiency Track (Section 5.2), and the Entity Ranking Track (Section 5.3).

Algorithm *INEX Overlap Control*
Input: Ranked list $R = e_1, e_2, \ldots$ of k elements.
Output: Ranked list with overlapped elements removed R^*

```
 1: let m be the length of R*
 2: m = 1
 3: for i = 1 to k do
 4:    skip = false
 5:    for j = 1 to i − 1 do
 6:       if e_i and e_j are on same branch then
 7:          skip = true
 8:       end if
 9:    end for
10:    if skip = false then
11:       R*[m] = e_i
12:       m = m + 1
13:    end if
14: end for
```

Fig. 3. Remove lowest ranked overlapped elements from a ranked list

Algorithm *Navigation Overlap Control*
Input: Ranked list $R = e_1, e_2, \ldots$ of k elements.
Input: Summary graph S with navigation $\pi_{(e)}$ for element e.
Output: Ranked list with overlapped elements removed R^*

```
 1: let window be the minimum distance between competing elements
 2: let m be the length of R*
 3: m = 0
 4: for i = 1 to k do
 5:    skip = false
 6:    for j = 1 to k do
 7:       if e_i and e_j are on same branch then
 8:          if |i − j| > 10 then
 9:             skip = true
10:          else
11:             if πe_i > π_{e_j} then
12:                skip = true
13:             end if
14:          end if
15:       end if
16:    end for
17:    if skip = false then
18:       R*[m] = e_i
19:       m = m + 1
20:    end if
21: end for
```

Fig. 4. Remove the least isolated overlapped elements

Algorithm *Ranked List Overlap Control*
Input: Ranked list $R = e_1, e_2, \ldots$ of k elements.
Input: Summary graph S
Output: Ranked list $R^{'}$ in Ω
Output: Ranked list in Ω with highest SRP R^{sr}
Output: Ranked list with overlapped elements removed R^{*}
1: let n be number of overlapped elements in R
2: let Ω be the scenarios of R
3: **for** $R^{'} \in \Omega$ **do**
4: **if** $SR(R^{'})/k > SR(R^{*})/k$ AND **then**
5: $R^{sr} = R^{'}$
6: **end if**
7: **end for**
8: R^{*} = INEX List Control(R^{sr})

Fig. 5. Find the highest scoring scenario.

5.1 Ad-hoc Focused Element Retrieval Content-Only Sub-Task

We submitted results for the Ad-hoc Focused Track using element retrieval in the Content-Only Sub-Task. We submitted 3 runs: B2U0_visit-heur, B2U0_tiny-path-sr, and B2U0_tiny-path-heur. Runs B2U0_visit-heur and B2U0_tiny-path-heur were boosted using the navigation models visit (Table 1B) and path (Table 1C), respectively. Both of these runs were post-processed using INEX overlap control. They showed similar performance. Unfortunately, the B2U0_tiny-path-sr run (which was boosted and post-processed using the path summary navigation model in Table 1C), was not admissible in the focused task due to overlap in the run (because of a syntax error in the ranked list overlap control post-processor code). Our results reported here show an extended set of runs that are indicative of the effectiveness of our proposed approach to structural scoring, and we rename the runs B2U0_visit-heur and B2U0_tiny-path-heur to PATH INEX NONE and VISIT INEX NONE, respectively in Table 4 and Figure 6.

The runs reported here are top-100 results for the Wikipedia collection across 235 topics in INEX 2008 evaluated using the official INEX measures (inex_eval) MAiP and (inex_eval) interpolated precision across interpolated recall points. The purpose of these runs was to investigate empirically whether there existed predictable combinations of boosting and post-processing that would result in more effective systems.

In Table 4, we show the MAiP evaluations for all tested configurations. The configuration for each run consisted of the type of navigation model used to boost results (Boost), the approach used to post-process the run to remove overlap (Overlap), and the navigation model used by the approach for removing overlap (Navigation). For instance, our best run was first boosted using the visit user navigation model, and then post-processed with the depth summary navigation model. The NAV runs (navigation overlap control using the algorithm shown in Figure 4) did not perform well (a maximum MAiP of 0.102 with

Boost	Overlap	Navigation	MAiP
NONE	INEX	NONE	0.111
NONE	NAV	DEPTH	0.0924
NONE	SR	ARTICLE	0.0685
NONE	SR	DEPTH	0.130
NONE	SR	SS2	0.10172
NONE	SR	VISIT	0.0817
PATH	INEX	NONE	0.115
PATH	NAV	DEPTH	0.107
PATH	SR	ARTICLE	0.0745
PATH	SR	DEPTH	0.139
PATH	SR	SS2	0.106
PATH	SR	VISIT	0.080
VISIT	INEX	NONE	0.123
VISIT	NAV	DEPTH	0.102
VISIT	SR	ARTICLE	0.0723
VISIT	SR	DEPTH	**0.145**
VISIT	SR	SS2	0.116
VISIT	SR	VISIT	0.089

Table 4. Mean-average interpolated precision using HiXEval for INEX 2008 Focused Runs (k=100, 235 topics)

boosting using the visit user navigation model was observed). In INEX 2007 [3], we observed that post-processing with the depth summary navigation model improved the effectiveness of systems. This was a full summary of the collection with a navigation graph that consisted of 55486 nodes, as opposed to the smaller model of only 5 nodes used this year. Moreover, we note that regardless of the boost, the best overall post-processor (by MAiP) was the depth summary navigation model. Additionally, we observed that, for each boost (NONE, PATH, VISIT), the relative performance of the post-processor configurations (Overlap-Navigation pairs in Table 4) was consistent, and was (listed from best configuration to worst) SR-DEPTH $\succ$ INEX-NONE $\succ$ SR-SS2 $\succ$ NAV-DEPTH $\succ$ SR-VISIT $\succ$ SR-ARTICLE.

In Figure 6, we show the interpolated I-R curves for the six best runs reported in Table 4. The runs were within +/-0.05 precision of each other across recall points. Using MAiP, significant differences in overall performance were observed; the depth summary navigation model and INEX overlap control consistently performed better than the other configurations. From these results (and observations from INEX 2007 using large models), it seems that system effectiveness can be improved by using separate summaries for boosting and post-processing. Moreover, we observed similar performance within the small models (specifically that the depth model out-performed other models) as in large models, suggesting that the size of the model is not as important as the type of model used. Finally, these preliminary results suggest that boosting is effective, and that boosting with a user navigation model is more effective than using summary navigation

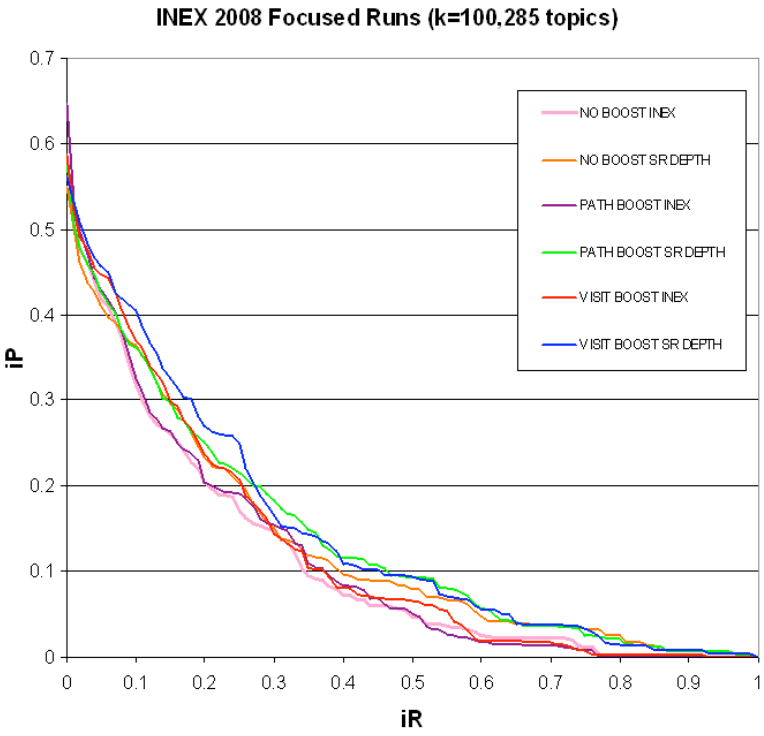


Fig. 6. Interpolated precision-recall in Focused Task using summary boosts, INEX overlap heuristics, and optimal structural relevance.

models. It remains to be seen whether these observations can be generalized across search tasks.

5.2 Efficiency Track

The Efficiency Track is a forum for the evaluation of both the effectiveness and efficiency of XML ranked retrieval approaches. For INEX 2008, the track consists of 568 queries for which participants provide timed runs. By query type, our results were:

(A) 540 Queries Ad-Hoc-style, average time is 3166 ms with query times varying from 15 to 26063 ms.

(B) 21 Queries High-dimensional content retrieval, average time is 8460 ms, with query times varying from 188 to 61141 ms.

(C) 7 Queries High-dimensional structure retrieval. average time is 12090 ms, with query times varying from 2948 to 35328 ms.

Figure 7 shows our overall performance results. The figure is a histogram of query execution times for all 568 queries. It shows that most queries took under 2 seconds to execute. The median of query times is 1703 ms. Nonetheless, a sizable proportion (about 16%) of the queries took more than 5 seconds to execute. The query execution time included two main components. First, the time required by Lucene to interpret the query, retrieve candidate elements from one or more indexes and return back a posting list with thorough results. Second, the time to post-process the results to output focused results. Our system was run on Windows XP in a virtual machines using VMWare GSX on a Sun Fire V20z Server cluster running on Red Hat Enterprise Linux. The VM was configured with 2048 MB of RAM and one virtual 2.39 GHz cpu running over an AMD opteran ^{TM}Processor 250. Our results are biased to some extent due to virtualization issues such as the slow I/O of virtual disks, and poor thread synchronization in VMWare for real-time applications. Nevertheless, for 35 topics, our system needed more than 10s. The slow queries occurred across all query types. Although, further research is needed to isolate both the queries and the characteristics of the queries that caused our system to process them inefficiently, we hypothesize that it was the processing of popular terms that slowed down our system.

5.3 Entity Ranking

The Entity Ranking Track uses the Wikipedia data, where systems may exploit the category metadata associated with entities in entity retrieval. For example, consider a category "Dutch politicians". The relevant entities are assumed to be labelled with this category or other closely related category in the categorization hierarchy, e.g. "politicians".

Our participation involved developing 6 new topics and conducting assessments. The results for this track are still pending, and, thus, we leave this discussion for future work.

6 Conclusions

In this work, we have shown how navigation models can be used effectively in both element score boosting and in the post-processing of overlap. The models can be derived either from collection statistics or user assessments. The most significant observation in this work is that small models based on either assessments or collection statistics can be used. Our results in INEX 2007 suggested that the depth summary navigation model was a good model to use for post-processing. In this study, our results have corroborated with the observations in INEX 2007, but, importantly we have shown that smaller models can be used. Small models

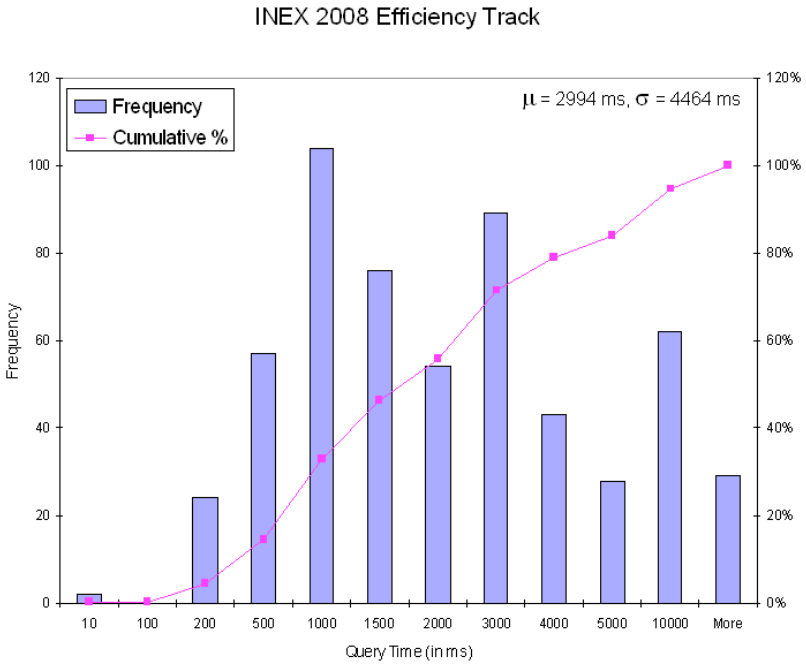


Fig. 7. Histogram of query times in the INEX 2008 Efficiency Track

are more easily interpreted and more efficient to use in our computations. In the future, we hope to generalize this methodology for structural scoring as a way to test and compare how different search engines handle different structural constraints and hints.

References

1. Apache Lucene Java. `http://lucene.apache.org`, 2008.
2. M. S. Ali, M. P. Consens, G. Kazai, and M. Lalmas. Structural relevance: a common basis for the evaluation of structured document retrieval. In *CIKM 2008*, pages 1153–1162, New York, NY, USA, 2008. ACM.
3. M. S. Ali, M. P. Consens, and S. Khatchadourian. XML retrieval by improving structural relevance measures obtained from summary models. In INEX 2007. *LNCS 4862*, pages 34–48, 2007.
4. M. S. Ali, M. P. Consens, S. Khatchadourian, and F. Rizzolo. DescribeX: Interacting with AxPRE Summaries. In *ICDE 2008*, pages 1540–1543. IEEE, 2008.
5. M. S. Ali, M. P. Consens, and M. Lalmas. Structural Relevance in XML Retrieval Evaluation. In *SIGIR 2007 Workshop on Focused Retrieval*, pages 1–8, 2007.
6. M. S. Ali, M. P. Consens, and B. Larsen. Representing user navigation in XML retrieval with structural summaries. (submitted for acceptance). In *ECIR 2009*.
7. C. Clarke. Controlling overlap in content-oriented XML retrieval. In *SIGIR 2005*, pages 314–321, New York, NY, USA, 2005. ACM Press.

8. M. P. Consens, F. Rizzolo, and A. A. Vaisman. AxPRE Summaries: Exploring the (Semi-)Structure of XML Web Collections. In *ICDE 2008*, pages 1519–1521, 2008.
9. L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked keyword search over xml documents. In *SIGMOD 2003*, New York, NY, USA, 2003. ACM Press.
10. G. Kazai, M. Lalmas, and A. P. de Vries. The overlap problem in content-oriented xml retrieval evaluation. In *SIGIR 2004*, pages 72–79, New York, NY, USA, 2004. ACM.
11. S. Malik, A. Tombros, and B. Larsen. The Interactive Track at INEX2006. In INEX 2006. *LNCS 4518*, pages 387–399, 2007.
12. B. Piwowarski, P. Gallinari, and G. Dupret. Precision recall with user modeling (PRUM): Application to structured information retrieval. *ACM Trans. Inf. Syst.*, 25(1):1, 2007.
13. S. M. Ross. *Introduction to Probability Models*. Academic Press, New York, 8th edition, 2003.
14. M. Theobald, R. Schenkel, and G. Weikum. An efficient and versatile query engine for TopX search. In *Proc. VLDB Conf.*, pages 625–636, 2005.
15. A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In INEX 2004. *LNCS 3493*, pages 34–48, 2005.

Proximity-Aware Scoring for XML Retrieval

Andreas Broschart[1,2], Ralf Schenkel[1,2], and Martin Theobald[1]

[1] [1]Max-Planck-Institut für Informatik, Saarbrücken, Germany
[2] [2]Saarland University, Saarbrücken, Germany
{abrosch,schenkel,mtb}@mpi-inf.mpg.de

Abstract. Proximity enhanced scoring models significantly improve retrieval quality in text retrieval. For XML IR, we can sometimes enhance the retrieval efficacy by exploiting knowledge about the document structure combined with established text IR methods. This paper elaborates on our approach used for INEX 2008 which modifies a proximity scoring model from text retrieval for usage in XML IR and extends it by taking the document structure information into account.

1 Introduction

Term proximity has been a common means to improve effectiveness for text retrieval, passage retrieval, and question answering, and several proximity scoring functions have been developed in recent years (for example, [4–7]). For XML retrieval, however, proximity scoring has not been similarly successful. To the best of our knowledge, there is only a single existing proposal for proximity-aware XML scoring [1] that computes, for each text position in an element, a fuzzy score for the query, and then computes the overall score for the element as average score over all its positions.

Our proximity score for content-only queries on XML data [2] extends the existing proximity score by Büttcher et al. [4], taking into account the document structure when computing the distance of term occurrences.

2 Proximity Scoring for XML

To compute a proximity score for an element e with respect to a query $q = \{t_1 \dots t_n\}$ with multiple terms, we first compute a linear representation of e's content that takes into account e's position in the document, and then apply a variant of the proximity score by Büttcher et al. [4] on that linearization.

Figure 1 shows an example for the linearization process. We start with the sequence of terms in the element's content. Now, as different elements often discuss different topics or different aspects of a topic, we aim at giving a higher weight to terms that occur together in the same element than to terms occurring close together, but in different elements. To reflect this in the linearization, we introduce virtual gaps at the borders of certain elements, whose sizes depend on the element's tag (or, more generally, on the tags of the path from the document's root to the element). In the example, gaps of `section` elements may

be larger than those of `p` (paragraph) elements, because the content of two adjacent `p` elements within the same `section` element may be considered related, whereas the content of two adjacent `section` elements could be less related. Some elements (like those used purely for layout purposes such as `bold` or for navigational purposes such as `link`) may get a zero gap size. The best choice for gaps depends on the collection. Gap sizes are currently chosen manually; an automated selection of gap sizes is subject to future work.

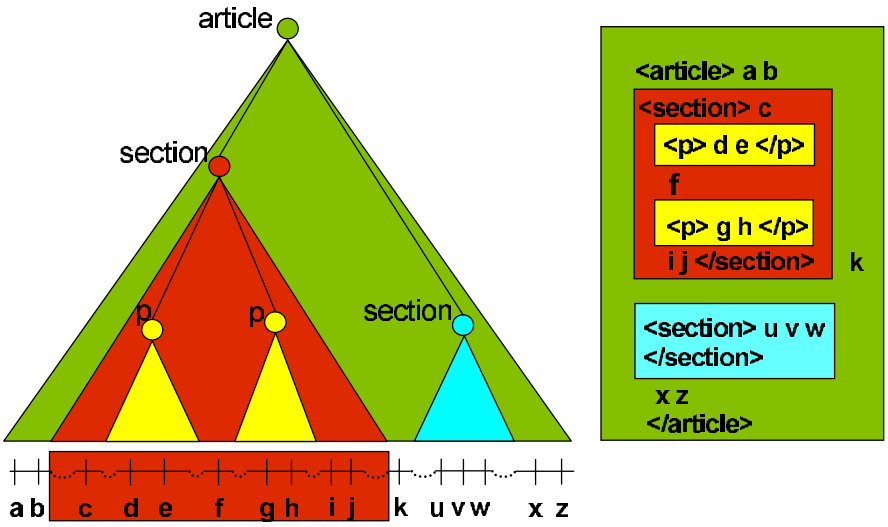


Fig. 1. An XML document and its linearization

Based on the linearization, we apply the proximity scoring model of Büttcher et al. [4] for each element in the collection to find the best matches for a query $q = \{t_1, \ldots, t_n\}$ with multiple terms. This model linearly combines, for each query term, a BM25 content score and a BM25-style proximity score into a proximity-aware score. Note that unlike the original, we compute these scores for elements, not for documents, so the query-independent term weights in the formulas are inverse *element* frequencies $ief(t) = \log_2 \frac{N - ef(t) + 0.5}{ef(t) + 1}$, where N is the number of elements in the collection and $ef(t)$ is the number of elements that contain the term t. Similarly, average and actual lengths are computed for elements. The BM25 score of an element for a query is

$$score_{\text{BM25}}(e, q) = \sum_{t \in q} ief(t) \frac{tf(e,t) \cdot (k_1 + 1)}{tf(e,t) + K}$$

To compute the proximity part of the score, Büttcher et al. first compute an accumulated interim score $acc(t_i)$ for each query term t_i that depends on the distance of this term's occurrences in the element to other, adjacent query term occurrences. Formally, for each adjacent occurrence of a term t_j at distance d to an occurrence of t_i, $acc(t_i)$ grows by $ief(t_j)/d$. The proximity part of an

element's score is then computed by plugging the *acc* values into a BM25-style scoring function:

$$score_{prox}(e,q) = \sum_{t \in q} min\{1, ief(t)\} \frac{acc(t) \cdot (k_1+1)}{acc(t)+K}$$

where, $K = k \cdot [(1-b) + b \cdot \frac{|e|}{avgel}]$ (analogously to the BM25 formula) and b, k_1, and k are configurable parameters that are set to $b = 0.5$ and $k = k_1 = 1.2$, respectively. The overall score is then the sum of the BM25 score and the proximity score:

$$score(e,q) = score_{\text{BM25}}(e,q) + score_{prox}(e,q)$$

3 AdHoc Track Results

3.1 Results for Focused Task

Our recent development of TopX focused on improving its retrieval quality. For the Focused Task, we submitted the following three runs:

- `TopX-CO-Baseline-articleOnly:` a CO run that considered the non-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 $score_{\text{BM25}}$ value as described in Section 2. Note that this approach corresponds to standard document-level retrieval.
- `TopX-CO-Proximity-articleOnly:` a CO run that reranked the results of the `TopX-CO-Baseline-articleOnly` run by adding the $score_{prox}$ part described in Section 2. We used gaps of size 30 for `section` and `p` elements. (Due to the limited number of runs we could not evaluate different gap sizes; see [2] for a more thorough study with older INEX topics.)
- `TopX-CO-Focused-all:` an element-level CO run that considered the terms in the title of a topic without phrases and negations, allowing all tags for results. Note that, unlike our runs in previous years, we did not use a tag-specific ief score, but a single global ief value per term; we demonstrated in [3] that this gives better results for CO queries than tag-specific inverse element frequencies.

run	**iP[0.00]**	**iP[0.01]**	**iP[0.05]**	**iP[0.10]**	**MAiP**
TopX-CO-Proximity-articleOnly	0.6804	0.6795 (3)	0.5807	0.5265	0.2967
TopX-CO-Baseline-articleOnly	0.6700	0.6689 (4)	0.5940	0.5354	0.2951
TopX-CO-Focused-all	0.7464	0.6441 (11)	0.5300	0.4675	0.1852

Table 1. Results for the Focused Task: interpolated precision at different recall levels (ranks for iP[0.01] are in parentheses) and mean average interpolated precision

Table 1 shows the official results for these runs. It is evident that element-level retrieval generally yields a higher early precision than article-level retrieval, but

the quality quickly falls behind that of article-level retrieval. This is reflected in the official results where our article-level runs are at positions 3 and 4, whereas the element-level run is at position 11. Proximity scoring with gaps can in general help to improve early precision with article-level retrieval, at the cost of a slightly reduced recall. However, the MAiP average of the proximity-based run slightly improves over the baseline without proximity.

3.2 Other Tasks

We submitted a run to each of the other two tasks in the AdHoc track, where each of them was based on the CO titles of topics and the BM25-style element-level score shown in Section 2. To produce the runs for the RelevantInContext task, we ran TopX in document mode. This yielded a list of documents ordered by the highest score of any element within the document, together with a list of elements and their scores for each document. This yielded reasonable results with a MAgP value of 0.19329103, corresponding to rank 6 of all runs; this is a good improvement over 2007, which we mainly attribute to the better performance of the new scoring function.

To compute the best entry point for a document, we post-processed the RelevantInContext runs by simply selecting the element with highest score from each document and ordered them by score. This yielded reasonable results as well, with a MAgP value of 0.16888404, corresponding to rank 13 among all runs.

4 Conclusions and Future Work

This paper presented a structure-aware proximity score for XML retrieval that helps to improve the retrieval effectiveness of gap-free approaches for article-level retrieval. Our future work will focus on automatic methods to determine good gap sizes for elements, determining characteristics for queries where proximity boosts performance, and extending proximity scoring to queries with structural constraints.

References

1. M. Beigbeder. ENSM-SE at INEX 2007: Scoring with proximity. In *Preproceedings of the 6th INEX Workshop*, pages 53–55, 2007.
2. A. Broschart and R. Schenkel. Proximity-aware scoring for XML retrieval. In *SIGIR*, pages 845–846, 2008.
3. A. Broschart, R. Schenkel, M. Theobald, and G. Weikum. TopX @ INEX 2007. In *INEX*, volume 4862 of *LNCS*, pages 49–56. Springer, 2007.
4. 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.
5. O. de Kretser and A. Moffat. Effective document presentation with a locality-based similarity heuristic. In *SIGIR*, pages 113–120, 1999.
6. Y. Rasolofo and J. Savoy. Term proximity scoring for keyword-based retrieval systems. In *ECIR*, pages 207–218, 2003.
7. R. Song, M. J. Taylor, J.-R. Wen, H.-W. Hon, and Y. Yu. Viewing term proximity from a different perspective. In *ECIR*, pages 346–357, 2008.

Finding Good Elements for Focused Retrieval

Carolyn J. Crouch, Donald B. Crouch, Salil Bapat,
Sarika Mehta, Darshan Paranjape

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

Abstract. This paper describes the integration of our methodology for the dynamic retrieval of XML elements [1] with traditional article retrieval to facilitate the Focused and the Relevant-in-Context Tasks of the INEX 2008 Ad Hoc Track. The particular problems that arise for dynamic element retrieval in working with text containing both tagged and untagged elements have been solved [2]. The current challenge involves utilizing its ability to produce a rank-ordered list of elements in the context of focused retrieval. Our system is based on the Vector Space Model [5]; basic functions are performed using the Smart experimental retrieval system [4]. Experimental results are reported for the Focused, Relevant-in-Context, and Best-in-Context Tasks of both the 2007 and 2008 INEX Ad Hoc Tracks.

1. Introduction

Our work for INEX 2008 centers on producing good elements in a focused retrieval environment. Dynamic element retrieval—i.e., the dynamic retrieval of elements at the desired degree of granularity—has been the focus of our investigations at INEX for some time [1, 2]. We have demonstrated that our method works well for both structured [1] and semi-structured text [2] and that it produces a result identical to that produced by the search of the same query against the corresponding all-element index [3]. In [2], 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. The picture changes, however, when overlap is no longer allowed—i.e., when the task changes to focused retrieval. A review of our Ad Hoc results for all three INEX 2007 tasks shows that in each case, our results rank in the mid-range of participant scores. In 2008, our goal is to improve those results. Since the Ad Hoc tasks for INEX 2008 are identical to those of INEX 2007 and the evaluation procedures remain essentially unchanged as well, we are able to compare our 2008 results not only to those of other participants but also to our own earlier (2007) results for the same tasks.

2. Experiments with the INEX 2007 and 2008 Collections

In this section, we include the results produced by our methods for the three INEX 2007 Ad-hoc tasks, namely, Focused, Relevant-in-Context, and Best-in-Context. To produce its best results, our system needs tuning to establish appropriate values for term weighting with respect to a metric (in this case, iP[0.01] for the focused task); these results reflect that tuning.

There are two important issues which arise with respect to the Focused and Relevant-in-Context (RIC) tasks. The first relates to how the documents of interest (i.e., with respect to a specific query) are identified. The second is the method by which the focused elements are selected from those documents. The results reported in [2] for 2007 compare the values achieved by dynamic element retrieval to (base case) all-element retrieval. The focused elements themselves are selected based on correlation (i.e., the highest-correlating element along a path is chosen). Thus, for these experiments, the documents of interest were determined by the process of dynamic element retrieval (see [2] for details), and the focused elements were selected based on correlation.

To improve these results, we revised our approach to focused retrieval by incorporating dynamic element retrieval with article retrieval as follows. For each query, we retrieve n articles. We then use dynamic element retrieval to produce the rank-ordered list of all elements (from these n documents) having a positive correlation with the query. Overlap is removed by what we refer to as a terminal-node strategy, which always gives precedence to the terminal node along a path (ignoring correlation). Then m focused elements from this list are reported. This is the method by which focused elements are produced for the 2007 and 2008 Focused and RIC tasks reported below. Negative terms (those preceded by a minus) are removed from both query sets.

2.1 Focused Task

The INEX 2007 Focused Task results are given in Table 1. Best results are produced when 25 documents are retrieved and 500 elements reported (n = 25, m = 500); at 0.5386 this value would place at rank 1 in the 2007 rankings. (If negative terms are not omitted from the query set, the corresponding value at n = 50, m = 750 is 0.5293, which still exceeds the first place value in the rankings).

The results produced by the same methodology for the INEX 2008 Focused Task are given in Table 2. Best case results here are produced at n = 25 and m = 500, where the value of iP[0.01] is 0.6225 (equivalent to rank 23). Our INEX 2008 submission (which included negative query terms) produced a value of 0.6251 at iP[0.01] for a rank of 21.

Table 1. iP[0.01] Results for Focused Task (2007)

NUMBER OF ARTICLES	NUMBER OF ELEMENTS						
	50	**250**	**500**	**1000**	**2000**	**3000**	**4000**
25	0.4972	0.5383	0.5386	0.5381	0.5381	0.5381	0.5381
50	0.4751	0.5338	0.5367	0.5343	0.5343	0.5343	0.5343
100	0.4510	0.4962	0.5242	0.5273	0.5238	0.5238	0.5238
250	0.4697	0.4603	0.4920	0.5031	0.5136	0.5098	0.5098
500	0.4664	0.4589	0.4746	0.4990	0.4995	0.5052	0.5058

Table 2. iP[0.01] Results for Focused Task (2008)

NUMBER OF ARTICLES	NUMBER OF ELEMENTS						
	50	**250**	**500**	**1000**	**2000**	**3000**	**4000**
25	0.6222	0.6225	0.6091	0.6091	0.6091	0.6091	0.6091
50	0.6064	0.5995	0.6104	0.6006	0.6006	0.6006	0.6006
100	0.6054	0.5759	0.5871	0.5997	0.5965	0.5965	0.5965
250	0.6009	0.5769	0.5711	0.5744	0.5746	0.5811	0.5749
500	0.5972	0.5930	0.5499	0.5740	0.5723	0.5643	0.5681

2.2 Relevant-in-Context Task

The RIC results are produced using the result file from the Focused task and grouping the elements by article. Results are reported in document-rank order. Table 3 shows the 2007 best case result with an MAgP value of 0.1415 at n = 250, m = 4000 (rank 9

for the 2007 rankings). 2008 RIC results are shown in Table 4. The best value is achieved at n = 500, m = 2000, where MAgP = 0.1751 (which would rank at 15 for 2008). Our submitted run (which includes negative terms) produced a MAgP value of 0.1714 and ranked 18.

Table 3. MAgP Results for RIC Task (2007)

NUMBER OF ARTICLES	NUMBER OF ELEMENTS								
	50	**250**	**500**	**750**	**1000**	**1500**	**2000**	**3000**	**4000**
25	0.0767	0.0951	0.0978	0.0977	0.0977	0.0977	0.0977	0.0977	0.0977
50	0.0772	0.1034	0.1129	0.0977	0.1155	0.1154	0.1154	0.1154	0.1154
100	0.0772	0.1098	0.1190	0.1253	0.1305	0.1310	0.1315	0.1313	0.1313
250	0.0771	0.1113	0.1243	0.1292	0.1330	0.1385	0.1408	0.1408	0.1415
500	0.0772	0.1112	0.1230	0.1292	0.1326	0.1373	0.1393	0.1391	0.1405

Table 4. MAgP Results for RIC Task (2008)

NUMBER OF ARTICLES	NUMBER OF ELEMENTS								
	50	**250**	**500**	**750**	**1000**	**1500**	**2000**	**3000**	**4000**
25	0.0975	0.1034	0.1026	0.1026	0.1026	0.1026	0.1026	0.1026	0.1026
50	0.1108	0.1277	0.127	0.126	0.1259	0.1259	0.1259	0.1259	0.1259
100	0.1128	0.1426	0.1488	0.1486	0.1465	0.1457	0.1457	0.1457	0.1457
250	0.1126	0.1508	0.1606	0.1645	0.1667	0.1661	0.1649	0.1632	0.1622
500	0.1134	0.154	0.1642	0.1678	0.1707	0.1749	**0.1751**	0.1726	0.1675

2.3 Best-in-Context Task

Finding the Best Entry Point (BEP) is a task which is not related to focused retrieval. We examined a number of factors which might be useful in determining the BEP, including correlation, tag set membership, and physical location. For 2007, best results were obtained based purely on physical position, but a very similar result was produced by a combination of two factors (tag set membership and location). Investigations into this question continue.

5. Conclusions

Our current methods performed very well for the 2007 Focused and RIC tasks. When we apply the same methods to the same tasks in 2008, the results, while acceptable, are not as good. Our current efforts are directed at determining the cause.

References

[1] Crouch, C. Dynamic element retrieval in a structured environment. *ACM TOIS* 24(4), 437-454 (2006).

[2] Crouch, C, Crouch, D., Kamat, N., Malik, V., Mone, A. Dynamic element retrieval in the Wikipedia collection. In: *Proceedings of INEX 2007*, pp. 70-29 (2007).

[3] 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

[4] Salton, G., ed. *The Smart Rretrieval System—Experiments in Automatic Document Processing*. Prentice-Hall. (1971)

[5] Salton, G., Wong, A., Yang, C. S. A vector space model for automatic indexing. *Comm. ACM* 18 (11), pp. 613-620 (1975)

Study of two learning to rank models for Inex'08

David Buffoni and Patrick Gallinari

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

Abstract. We present a Retrieval Information system for XML documents using a learning to rank approach. This system learns a ranking function using a training of queries and relevance judgments on a subset of documents. We compare two ranking models, one that optimizes a classical ranking loss and one that takes into account the position of the mis-ordered pair in the list.

1 Introduction

Ranking algorithms have been applied in the Machine Learning field for some time. In the field of IR, they have first been used for combining features or preference relations in tasks such as meta search [1], [2]. Using learning to rank functions has led to improved performances in a series of tasks such as passage classification or automatic summarization [3]. More recently, they have been used for learning the ranking function of search engines [4], [5], [8], [9].

Ranking algorithms work by combining features which characterize the data elements to be ranked. In our case, these features will depend on the document itself. The ranking algorithm will learn to combine these different features in an optimal way, according to a specific loss function, using a set of examples. The direct approach of learning to rank algorithms is to optimize the mean pairwise error as in [5] or in [2]. This criterion is never used as an evaluation measure in Information Retrieval, because users are more interested in the top ranked items. Recent works design approximation on Information Retrieval's error measures as Mean Average Precision [6] or Discounted Cumulated Gain [7].

We propose here an approach of a learning to rank algorithm which optimizes an error that focuses on the top ranked documents.

The paper is organized as follows, in section 2 we present the ranking model. In section 3 we comment the results obtained by our model and compare them to a ranking model which optimizes the mean of mis-ordered pairs [5].

2 Ranking model

In this section we briefly describe a probabilistic model of ranking which can be adapted to Information Retrieval or Structured Information Retrieval.

The main idea behind the Machine Learning Ranking is to learn a total strict order on $\mathcal{X}$, a set of elements. This allows us to compare any pair of elements in the set.

Given this total order, we are able to order any subset of $\mathcal{X}$ in a ranking list. For Information Retrieval on XML documents, $\mathcal{X}$ will be the set of couples (documents, query) for all documents and queries in the collection and the total order is the natural order on the document's scores. In addition, we need a training set of ordered pairs of examples to learn how to rank. This training set will provide us with a partial order on the elements of $\mathcal{X}$. Our algorithm will use this information to learn a total order on $\mathcal{X}$ and it will then be able to rank new elements.

2.1 Notations

As described above, we assume available a set $\mathcal{X}$ of elements ordered by a partial order noted $\prec$. This relation will be used when it is possible to compare element pairs of $\mathcal{X}$. Let $\mathcal{D}$ be the set of all documents in the Wikipedia collection and $\mathcal{Q}$ be the set of CO-queries. In the context of structured IR, we will make define $\mathcal{X} = \mathcal{Q} \times \mathcal{D}$. The partial order hypothesis on $\mathcal{X} = \mathcal{Q} \times \mathcal{D}$, means that for a subset of the queries in $\mathcal{Q}$ we know preferences between some of the documents in $\mathcal{D}$. For a given query, these preferences will define a partial order on the documents in $\mathcal{D}$. The preferences among documents are provided by manual assessments during INEX'06 and INEX'07 sessions.

Ranking We represent each element $x \in \mathcal{X}$ by a vector $(x_1, x_2, ..., x_l)$ where x_i are features needed to rank elements of $\mathcal{X}$. The following linear combination of features is used to define the ranking function f_ω, that we will use to learn a total order on $\mathcal{X}$:

$$f_\omega(x) = \omega_1 \cdot Indri(x) + \omega_2 \cdot Okapi(title(x)) + \omega_3 \cdot LM(title(x)) \qquad (1)$$

where ω_i^l are the parameters of the combination to be learned, Indri [10] is the score of the search engine in the Lemur toolkit, Okapi is an Okapi [12] Model applied on the title of a document and LM for a Language Model using a Jelinek-Mercer smoothing function.

Ranking loss The first model tries to minimize the exponential AUC loss function as below :

$$R_e(\mathcal{X}, \omega) = \sum_{q \in \mathcal{Q}} \left\{ \left(\sum_{x \in \mathcal{X}_q^-} e^{f_\omega(x)} \right) \left(\sum_{x' \in \mathcal{X}_q^+} e^{-f_\omega(x')} \right) \right\} \qquad (2)$$

where $\mathcal{X}_q^+ \prec \mathcal{X}_q^-$ means that $\mathcal{X}_q^-$ is better than $\mathcal{X}_q^-$.

The second model tries to optimize a loss which depends on the position in the ranked list. We assume a mis-ordered pair at the top of the list will cost more than a mis-ordered pair at the end.

$$R_\Phi(\mathcal{X}, \omega) = \sum_{q \in \mathcal{Q}} \left\{ \sum_{x \in \mathcal{X}_q^- \, x' \in \mathcal{X}_q^+} Owa(f(x') - f(x)) \right\} \tag{3}$$

where $Owa(f(x') - f(x))$ is a decrease function : $R^n \rightarrow R$.

3 Experiments

3.1 Learning base

The Wikipedia collection [13] has been used with different sets of queries for training and testing. INEX 2006 queries and assessments were used for training and the relevance judgments of INEX 2007 gived us a validation set to determine hyperparameters of the model. We built for each topic, a subset of 1500 documents according to the Indri search engine and we transformed each document in a vector of features as presented before. At the end, we tested the two ranking models on the INEX 2008 queries.

3.2 Results

The runs submitted to the official evaluation were bugged. New results will be presented at the workshop.

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. Amini, M.R., Usunier, N., Gallinari, P.: Automatic text summarization based on word-clusters and ranking algorithms. In: ECIR'05: European Conference on Information Retrieval. (2005) 142–156
4. Craswell, N., Robertson, S., Zaragoza, H., Taylor, M.: Relevance weighting for query independent evidence. In: SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference. (2005)
5. Vittaut, J-N., Gallinari, P.: Machine Learning Ranking for Structured Information Retrieval. In: ECIR'06: European Conference on Information Retrieval. (2006) 338–349
6. Yue,Y., Finley,T.,Radlinski,F.,Joachims,T. : A support vector method for optimizing average precision. In: SIGIR'07:Proceedings of the 28th annual international ACM SIGIR conference. (2007)

7. Chapelle, O., Le, Q., Smola, A.: Large margin optimization of ranking measures. In: NIPS'07: Neurol Information Processing Systems, (2007)
8. Xu, J., Li, H.: AdaRank: A Boosting Algorithm for Information Retrieval. In: SIGIR '07: Proceedings of the 28th annual international ACM SIGIR conference. (2007)
9. 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)
10. Metzler, D., Croft, W.B., : Combining the Language Model and Inference Network Approaches to Retrieval. In: Information Processing and Management Special Issue on Bayesian Networks and Information Retrieval. (2004)
11. Vittaut J.N., Gallinari P. : Supervised and Semi-Supervised Machine Learning Ranking. INEX 2006 preproceedings, (2006)
12. Robertson, S.E., Walker, S., Hancock-Beaulieu, M., Gull, A., Lau, M.: Okapi at TREC. In: Text REtrieval Conference. (1992) 21–30
13. Denoyer L., Gallinari P.: The Wikipedia XML Corpus SIGIR Forum (2006)

New Utility Models for the Garnata Information Retrieval System at INEX'08

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

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

Abstract. In this work we propose new utility models for the structured information retrieval system Garnata, and expose the results of our participation at INEX'08 in the AdHoc track using this system.

1 Introduction

Garnata [5] is a Structured Information Retrieval System for XML documents, based on probabilistic graphical models [7, 8], developed by members of the research group "Uncertainty Treatment in Artificial Intelligence" at the University of Granada. Garnata has already been tested at two editions of the INEX Workshop [4, 6], and its theoretical basis is explained in more detail in [1, 2].

Garnata computes the relevance degree of each component or structural unit in a document by combining two different types of information. On the one hand, the specificity of the component with respect to the query: the more terms in the component appear in the query, the more relevant becomes the component, that is to say, the more clearly the component is only about (at least a part of) the topic of the query. On the other hand, the exhaustivity of the component with respect to the query: the more terms in the query match with terms in the component, the more relevant the component is, i.e., the more clearly the component comprises the topic of the query. The components that best satisfy the user information need expressed by means of the query should be, simultaneously, as specific and exhaustive as possible.

These two dimensions of the relevance of a component with respect to the query are calculated in a different way. To compute the specificity, the probability of relevance of each component is obtained through an inference process in a Bayesian network representing the structured document collection. The exhaustivity is obtained by first defining the utility of each component as a function of the proportion of the terms in the query that appear in this component. Then the Bayesian network is transformed into an influence diagram which computes the expected utility of each component, by combining the probabilities of relevance and the utilities in a principled way.

In this work we propose a modification of the system by defining the utility in a different manner, in such a way that those components that do not contain most of the query terms are penalized more heavily. By defining a parametric model, it is possible to adjust the degree of utility to make the system behave more similarly to a strict AND (if not all or almost all the query terms are in the considered component, this one will be scarcely relevant) or to a less strict AND.

2 Utility Models in the Garnata System

As we focus in this work on the utility component of the Garnata system, we will not enter into details of the Bayesian network model representing the document collection. This model is able to efficiently compute the posterior probabilities of relevance of all the structural units U of all the documents, given a query Q, $p(U|Q)$. These probabilities represent the specificity component of each structural unit U: the more terms indexing U also belong to Q, the more probable is U.

The Bayesian network is then enlarged by including decision variables R_U, representing the possible alternatives available to the decision maker (retrieve unit U), and utility variables V_U, thus transforming it into an influence diagram. The objective is to compute the expected utility of each decision given Q, $EU(R_U|Q)$.

In Garnata the utility value V_U of each structural unit U is composed of a component which depends on the involved unit, other component which depends only on the kind of tag associated to that unit, and another component independent on the specific unit (these three components are multiplied in order to form the utility value, see [4]).

The part depending on the involved unit, which is the only one we are going to modify, is defined as the sum of the inverted document frequencies of those terms contained in U that also belong to the query Q, normalized by the sum of the idfs of the terms contained in the query: a unit U will be more useful (more exhaustive), with respect to a query Q, as more terms of Q also belong to U:

$$nidf_Q(U) = \frac{\sum_{T \in An(U) \cap Q} idf(T)}{\sum_{T \in Q} idf(T)} \tag{1}$$

$An(U)$ in the previous equation represents the set of terms contained (either directly or indirectly) in the structural unit U.

3 New Utility Models

As it can be observed from equation (1), the utility or exhaustivity of a structural unit U with respect to a query Q grows linearly with the number of query terms appearing in U (reaching a maximum equal to 1 when all the terms of the

query appear in the unit). In our experience with the system in different applications [3, 4], we have observed that this linear growing, when combined with the probabilities computed from the Bayesian network (which measure specificity), can cause that small structural units, which only match with a fraction of the query terms, become more relevant that other, greater structural units that contain more terms from the query. In many cases this behaviour is not the expected one, because probably a user who employs several terms to express his/her query is expecting to find most of these terms in the structural units obtained as the answer of the system to this query. For that reason we believe that it is interesting to define other utility models which give more importance (in a non-linear way) to the appearance of most of the terms in the query.

In this work we propose a parametric non-linear utility model that, as the parameter grows, the more terms from the query must be contained in a structural unit to get a high utility value. A way of obtaining this behaviour is by using the following transformation:

$$nidf_{Q,n}(U) = nidf_Q(U)\frac{e^{(nidf_Q(U))^n} - 1}{e - 1} \tag{2}$$

In this way, when $n = 0$ we have $nidf_{Q,0}(U) = nidf_Q(U)$, that is to say, we reproduce the original model, and the greater the value of the integer parameter n we obtain a behaviour more similar to a strict AND operator. In Figure 1 we can observe several plots of the function $x\frac{e^{x^n}-1}{e-1}$ for different values of n.

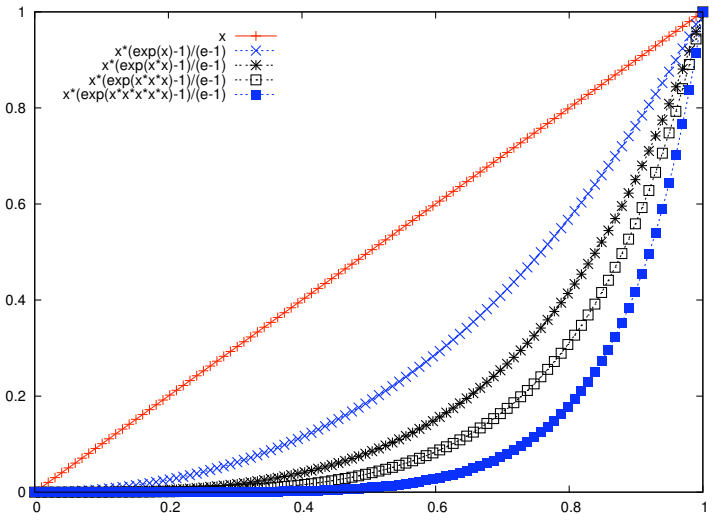


Fig. 1. Function $x\frac{e^{x^n}-1}{e-1}$, for $n = 0, 1, 2, 3, 5$.

4 Experimental Results

In this INEX 2008 edition, we have participated submitting nine runs in the ad hoc track (content only). More specifically, three in each of the *Focused*, *Relevant in Context* and *Best in Context* sub-tasks. Table 1 shows the positions in the ranking according to the official evaluation measures ($MAgP$ for *Best In Context* and *Relevant in Context*, and $iP[0.01]$ for *Focused*), the sub-task and finally the run identifier.

Position	Value	Sub-task	RunId
52	0.468793	Focused	p8_u3_exp_5_1110
53	0.46700756	Focused	p8_u3_exp_3_1110
54	0.44861908	Focused	p15_u3_exp_5_1110
25	0.1575066	Relevant In Context	p8_u3_exp_5_1110
26	0.1575066	Relevant In Context	p8_u3_exp_5_0100
27	0.15174653	Relevant In Context	p8_u3_exp_3_1110
18	0.14567388	Best In Context	p8_u3_exp_5_0100
19	0.14563043	Best In Context	p8_u3_exp_3_0100
22	0.13697283	Best In Context	p15_u3_exp_3_0100

Table 1. Runs submitted to the INEX'2008 ad hoc tasks and positions in the rankings.

With respect to the parameters, we have used the weight files 8 and 15 (p8 and p15 as prefixes of the run identifiers), and utility files 3 (u3, contained in the identifiers), with the first values presented in Table 2 and in Table 3 the second ones (see [4] for details about these parameters and their use by the model).

Tag	Weight file 8	Weight file 15
name	20	200
title	20	50
caption	10	30
collectionlink	10	30
emph2	10	30
emph3	10	30
conversionwarning	0	0
languagelink	0	0
template	0	0
default value	1	1

Table 2. Importance of the different types of units used in the official runs.

Tag	Utility file 3
conversionwarning	0
name	0.85
title	0.85
collectionlink	0.75
languagelink	0.0
article	2.5
section	1.25
p	1.5
body	2.0
emph2	1.0
emph3	1.0
default value	1.0

Table 3. Relative utility values of the different types of units used in the official runs.

Finally, the suffix of the run identifier corresponds to the values of each of the four configurations of the component of the utility function independent on the involved unit (see [4]).

Although there has been a significant reduction of runs submitted in this 2008 edition – measured as focused retrieval – (*Focused*: from 79 last year to 61 this edition; *Relevant in Context*: from 66 to 40; *Best in context*: from 71 to 35), we could say that in terms of the percentiles of the positions in the rankings, we are improving our results in *Relevant in Context* (from percentiles 66-74 last year, to 62-67 this year) and *Best in Context* (from 63-70 to 51-62), and more or less we maintain the positions in *Focused* (from 84-89 to 85-88).

It is noticeable that within the *Focused* task, Garnata's performance is relatively low, and keeps more or less the same positions than last year, and how the methods described in [4] for adjusting the output for the requirements of the other two tasks make a good job from the raw results from Garnata. Clearly *Best in Context* is the sub-task where the performance is higher, and where the best improvement is achieved.

Finally, we have run an experiment with the 2008 assessments, evaluated with the same evaluation tools used in this edition, where we do not apply the transformation presented in Eq. (2), but applying the original Eq. (1), $nidf_Q(U)$, in order to determine the improvement of the new approach. Table 4 shows the values of the official evaluation measures with the utility model used in previous editions (first column), this year with the new model (second column) and the percentace of change (third column). As noticed, the percentages of change are most of them very large, fact that confirms our initial hypothesis that the new transformation could improve the results.

With $nidf_Q(U)$	With $nidf_{Q,n}(U)$	%Change	Sub-tasks	Run Id.
0.365532	0.468793	28.25	Focused	p8_u3_exp_5_1110
0.365532	0.467008	27.76	Focused	p8_u3_exp_3_1110
0.341435	0.448619	31.39	Focused	p15_u3_exp_5_1110
0.082615	0.157507	90.65	Relevant In Context	p8_u3_exp_5_1110
0.067482	0.157507	133.41	Relevant In Context	p8_u3_exp_5_0100
0.082615	0.151747	83.68	Relevant In Context	p8_u3_exp_3_1110
0.075307	0.145674	93.44	Best In Context	p8_u3_exp_5_0100
0.075307	0.145630	93.38	Best In Context	p8_u3_exp_3_0100
0.078243	0.136973	75.06	Best In Context	p15_u3_exp_3_0100

Table 4. Comparison between runs applying or not the transformation in Eq. (2).

5 Concluding Remarks

In this paper we have presented the participation of the University of Granada group in the 2008 INEX edition in the AdHoc tasks. This is based on the work developed in previous years, but introducing a new utility model which gives more importance (in a non-linear way) to the appearance of most of the terms in the query. We have shown in the previous section that this new approach considerably improves the results with respect to not to use it.

With respect to the comparison of our results with the rest of participants, we could say that we are in the middle of the rankings, improving with respect to the last edition of INEX.

Regarding future research in the context of INEX, we have to work in the improvement of the raw results of Garnata, as they are the base for the different sub-tasks. Also, we have designed an approach to answer CAS queries, which will be evaluated in the next edition of the evaluation campaign.

Acknowledgments. This work has been jointly supported by the Spanish Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía and Ministerio de Educación y Ciencia, under projects TIC-276 and TIN2005-02516, respectively.

References

1. L.M. de Campos, J.M. Fernández-Luna and J.F. Huete. Using context information in structured document retrieval: An approach using Influence diagrams. *Information Processing & Management*, **40**(5):829–847, 2004.
2. L.M. de Campos, J.M. Fernández-Luna and J.F. Huete. Improving the context-based influence diagram for structured retrieval. *Lecture Notes in Computer Science*, **3408**:215–229, 2005.
3. L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, C. Martín, A.E. Romero. An information retrieval system for parliamentary documents. In *Bayesian Networks: A Practical Guide to Applications*, O. Pourret, P. Naim, B. Marcot (Eds.), 203–223, Wiley, 2008.

4. L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, Carlos Martín-Dancausa, A.E. Romero. The Garnata information retrieval system at INEX'07. Focused Access to XML Documents, N. Fuhr, J. Kamps, M. Lalmas, A. Trotman (Eds.), *Lecture Notes in Computer Science*, **4862**:57–69, 2008.
5. L.M. de Campos, J.M. Fernández-Luna, J.F. Huete and A.E. Romero. Garnata: An information retrieval system for structured documents based on probabilistic graphical models. Proceedings of the Eleventh International Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU), 1024–1031, 2006.
6. L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, A.E. Romero. Influence diagrams and structured retrieval: Garnata implementing the SID and CID models at INEX'06. Comparative Evaluation of XML Information Retrieval Systems, N. Fuhr, M. Lalmas, A. Trotman (Eds.), *Lecture Notes in Computer Science*, **4518**:165–177, 2007.
7. F.V. Jensen. *Bayesian Networks and Decision Graphs*, Springer Verlag, 2001.
8. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan and Kaufmann, San Mateo, 1988.

The University of Amsterdam at INEX 2008: Ad Hoc, Book, Entity Ranking, Interactive, Link the Wiki, and XML Mining Tracks

Khairun Nisa Fachry[1], Jaap Kamps[1,2], Rianne Kaptein[1], Marijn Koolen[1], and Junte Zhang[1]

[1] Archives and Information Studies, Faculty of Humanities, University of Amsterdam
[2] ISLA, Faculty of Science, University of Amsterdam

Abstract. This paper documents our experiments in six of the INEX 2008 tracks: Ad Hoc, Book Search, Entity Ranking, Interactive, Link the Wiki, and XML Mining. We discuss our aims, experiments, results and findings for each of these tracks.

1 Introduction

In this paper, we describe our participation in six of the INEX 2008 tracks.

For the *Ad Hoc Track*, we explore the good performance of standard document retrieval systems in term of their superior document ranking when compared to element retrieval approaches. Our aim is to investigate the relative effectiveness of both approaches. We experiment with combining the two approaches to get the best of both worlds.

For the *Book Track*, our aims were two-fold: 1) to investigate the effectiveness of using book level evidence for page level retrieval, and 2) experiment with using Wikipedia as a rich resource for topical descriptions of the knowledge found in books, to mediate between user queries and books in the INEX Book Track collection.

For the *Entity Ranking Track*, our aim was to explore the relations and dependencies between Wikipedia pages, categories and links.

For the *Interactive Track*, we particated in the concerted experiment designed by the organizers. The specific aims of the experiment were to investigate the impact of the task context for two types of simulated tasks were defined that are believed to represent typical information needs of Wikipedia-users: fact-finding tasks and research tasks.

For the *Link the Wiki Track*, our aim was to investigate a two-tier approach to link detection: first, a relevant pool of foster (candidate) articles is collected; second, substring matching with the list of collected titles to establish an actual link. We specifically look at the effectiveness of the two-tier approach on early precision and on recall.

For the *XML Mining Track*, our aim was to explore whether we can use link information to improve classification accuracy.

The document collection for all tracks except the Book Search track is based on the English Wikipedia [31]. The collection has been converted from the wiki-syntax to an XML format [4]. The XML collection has more than 650,000 documents and over 50,000,000 elements using 1,241 different tag names. However, of these, 779 tags occur

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

The rest of the paper describes our experiments for each of the tracks in a relatively self-contained sections. First, in Section 2, we report the results for the Ad Hoc Track. Then Section 3 presents our retrieval approach in the Book Track. Followed by our results for the Entity Ranking Track in Section 4. Then, in Section 5, we discuss our Interactive Track experiments. Followed by Section 6 detailing our approach and results for the Link the Wiki Track. And last but not least, in Section 7, we discuss our XML Mining Track experiments. Finally, in Section 8, we discuss our findings and draw some conclusions.

2 Ad Hoc Track

For the INEX 2008 Ad Hoc Track we aim to investigate several methods of combining article retrieval and element retrieval approaches. We will first describe our indexing approach, then the run combination methods we adopted, the retrieval framework, and finally per task, we present and discuss our results.

2.1 Retrieval Model and Indexing

Our retrieval system is based on the Lucene engine with a number of home-grown extensions [11, 20].

For the Ad Hoc Track, we use a language model where the score for a element e given a query q is calculated as:

$$P(e|q) = P(e) \cdot P(q|e) \tag{1}$$

where $P(q|e)$ can be viewed as a query generation process—what is the chance that the query is derived from this element—and $P(e)$ an element prior that provides an elegant way to incorporate link evidence and other query independent evidence [9, 19].

We estimate $P(q|e)$ using Jelinek-Mercer smoothing against the whole collection, i.e., for a collection D, element e and query q:

$$P(q|e) = \prod_{t \in q} \left((1 - \lambda) \cdot P(t|D) + \lambda \cdot P(t|e) \right), \tag{2}$$

where $P(t|e) = \frac{\text{freq}(t,e)}{|e|}$ and $P(t|D) = \frac{\text{freq}(t,D)}{\sum_{e' \in D} |e|}$.

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

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

where $|e|$ is the size of an element e. The β parameter introduces a length bias which is proportional to the element length with $\beta = 1$ (the default setting). For a more thorough description of our retrieval approach we refer to [27]. For comprehensive experiments on the earlier INEX data, see [24].

Our indexing approach is based on our earlier work [6, 15, 16, 25, 26, 27].

– *Element index*: Our main index contains all retrievable elements, where we index all textual content of the element including the textual content of their descendants. This results in the "traditional" overlapping element index in the same way as we have done in the previous years [26].
– *Article index*: We also build an index containing all full-text articles (i.e., all wikipages) as is standard in IR.

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

2.2 Combining Article and Element Retrieval

Our experiments with combining runs all use the same two base runs:

– *Article*: a run using the Article index; and
– *Element*: a run using the element index.

Both runs use default parameters for the language model ($\lambda = 0.15, \beta = 1.0$). As shown by Kamps et al. [17], article retrieval leads to a better document ranking, whereas element retrieval fares better at retrieving relevant text within documents. We therefore assume that a combined approach, using the document ranking of an article level run with the within document element ranking of an element level run, outperforms both runs on the "in context" tasks.

We experiment with three methods of combining the article and element results.

1. *ArtRank*: retain the article ranking, replacing each article by its elements retrieved in the element run. If no elements are retrieved, use the full article.
2. *Multiplication*: multiply element score with article score of the article it belongs to. If an element's corresponding article is not retrieved in the top 1,000 results of the article run, use only the element score.
3. *CombSUM*: normalise retrieval scores (by dividing by highest score in the results list) and add the article score to each element score (if article is not in top 1,000 results for that topic, only element score is used). Thus elements get a boost if the full article is retrieved in the top 1,000 results of the article run.

Our Focused and Relevant in Context submissions are all based on the following base "Thorough" runs:

– `inex08_art_B1_loc_in_100_and_el_B1_T`: using the ArtRank method to combine article and element runs;
– `inex08_art_B1_loc_in_100_comb_sum_el_B1_T`: using the CombSUM method to combine article and element runs; and
– `inex08_art_B1_loc_in_100_x_el_B1_T`: using Multiplication to combine article and element runs.

Table 1: Results for the Ad Hoc Track Focused Task (runs in emphatic are no official submissions)

Run	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
Article	0.5712	0.5635	0.5189	0.4522	**0.2308**
Element	**0.6627**	0.5535	0.4586	0.4062	0.1710
ArtRank	0.6320	**0.6025**	0.5054	0.4569	0.1991
CombSUM	0.6556	0.5901	0.4983	0.4553	0.1989
Multiplication	0.6508	0.5614	0.4547	0.4117	0.1815
Element CAS	0.6196	0.5607	0.4941	0.4396	0.2000
ArtRank CAS	0.6096	0.5891	**0.5361**	**0.4629**	0.2140
CombSUM CAS	0.6038	0.5811	0.5158	0.4506	0.2044
Multiplication CAS	0.6077	0.5855	0.5328	0.4601	0.2126

We also made CAS versions of these Thorough runs, using the same filtering method as last year [6]. That is, we pool all the target elements of all topics in the 2008 topic set, and filter all runs by removing any element type that is not in this pool of target elements.

All our official runs for all three tasks are based on these Thorough runs. Because of the lengthy names of the runs, and to increase clarity and consistency of presentation, we denote all the official runs by the methods used, instead of the official run names we used for submission.

2.3 Focused Task

To ensure the Focused run has no overlap, it is post-processed by a straightforward list-based removal strategy. We traverse the list top-down, and simply remove any element that is an ancestor or descendant of an element seen earlier in the list. For example, if the first result from an article is the article itself, we will not include any further element from this article. In the case of the CAS runs, we first apply the CAS filter and then remove overlap. Doing this the other way around, we would first remove possibly relevant target elements if some overlapping non-target elements receive a higher score. These high scoring non-target elements are then removed in the CAS filtering step, and we would lose many more promising elements than if we apply the CAS filter first.

Table 1 shows the results for the Focused Task. Somewhat surprisingly, the Article run outperforms the Element run on the official Focused measure iP[0.01], although the Element run fares much better at the earliest precision level iP[0.00]. Both CombSUM and Multiplication attain higher scores for iP[0.00] than ArtRank, but the latter keeps higher precision at further recall levels. The Multiplication method loses much more precision than the other two methods. Compared to the baseline runs Article and Element, the combination methods ArtRank and CombSUM lead to substantial improvements at iP[0.01], where the Multiplication method performs slightly worse than the Article run. However, the standard Article run clearly outperforms all other runs when looking at overall precision.

Looking at the CAS runs, we see that the differences are small, with ArtRank leading to the highest iP[0.01] and MAiP scores. The CAS filtering method leads to improvements in overall precision—all MAiP scores go up compared to the non CAS

Table 2: Results for the Ad Hoc Track Relevant in Context Task (runs in emphatic are no official submissions)

Run	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
Article	0.3376	0.2807	0.2107	0.1605	0.1634
Element	0.2784	0.2407	0.1879	0.1471	0.1484
ArtRank	0.3406	0.2820	0.2120	0.1627	0.1692
CombSUM	0.3281	0.2693	0.2099	0.1615	0.1665
Multiplication	0.3295	0.2827	0.2136	0.1654	0.1695
Element CAS	0.3378	0.2837	**0.2236**	0.1719	0.1703
ArtRank CAS	0.3437	0.2897	0.2207	0.1712	0.1734
CombSUM CAS	0.3481	**0.2991**	0.2200	**0.1726**	**0.1752**
Multiplication CAS	**0.3482**	0.2888	0.2198	0.1724	0.1748

variants—but has a negative effect for early precision as both iP[0.00] and iP[0.01] scores go down, except for the Multiplication run, where the iP[0.01] score goes up. Also, the CAS version of the Multiplication run does improve upon the Article run for precision up to 10% recall.

2.4 Relevant in Context Task

For the Relevant in Context task, we use the Focused runs and cluster all elements belonging to the same article together, and order the article clusters by the highest scoring element. Table 2 shows the results for the Relevant in Context Task. The Article run is better than the Element across the board, which is to be expected, given the results reported in [17]. It has a superior article ranking compared to the Element run, and as we saw in the previous section, it even outperformed the Element run on the official measure for the Focused task. However, this time, the combination methods ArtRank and Multiplication do better than the Article run on all reported measures, except for the Multiplication run on gP[5]. Since they use the same article ranking as the Article run, the higher precision scores of the ArtRank and Multiplication show that the elements retrieved in the Element run can improve the precision of the Article run. The CombSUM method, while not far behind, fails to improve upon the Article run on early precision levels (cutoffs 5, 10, and 25). Through the weighted combination of article and element scores, its article ranking is somewhat different from the article ranking of the Article run (and the ArtRank and Multiplication runs).

The CAS filtering method leads to further improvements. The Element CAS run outperforms the standard Article run, and the combination methods show higher precision scores than their non CAS counterparts on all cutoff levels. This time, the CombSUM method benefits most from the CAS filter. Whereas it was well behind on performance compared to the other two combination methods, its CAS version has the highest scores for gP[10], gP[50] and MAgP. Perhaps surprisingly, the Element CAS run is even on par with the combined runs.

2.5 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 3: Results for the Ad Hoc Track Best in Context Task (runs in emphatic are no official submissions)

Run	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
Element	0.2372	0.2213	0.1778	0.1384	0.1394
Article	**0.3447**	**0.2870**	**0.2203**	**0.1681**	**0.1693**
Article offset 190	0.2462	0.2042	0.1581	0.1204	0.1228
ArtRank	0.2954	0.2495	0.1849	0.1456	0.1580
CombSUM	0.2720	0.2255	0.1872	0.1487	0.1560
Multiplication	0.2782	0.2399	0.1866	0.1496	0.1577
Element CAS	0.2758	0.2410	0.1929	0.1517	0.1487
ArtRank CAS	0.3101	0.2616	0.1952	0.1539	0.1587
CombSUM CAS	0.3081	0.2547	0.1942	0.1532	0.1581
Multiplication CAS	0.3098	0.2595	0.1944	0.1545	0.1596

We experimented with three methods of selecting the best entry point:

- *Highest Scoring Element*: simply the highest scoring element (HSE) returned for each article. We use this on the ArtRank combined run;
- *offset 0*: the start of each returned article; and
- *offset 190*: the median distance from the start of the article of the best entry points in the 2007 assessments.

Table 3 shows the results for the Best in Context Task.

The Article run is far superior to the Element run for the Best in Context Task, on all cutoff levels and in MAgP. In fact, the Article run outperforms all combined runs and CAS runs. The combined ArtRank run does better than the pure article run with BEPs at offset 190. Note that both these two runs have the same article ranking as the standard Article run. The highest scoring element is thus a better estimation of the BEP than the median BEP offset over a large number of topics. However, using the start of the element clearly outperforms both other runs. Of the three run combination methods, ArtRank gets better scores at early precision levels (cutoffs 5 and 10), but is overtaken by the Multiplication method at further cutoff levels. All three combinations do outperform the Element run and the article run with fixed offset of 190.

The CAS runs again improve upon their non CAS variants, showing that our filtering method is robust over tasks, retrieval approaches and combination methods. As for the non CAS variants, ArtRank gives the best early precision, but the Multiplication gets better precision at later cutoff levels.

The combination methods consistently improve upon the Element retrieval approach, but are far behind the standard Article run. This means that our focused retrieval techniques fail to improve upon an article retrieval approach when it comes to selecting the best point to start reading a document. A closer look at the distribution of BEPs might explain the big difference between the standard Article run and the other runs. The median BEP offset for the 2008 topics is 14 and 49% of all BEPs is at the first character. This shows that choosing the start of the article will in most cases result in a much better document score than any offset further in the document.

2.6 Findings

To sum up, the combination methods seem to be effective in improving early precision. For the official Focused measure, iP[0.01], they lead to substantial improvements over both the Article run and the Element run. The ArtRank method gives the best results for the official measure. Although the Element run scores slightly better at iP[0.00], the combination methods show a good trade off between the good overall precision of the Article run and the good early precision of the Element run. Combining them with the CAS filter improves their overall precision but hurts early precision.

For the Relevant in Context task, all three methods improve upon the Article and Element runs for MAgP. The ArtRank method shows improvement across all cutoff levels. The Multiplication method leads to the highest MAgP scores of the three methods. The CAS filter further improves their effectiveness, although the differences are small for the ArtRank method. Here, the combined runs show the best of both worlds: the good article ranking of the Article run and the more precise retrieval of relevant text within the article of the Element run.

In the Best in Context task, of the three combination methods, ArtRank scores better on early precision, while the other two methods do better at later cutoff levels. However, no focused retrieval method comes close to the effectiveness of the pure Article run. With most of the BEPs at, or *very* close to, the start of the article, there seems to be little need for focused access methods for the Wikipedia collection. This result might be explained by the nature of the collection. The Wikipedia collection contains many short articles, where the entire article easily fits on a computer screen, and are all focused on very specific topics. If any text in such a short article is relevant, it usually makes sense to start reading at the start of the article.

Finally, the CAS filtering method shows to be robust over all tasks and focused retrieval methods used here, leading to consistent and substantial improvements upon the non CAS filtered variants.

3 Book Track

For the Book Track our aims were two-fold: 1) to investigate the effectiveness of using book level evidence for page level retrieval, and 2) experiment with using Wikipedia as a rich resource for topical descriptions of the knowledge found in books, to mediate between user queries and books in the INEX Book Track collection.

We use Indri [28] for our retrieval experiments, with default settings for all parameters. We made one index for both book and page level, using the Krovetz stemmer, no stopword removal, and created two base runs, one at the *book* level and one at the *page* level.

3.1 Book Retrieval Task

Koolen et al. [18] have used Wikipedia as an intermediary between search queries and books in the INEX Book collection. They experimented with using the link distance between so called *query* pages—Wikipedia pages with titles exactly matching

the queries—and *book* pages—each book in the collection is associated with one or more Wikipedia pages based on document similarity—as external evidence to improve retrieval performance. We adopt this approach with the aim to investigate its effectiveness 1) on queries that have no exact matching Wikipedia page and 2) in the context of focussed retrieval, namely, the Page in Context task.

We obtained the *query* pages by sending each query to the online version of Wikipedia and choosing the first returned result. If the query exactly matches a Wikipedia page, Wikipedia automatically returns that page. Otherwise, Wikipedia returns a results list, and we pick the top result. The idea is that most search topics have a dedicated page on Wikipedia. With the 70 topics of the 2008 collection, we found dedicated Wikipedia pages for 23 queries (38.6%). The *book* pages are obtained by taking the top 100 $tf.idf$ terms of each book (w.r.t. the whole collection) as a query to an Indri index of all Wikipedia pages.[1] Next, we computed the link distance between *query* pages and *book* pages by applying a random walk model on the Wikipedia link graph to obtain a measure of closeness between these pages. Books associated with Wikipedia pages closer in the link graph to the *query* page have a higher probability of being relevant [18]. We then combine these closeness scores with the retrieval scores from an Indri run.

The probability of going from node j at step s from the *query* node to node k is computed as:

$$P_{s+1|s}(k|j) = P_{s|s-1}(j) * \frac{l_{jk}}{l_j} \tag{4}$$

where l_{jk} is the number of links from node j to node k, l_j is the total number of links from node j and $P_{s|s-1}(j)$ is the probability of being at node j after step s.

To combine content based retrieval scores with external evidence, Craswell et al. [3] guessed transformation functions from looking at distributions of log odds estimates for different features. URL length, link indegree and click distance (the minimum of clicks to the page from a root page) were modelled by sigmoid functions, leading to substantial improvements when combined with a BM25 baseline. We choose a standard sigmoid function for the transformation:

$$sigmoid(b, q) = \frac{1}{1 + e^{-cl(b,q)}} \tag{5}$$

where $cl(b, q)$ is the closeness score for book b and query q. The sigmoid function ensures that at the low end of the distribution, where there is no relation to relevance, the closeness scores are transformed to values very close to 0.5 (a closeness score of zero would be transformed to 0.5). Close to 1, the closeness scores rapidly increase to 0.73. Thus, only the books at the high end of the distribution receive a boost. We combine this with Indri's retrieval score by simple addition:

$$S(b, q) = Indri(b, q) + sigmoid(b, q) \tag{6}$$

The INEX 2007 Book Track test collection contains only judgements on the book level, with shallow pools – an average of 15.7 books judged for each of the 250 topics. Although there are only 70 topics in the 2008 topic set, the test collection should contain

[1] This is based on the Wikipedia dump of 12 March, 2008.

more judgements per topic and even judgements on the page level. This will allow for a much more detailed analysis of the effectiveness of using Wikipedia as an intermediary between search queries and books.

3.2 Page in Context

As in the Ad Hoc Track, we experiment with methods of re-ranking the *page* level runs using *book* level evidence. Because Indri scores are always negative (the log of a probability, i.e. ranging from $-\infty$ to 0), combining scores can lead to unwanted effects (page score + book score is lower than page score alone). We therefore transform all scores back to probabilities by taking the exponents of the scores. We experimented with the following three methods.

1. *Plus*: add exponents of page score and book score (if the book is not retrieved, use only page score).
2. *Multiplication*: multiply exponents of page and book scores (if book is not retrieved, discard page).
3. *BookRank*: use book score for all retrieved pages.

The 2008 Book Track assessment still have to take place at the time of writing, so we cannot give any results on our submitted runs.

4 Entity Ranking

In the entity ranking track, our aim is to explore the relations and dependencies between Wikipedia pages, categories and links. For the entity ranking task we have looked at some approaches that proved to be successful in last year's entity ranking and ad hoc track. In these tracks it has been shown that link information can be useful. Kamps and Koolen [14] use link evidence as document priors, where a weighted combination of the number of incoming links from the entire collection and the number of incoming links from the retrieved results for one topic is used. Tsikrika et al. [29] use random walks to model multi-step relevance propagation from entities to their linked entities. For the entity ranking track specifically also the category assignments of entities can be used. Vercoustre et al. [30] use the Wikipedia categories by defining similarity functions between the categories of retrieved entities and the target categories. The similarity scores are estimated using lexical similarity of category names. We implemented, extended and combined the aforementioned approaches..

4.1 Model

We would like to create a model that exploits both link and category information and try to find a natural way of combining these different sources of information.

Category information Although for each topic one or a few target categories are provided, relevant entities are not necessarily associated with these provided target categories. Relevant entities can also be associated with descendants of the target category or other similar categories. Therefore, simply filtering on the target categories is not sufficient. Also, since Wikipedia pages are usually assigned to multiple categories, not all categories of an answer entity will be similar to the target category. We calculate for each target category the distances to the categories assigned to the answer entity. To calculate the distance between two categories, we tried three options. The first option (binary distance) is a very simple method: the distance is 0 if two categories are the same, and 1 otherwise. The second option (contents distance) calculates distances according to the contents of each category, the third and option (title distance) calculates a distance according to the category titles. For the title and contents distance, we need to calculate the probability of a term occurring in a category. To avoid a division by zero, we smooth the probabilities of a term occurring in a category with the background collection:

$$P(t_1, ..., t_n|C) = \sum_{i=1}^{n} \lambda P(t_i|C) + (1 - \lambda)P(t_i|D) \tag{7}$$

where C, the category, consists either of the category title to calculate title distance, or of the concatenated text of all pages belonging to that category to calculate contents distance. D is the entire wikipedia document collection, which is used to estimate background probabilities.

We estimate $P(t|C)$ using a parsimonious model [10] that use an iterative EM algorithm as follows:

$$\begin{aligned} &\text{E-step:} && e_t = tf_{t,C} \cdot \frac{\alpha P(t|C)}{\alpha P(t|C) + (1 - \alpha)P(t|D)} \\ &\text{M-step:} && P(t|C) = \frac{e_t}{\sum_t e_t}, \text{i.e. normalize the model} \end{aligned} \tag{8}$$

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:

$$S_{cat}(C_d|C_t) = -D_{KL}(C_d|C_t) = -\sum_{t \in D} \left(P(t|C_t) * \log \left(\frac{P(t|C_t)}{P(t|C_d)} \right) \right) \tag{9}$$

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, corresponding to the smallest distance, from the scores $S(C_d|C_t)$, the scores for the distances from the categories of the document to the target category.

In contrast to [30], where a ratio of common categories between the categories associated with an answer entity and the provided target categories is calculated, we take for each target category only the minimal distance of the distances from the answer entity categories 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:

$$S_{cat}(d|QT) = \sum_{C_t \in QT} \underset{C_d \in d}{\operatorname{argmax}} S(C_d|C_t) \tag{10}$$

List Completion The second task in the entity ranking track is list completion. Instead of the target category, for each topic a few relevant examples entities are given. We treat all categories assigned to the example entities as target categories. Our approach for using the link and the category information is the same as before. But to get the final score of an article in relation to a topic, we use two variants. The first one is:

$$S_{Sum}(d|QT) = \sum_{ex \in QT} \sum_{C_{ex} \in ex} \underset{C_d \in d}{\operatorname{argmax}} S_{cat}(C_d|C_{ex}) \tag{11}$$

In the second variant $S_{Max}(d|QT)$, instead of summing the score of each example category, we only take the maximum score i.e. shortest distance for all example categories of the entity examples to one of the categories of the document. Furthermore, we use the example entities for explicit relevance feedback through query expansion.

Link Information We implement two options to use the link information: relevance propagation and document link degree prior. For the document link degree prior we use the same approach as in [14]. The prior for a document d is:

$$P_{Link}(d) = 1 + \frac{Indegree_{Local}(d)}{1 + Indegree_{Global}(d)} \tag{12}$$

The local indegree is equal to the number of incoming links from within the top ranked documents retrieved for one topic. The global indegree is equal to the number of incoming links from the entire collection.

The second use of link information is through relevance propagation from initially retrieved entities, as was done last year in the entity ranking track by Tsikrika et al. [29].

$$P_0(d) = P(q|d) \tag{13}$$

$$P_i(d) = P(q|d)P_{i-1}(d) + \sum_{d' \to d} (1 - P(q|d'))P(d|d')P_{i-1}(d') \tag{14}$$

Probabilities $P(d|d')$ are uniformly distributed among all outgoing links from the document. Documents are ranked using a weighted sum of probabilities at different steps:

$$P(d) = \mu_0 P_0(d) + (1 - \mu_0) \sum_{i=1}^{K} \mu_i P_i(d) \tag{15}$$

For K we take a value of 3, which was found to be the optimal value last year. We try different values of μ_0 and distribute $\mu_1 ... \mu_K$ uniformly.

Table 4: Document link degree prior results

# docs	μ	MAP	P10
Baseline		0.1840	0.1920
50	0.6	0.1898^{-}	**0.2040**$^{-}$
50	0.5	0.1876^{-}	0.2000^{-}
100	0.7	0.1747^{-}	0.2000^{-}
100	0.3	0.1909^{-}	0.1920^{-}
500	0.5	**0.1982**$^{\circ}$	0.2000^{-}
500	0.3	0.1915^{-}	**0.2040**$^{\circ}$
1,000	0.5	0.1965^{-}	0.1960^{-}
1,000	0.4	0.1965°	0.2000^{-}

Table 5: Category distances results

Dist.	Weight	MAP	P10
Binary	0.1	0.2145^{-}	0.1880^{-}
Cont.	0.1	0.2481$^{\bullet}$	0.2320°
Title	0.1	0.2509°	0.2360°
Cont.	0.05	**0.2618$^{\bullet}$**	**0.2480$^{\bullet}$**
Title	0.05		

Note: *Significance of increase over baseline according to t-test, one-tailed, at significance levels 0.05 ($^{\circ}$), 0.01 ($^{\bullet}$), and 0.001 ($^{\bullet}$).*

Combining Information Finally, we have to combine our different sources of information. We start with our baseline model which is a standard language model. We have two possibilities to combine information. We can make a linear combination of the probabilities and category score. All scores and probabilities are calculated in the log space, and then a weighted addition is made. Secondly, we can use a two step model. Relevance propagation takes as input initial probabilities. Instead of the baseline probability, we can use the scores of the run that combines the baseline score with the category information. Similarly, for the link degree prior we can use the top results of the baseline combined with the category information instead of the baseline ranking.

4.2 Experiments

For our training data we use the 25 genuine entity ranking test topics that were developed for the entity ranking track last year. Since no results are available on the test data yet, we only report on our results on the training data.

For our baseline run and to get initial probabilities we use the language modeling approach with Jelinek-Mercer smoothing, Porter stemming and pseudo relevance feedback as implemented in Indri [28] to estimate $P(d|q)$. We tried different values for the smoothing λ, and $\lambda = 0.1$ gives the best results, with a MAP of 0.1840 and a P10 of 0.1920. For the document link degree prior we have to set two parameters: μ, that determines the weight of the document prior. For μ we try all values from 0 to 1 with steps of 0.1. The weight of the baseline is here $(1 - \mu)$. Only values of μ that give the best MAP and P10 are shown in Table 4. We have to decide how many documents are used to calculate the document prior, and look at 50, 100, 500 and 1,000 documents.

The results of using category information are summarized in Table 5. The weight of the baseline score is 1.0 minus the weight of the category information. For all three distances, a weight of 0.1 gives the best results. In addition to these combinations, we also made a run that combines the original score, the contents distance and the title distance. When a single distance is used, the title distance gives the best results. The combination of contents and title distance gives the best results overall.

In our final experiments we try to combine all information we have, the baseline score, the category and the link information. Firstly, we combine all scores by making

Table 6: Results linear combination

Link Info	Weight	MAP	P10
Prior	1.0	0.2564$^{\circledcirc}$	0.2680$^{\circledcirc}$
Prior	0.5	0.2620$^{\circledcirc}$	0.2560$^{\circledcirc}$
Prop.	0.1	**0.2777**$^{\circledcirc}$	**0.2720**$^{\circledcirc}$

Table 7: Results two step model

Link info	Weight	MAP	P10
Prior	0.5	0.2526$^{\circledcirc}$	0.2600$^{\circledcirc}$
Prop.	0.2	0.2588$^{\circledcirc}$	**0.2960** $^{\bullet}$
Prop.	0.1	**0.2767**$^{\circledcirc}$	0.2720$^{\circledcirc}$

Table 8: Feedback results

RF	PRF	MAP	P10
No	No	0.1409	0.1240
Yes	No	**0.1611**	0.1600
Yes	Yes	0.1341	**0.1960**

Table 9: List Completion results

Dist.	Weight	$S(A\|QT)$	C_t	MAP	P10
Baseline LC				0.1611	0.1600
Cont.	0.1	Sum	No	0.2385$^{\circledcirc}$	0.2520°
Cont.	0.9	Sum	Yes	0.2467$^{\bullet}$	0.2560°
Cont.	0.2	Max	No	0.1845^{-}	0.2360^{-}
Title	0.1	Sum	No	0.2524$^{\circledcirc}$	0.2640°
Title	0.9	Sum	Yes	**0.2641**$^{\bullet}$	**0.2760**$^{\circ}$
Title	0.5	Max	No	0.1618^{-}	0.2080^{-}
Cont. Title	0.05 0.05	Sum	No	0.2528$^{\bullet}$	0.2640°

a linear combination of the scores and probabilities (shown in Table 6). The best run using category information (weight contents = 0.05, and weight title = 0.05) is used in combination with the link information. Secondly, we combine the different sources of information by using the two step model (see Table 7). Link information is mostly useful to improve early precision, depending on the desired results we can tune the parameters to get optimal P10, or optimal MAP. Relevance propagation performs better than the document link degree prior in both combinations.

For the list completion task, we can also use the examples for relevance feedback. To evaluate the list completion results, we remove the example entities from our ranking. Applying explicit and pseudo relevance feedback leads to the results given in Table 8. Additional pseudo relevance feedback after the explicit feedback, only improves early precision, and harms MAP. We take the run using only relevance feedback as our baseline for the list completion task.

When we look at the previous entity ranking task, the largest part of the improvement comes from using category information. So here we only experiment with using the category information, and not the link information. We have again the different category representations, content and category titles. Another variable here is how we combine the scores, either add up all the category scores ($S_{Sum}(A|QT)$) or taking only the maximum score ($S_{Max}(A|QT)$). Not part of the official task, we also make some runs that use not only the categories of the example entities, but also the target category(ies) provided with the query topic. In Table 9 we summarize some of the best results. The combination of contents and title distance, does not lead to an improvement over using only the title distance. The maximum score does not perform as well as the summed scores. We use all categories assigned to the entity examples as target categories, but some of these categories will not be relevant to the query topic introducing noise in the target categories. When the scores are summed, this noise is leveled out, but when only the maximum score is used it can be harmful.

4.3 Findings

We have presented our entity ranking approach where we use category and link information. Category information is the factor that proves to be most useful and we can do more than simply filtering on the target categories. Category information can both be extracted from the category titles and from the contents of the category. Link information can also be used to improve results, especially early precision, but these improvements are small.

5 Interactive Experiments

In this section, we discuss our participation in the INEX 2008 Interactive Track (iTrack). For the Interactive Track, we participated in the concerted experiment designed by the organizers. The overall aim of the iTrack is to understand the how users interact with structured XML documents; the specific aims of the INEX 2008 Interactive Track were to investigate the impact of the task context. Two types of simulated tasks were defined that are believed to represent typical information needs of Wikipedia-users: fact-finding tasks and research tasks. The track is set up to investigate whether the different task types lead to different information interaction: for example, a test-person may prefer different levels of granularity to view documents, different numbers of results, etc. In addition, other factors that may impact the test person's information seeking behavior and satisfaction, such as her knowledge of and interest in the task at hand, are also recorded

As in previous years, the Daffodil system is used for element retrieval. The system returns elements of varying granularity based on the hierarchical document structure. All elements coming from the same document are grouped together in the hitlist (as in the Ad Hoc Track's Relevant in Context task). Any result in the hitlist is clickable and shows the corresponding part of the document in the result display. In addition, all elements matching the query are shown with background highlighting in the result display. For further information about the track set-up we refer to [22].

5.1 Approach

We recruited eight test persons in which seven of them completed the experiment. One test person failed to complete the experiment due to system failure. Hence, our results are based on the data from seven test persons. The organizers provided six simulated search tasks corresponding to two different task types. The first type is *fact-finding*: search tasks that request specific information for a topic. An example fact-finding task is:

> *As a frequent traveler and visitor of many airports around the world you are keen on finding out which is the largest airport. You also want to know the criteria used for defining large airports.*

The second type is *research*: search tasks that required broad information of a topic and the information can only be found by collecting information from several documents. An example research task is:

Table 10: Post-task questionnaire, with answers on a 5-point scale (1-5).

Q3.1: How understandable was the task?
Q3.2: How easy was the task?
Q3.3: To what extent did you find the task similar to other searching tasks that you typically perform?
Q3.6: Are you satisfied with your search results?
Q3.7: How relevant was the information you found?
Q3.8: Did you have enough time to do an effective search?
Q3.9: How certain are you that you completed the task?

Table 11: Post-task responses on searching experience: mean scores and standard deviations (in brackets)

Type	Q3.1	Q3.2	Q3.3	Q3.6	Q3.7	Q3.8	Q3.9
All tasks	4.50 (0.65)	3.29 (1.38)	3.71 (1.27)	3.14 (1.61)	3.29 (1.59)	3.14 (1.35)	2.86 (1.41)
Fact Finding	4.71 (0.49)	3.00 (1.91)	3.43 (1.72)	2.43 (1.81)	2.43 (1.81)	2.86 (1.77)	2.57 (1.81)
Research	4.29 (0.76)	3.57 (0.53)	4.00 (0.58)	3.86 (1.07)	4.14 (0.69)	3.43 (0.79)	3.14 (0.90)

You are writing a term paper about political processes in the United States and Europe, and want to focus on the differences in the presidential elections of France and the United States. Find material that describes the procedure of selecting the candidates for presidential elections in the two countries.

Each test person chose and worked with one simulated task from each category.

5.2 Results

We provide an initial analysis of the data gathered through questionnaires and log files. We will limit our attention here to the post-task questionnaire (Q3, selected questions are shown in Table 10) and the corresponding log data.

The responses of the test persons are summarized in Table 11. If we look at the responses over all tasks the average response varies from 2.86 to 4.50 signaling that the test persons were reasonably positive. We also look at the responses for each task type. Here we see that test persons understood both tasks very well (Q3.1). Fact finding receives higher responses on average, which makes sense given the nature of the simulated tasks and thereby confirms that the chosen simulated tasks represent the particular task types. For all other questions in Table 11, the research task type is receiving higher responses on average. That is, the research task was regarded easier (Q3.2) and more similar to other searching task (Q3.3) compare to the fact finding task. Admittedly, this may be a result of our choice of test persons who all have an academic education. Moreover, test persons were more satisfied with the search results provided by the system (Q3.6) for the research task. A possible explanation is that the research tasks are more open-ended than the fact finding task where test persons need to find a specific and precise answer. Hence, additional material provided by the system may be more useful in the research task context. This explanation is supported by the response when asked about the relevancy of the found information (Q3.7). Test persons believed that they found more relevant results for the research task.

Next, we look at the time test persons spent on each task. On the question whether there was enough time for an effective search (Q3.8), responses for the fact finding tasks were lower than for the research tasks. This is rather surprising since the fact-finding task is less open-ended. As a case in point, the log data shows that, on average, test persons spent 9 minutes and 17 seconds on a fact finding task and 11 minutes 17 seconds on a research task. Almost all test persons did not use the maximally allotted time of 15 minutes per task. A possible explanation is that the system did not support them well enough in finding relevant results for fact-finding (Q3.6), or at least that they expected the system to do better (which may be an unrealistic expectation based on searching for similar questions on the Web at large using Internet search engines). This is consistent with the assessment of task completion (Q3.9) where, on average, test persons were less certain that they completed the fact finding task compared to the research task. Also note that the standard deviation for fact finding task in almost all questions are larger than the research task. A possible explanation is again that several test persons were not satisfied with the results they found when completing the fact finding task. We look forward to analysing this hypothesis against the data collected in the other experiments, hoping to reveal whether it can indeed be attributed to the impact of the task type.

5.3 Findings

We reported the result of the Interactive Track experiment based on seven test persons using the post-task questionnaire and corresponding log files. We found that our test persons spent more time on completing the research task in comparison to fact finding task. We also found that test persons regarded the research task easier, were more satisfied with the search result and found more relevant information for the research task. This is plausibly related to the task type, where test persons regard more information as relevant or useful when searching for a more open-ended research task. Fact finding tasks require a more specific and precise answer, which may diminish the additional value of exploring a wide range of search results. We plan to analyze the data in greater detail, e.g., by examining possible differences between tasks types also for the data collected by other groups.

6 Link Detection Experiments

In this section, we discuss our participation in the Link The Wiki (LTW) track. LTW is aimed at detecting or discovering missing links between a set of topics, and the remainder of the collection, specifically detecting links between an origin node and a destination node, hence effectively establishing cross-links between Wikipedia articles. This year the track consisted of two tasks. What both tasks had in common was that it consisted of 2 sub-tasks; the detection of links from an ‘orphan’ (outgoing) and to an ‘orphan’ (incoming). The issue of link density and link repetition as mentioned in [33] has not been addressed, henceforth we restricted our experimentation to detecting unique cross-links.

The first task was a continuation of the track of last year with the detection of links between whole articles where no anchor or Best Entry Point (BEP) was required. A

difference with the task of the previous year was the number of 'orphan' topics, namely a random sample of 6,600 topics which equals about 1% of the total collection. Existing links in origin nodes were removed from the topics, making these articles 'orphans.' A threshold of 250 was set for both the number of incoming and outgoing links.

The second task used 50 orphan topics which were submitted by the track participants and went further than link detection on the article level as the BEP was required. Another requirement for these topics was that a plausible number of outgoing and incoming links is 50. Multiple BEPs were allowed; each anchor was allowed to have 5 BEPs.

6.1 Experiments with Detection of Article-to-Article Links

Information Retrieval methods have been employed to automatically construct hypertext on the Web [1, 2]. Previous research with generic and learning approaches of link detection in the Wikipedia are for example [6, 7, 8, 12, 13, 21]. General information retrieval techniques were used in [6, 7, 8, 13]. An approach that used link structures to propagate more likely 'linkable' anchor texts was presented in [12], and a machine learning approach with standard classifiers was presented in [21].

We have chosen to employ a collection-independent approach with IR techniques as outlined in [6] and continue the experimentation with that approach and put it more under the test. This means we do not rely on any learning, heavy heuristics or existing link structures in the Wikipedia. Our approach is mostly based on the assumption that to detect whether two nodes are implicitly connected, it is necessary to search the Wikipedia pages for some text segments that both nodes share. Usually it is only one specific and extract string [1]. One text segment is defined as a single line, and a string that both nodes share is a potentially relevant substring. Only relevant substrings of at least 3 characters length are considered in our approach, because anchor texts of 3 characters or less do not occur frequently, and to prevent detecting too many false positives.

We adopt a *breadth m–depth n* technique for automatic text structuring for identifying candidate anchors and text node, i.e. a fixed number of documents accepted in response to a query and fixed number of iterative searches. So the similarity on the document level and text segment level is used as evidence. We used the whole document as a query with the standard Vector Space Model (VSM) implementation of Lucene [20], i.e., for a collection D, document d, query q and query term t:

$$sim(q,d) = \sum_{t \in q} \frac{tf_{t,q} \cdot idf_t}{norm_q} \cdot \frac{tf_{t,d} \cdot idf_t}{norm_d} \cdot coord_{q,d} \cdot weight_t, \tag{16}$$

where

$$\begin{aligned} tf_{t,X} &= \sqrt{\text{freq}(t,X)} \\ idf_t &= 1 + \log \frac{|D|}{\text{freq}(t,D)} \\ norm_q &= \sqrt{\sum_{t \in q} tf_{t,q} \cdot idf_t{}^2} \\ norm_d &= \sqrt{|d|} \\ coord_{q,d} &= \frac{|q \cap d|}{|q|}. \end{aligned}$$

Moreover, we also used the standard Language Modeling (LM) framework of ILPS-Lucene [11], which we already discussed in Section 2.

Before the actual link detection process starts, we do some pre-processing by extracting for each topic the title enclosed with the <name> tag with a regular expression and store that in a hash-table for substring matching. We do not apply case-folding, but we do remove any existing disambiguation information put between brackets behind the title.

We do not assume that links are reciprocal, so we have different approaches for detecting outgoing and incoming links, though we set a threshold of 250 for both type of links and do not allow duplicated links. Links also appear locally within an article to improve navigation on that page, but this was outside the scope of the LTW track.

- There is an *outgoing link* for an 'orphan' topic when the title of a 'foster' article occurs in the orphan topic.
- There is an *incoming link* for an orphan when the title of the orphan occurs in a foster topic.

We submitted 3 official runs based on this generic approach:

`Amsterdam_Turfdraagsterpad(UvA)_a2a_1` The whole orphan article is used as a query, where the VSM is used. The pool of plausible 'foster' (candidate) articles is the top 300 of the ranked list returned by this query. This is our baseline run.

`Amsterdam_Turfdraagsterpad(UvA)_a2a_2` The term frequencies of the orphan article are conflated, and the resulting bag of words is used as query with LM with default settings. The pool of plausible 'foster' (candidate) articles is the top 300 of the ranked list returned by this query.

`Amsterdam_Turfdraagsterpad(UvA)_a2a_3` The whole orphan article is used as a query, where the VSM is used. The pool of plausible 'foster' (candidate) articles is the top 500 of the ranked list returned by this query.

6.2 Experiments with Detection of Article-to-BEP Links

We submitted 4 runs for the article-to-BEP task. For all of these runs, we assume that the BEP is always the start of the article, thus the offset is always 0. Another difference with the first task was that actual anchor text had to be specified using the File-Offset-Length (FOL) notation. Multiple BEPs per anchor were only computed for the run `Amsterdam_Turfdraagsterpad(UvA)_a2bep_5`. The 4 official submitted Article-to-BEP runs were:

`Amsterdam_Turfdraagsterpad(UvA)_a2bep_1` The whole orphan article is used as query with the VSM, and the top 300 results are used to find potential cross-links.

`Amsterdam_Turfdraagsterpad(UvA)_a2bep_2` The term frequencies of the orphan article are conflated, and the resulting bag of words is used as query with LM with default settings. The pool of plausible 'foster' (candidate) articles is the top 300 of the ranked list returned by this query.

`Amsterdam_Turfdraagsterpad(UvA)_a2bep_3` The whole orphan article is used as query with the VSM. The top 50 ranking articles is harvested. Each of these article is used again as a query to retrieve its top 6 results. This results in a potential pool of 300 foster articles.

Table 12: Official results of the Article-to-Article runs for the Link The Wiki Track

Run	Links	MAP	P@5	Rank
Amsterdam_Turfdraagsterpad(UvA)_a2a_1	In	0.339272695	0.833743098	16/25
Amsterdam_Turfdraagsterpad(UvA)_a2a_2	In	0.287957616	0.832444188	22/25
Amsterdam_Turfdraagsterpad(UvA)_a2a_3	In	**0.357581856**	**0.837737848**	15/25
Amsterdam_Turfdraagsterpad(UvA)_a2a_1	Out	0.107164298	0.284862095	15/25
Amsterdam_Turfdraagsterpad(UvA)_a2a_2	Out	**0.108885207**	**0.293142884**	14/25
Amsterdam_Turfdraagsterpad(UvA)_a2a_3	Out	0.101743892	0.268992346	17/25

Table 13: Official results of the Article-to-BEP runs for the Link The Wiki Track

Run	Links	MAP	P@5	Rank
Amsterdam_Turfdraagsterpad(UvA)_a2bep_1	In	**0.234947342**	0.932647059	9/30
Amsterdam_Turfdraagsterpad(UvA)_a2bep_2	In	0.16157935	**0.949807692**	22/30
Amsterdam_Turfdraagsterpad(UvA)_a2bep_3	In	0.156616929	0.933123249	25/30
Amsterdam_Turfdraagsterpad(UvA)_a2bep_5	In	**0.234947342**	0.932647059	8/30
Amsterdam_Turfdraagsterpad(UvA)_a2bep_1	Out	0.097272945	0.524529751	20/30
Amsterdam_Turfdraagsterpad(UvA)_a2bep_2	Out	0.087151118	0.48392250	22/30
Amsterdam_Turfdraagsterpad(UvA)_a2bep_3	Out	0.091064252	0.604179122	21/30
Amsterdam_Turfdraagsterpad(UvA)_a2bep_5	Out	**0.143689485**	**0.748596188**	14/30

Amsterdam_Turfdraagsterpad(UvA)_a2bep_5 This run is similar to the first Article-to-BEP run, but we expanded this run by allowing more than 1 BEP for each anchor. We use the depth-first strategy, and the broader-narrower conceptualization of terms by re-grouping the extracted list of titles based on a common substring. For example, the anchor text *Gothic* could refer to the topic "Gothic," but also to topics with the titles *"Gothic alphabet"*, *"Gothic architecture"*, *"Gothic art"*, *"Gothic Chess"*, and so on.

6.3 Experimental Results

The current experimental results are evaluated against the set of existing links (in the un-orphaned version of the topics) both for the sample of 6,600 topics in the first task as well as the 50 topics of the article-to-BEP task. The results of the user assessments of the 50 topics have not been released yet of this writing. The results of our official submissions are depicted in Table 12 for the first task with the detection of links on the article-level and in Table 13 for the article-to-BEP. The performance was measured with the Mean Average Precision (MAP) and the Precision at rank like the precision at 5 percent (P@5). Moreover, our runs are put in context by ranking them in the list that includes the runs of the other participants based on the MAP score.

These results, especially the sub-optimal results for the outgoing links and the general results on the article-level, warrant some reflection on several limitations of our approach. We did exact string matching with the titles of the potential foster topics and did no apply case-folding or any kind of normalization. This means we could have incorrectly discarded a significant number of relevant foster articles (false negatives); effectively under-generating the outgoing links and under-linking the topics.

We achieved one of the highest early precisions for the incoming links for both tasks. The precision drops later on as the fallout increases, which hurts the eventual MAP score—this suggests that the ranked list of results should be cut-off earlier, and that more relevant articles should be retrieved in the top. The limitations of using whole document as query, combined with the VSM, has been tested. We also tested the LM method with default parameter values for compiling such a list by using a conflated set of terms from an orphan topic, however, this has proven not to be very effective for the accuracy of the detection of the incoming links, and also hardly affects the accuracy of the detection of the outgoing links.

6.4 Findings

In summary, we continued with our experimentation with the Vector Space Model, conducted some tests with Language Modeling, and simple string processing techniques for detecting missing links in the Wikipedia. The link detection occurred in 2 steps: first, a relevant pool of foster (candidate) articles is collected; second, substring matching with the list of collected titles to establish an actual link. We used entire orphaned articles as query. Clearly, we showed the constraints of this approach, especially on the article level. Our Article-to-BEP runs are also dependent on this first step. However, we found that we can significantly improve the accuracy of the detection of our outgoing links by generating multiple BEPs for an anchor.

7 XML Mining Track

Previous years of the XML mining track have explored the utility of using XML document structure for classification accuracy. It proved to be difficult to obtain better performance [5]. This year the data consist of a collection of wikipedia XML documents that have to be categorized into fairly high-level wikipedia categories and the link structure between these documents. Link structure has been found to be a useful additional source of information for other tasks such as ad hoc retrieval [14] and entity ranking (see Section 4). Our aim at the XML Mining Track is to examine whether link structure can also be exploited for this classification task.

7.1 Classification Model

For our baseline classification model we use a classical Naive Bayes model [23]. The probability of a category given a document is:

$$P(cat|d) = \frac{P(d|cat) * P(cat)}{P(d)} \tag{17}$$

Since $P(d)$ does not change over the range of categories we can omit it. For each document the categories are ranked by their probabilities, and the category with the highest probability is assigned to the document:

$$\begin{aligned} ass\_cat(d) &= \arg\max_{cat \in cats} P(d|cat) * P(cat) \\ &= \arg\max_{cat \in cats} P(t_1|cat) * P(t_2|cat) * .. * P(t_n|cat) * P(cat) \end{aligned} \tag{18}$$

where $t_i ... t_n$ are all terms in a document. The probability of a term occurring in a category is equal to its term frequency in the category divided by the total number of terms in the category. Feature (term) selection is done according to document frequency. We keep 20% of the total number of features [32]. The link information is used as follows:

$$\begin{aligned} P_0(cat|d) &= P(cat|d) \\ P_1(cat|d) &= \sum_{d \to d'} P(d|d')P(cat|d') \\ P_2(cat|d) &= \sum_{d' \to d''} P(d|d'')P(cat|d'') \end{aligned} \tag{19}$$

where d' consist of all documents that are linked to or from d, and d'' are all documents that are linked to or from all documents d'. The probabilities are uniformly distributed among the incoming and/or outgoing links. The final probability of a category given a document is now:

$$P'(cat|d) = \mu P_0(cat|d) + (1-\mu)(\alpha P_1(cat|d) + (1-\alpha)P_2(cat|d)) \tag{20}$$

The parameter μ determines the weight of the original classification versus the weight of the probabilities of the linked documents. Parameter α determines the weight of the first order links versus the weight of the second order links.

7.2 Experiments

Documents have to be categorized into one of fifteen categories. For our training experiments, we use 66% of the training data for training, and we test on the remaining 33%. We measure accuracy, which is defined as the percentage of documents that is correctly classified, which is equal to micro average recall. Our baseline Naive Bayes model achieves an accuracy of 67.59%. Macro average recall of the baseline run is considerably lower at 49.95%. All documents in the two smallest categories are misclassified. Balancing the training data can improve our macro average recall.

When we use the link information we try three variants: do not use category information of linked data, use category information of the training data, and always use category information of linked data. Other parameters are whether to use incoming or outgoing links, μ and α. For parameter μ we tried all values from 0 to 1 with steps of 0.1, only the best run is shown. The results are given in Table 14. The accuracy of the runs using link information is at best only marginally better than the accuracy of the baseline. This means that the difficult pages, which are misclassified in the baseline model, do not profit from the link information. The links to or from pages that do not clearly belong to a category and are misclassified in the baseline run, do not seem to contribute to classification performance. These linked pages might also be more likely to belong to a different category.

On the test data we made two runs, a baseline run that achieves an accuracy of 69.79%, and a run that uses in- and outlinks, $\alpha = 0.5$ and $\mu = 0.4$, with an accuracy of 69.81%. Again the improvement in accuracy when link information is used is only marginal.

Table 14: Training Classification results

Link info	α	Inlinks μ	Inlinks Accuracy	Outlinks μ	Outlinks Accuracy	In- and Outlinks μ	In- and Outlinks Accuracy
Baseline			0.6759		0.6759		0.6759
None	0.5	0.5	0.6766	1.0	0.6759	0.4	0.6764
None	0.75	1.0	0.6759	1.0	0.6759	1.0	0.6759
None	1.0	1.0	0.6759	1.0	0.6759	1.0	0.6759
Training	0.5	0.5	**0.6793**	0.4	0.6777	0.4	0.6819
Training	0.75	0.5	**0.6793**	0.5	0.6777	0.5	0.6806
Training	1.0	0.6	0.6780	0.5	0.6780	0.6	0.6777
All	0.5	0.5	0.6780	0.3	0.6816	0.4	**0.6858**
All	0.75	0.6	0.6780	0.3	0.6848	0.5	0.6819
All	1.0	0.6	0.6784	0.4	**0.6858**	0.6	0.6787

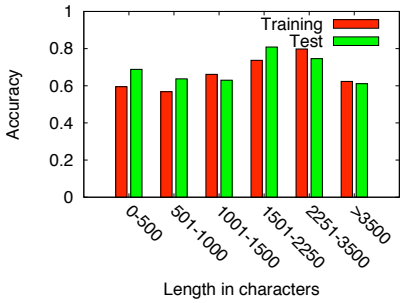


Fig. 1: Accuracy vs. document length

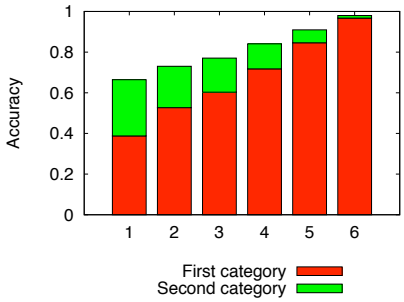


Fig. 2: Accuracy of first two categories

7.3 Analysis

We try to analyze on which kind of pages the most errors are made in our baseline run. Considering the length of pages, shorter pages do not tend to be more difficult than longer pages as can be seen in Figure 1. The difficult pages to classify can be recognized by comparing the output probability of the two highest scoring categories. This is shown in Figure 2 where we divided the training data over 6 bins of approximately the same size sorted by the fraction (P_{cat1}/P_{cat2}).

In our baseline run pages without links also seem to get misclassified more often than pages with in- and/or outlinks (see Figure 3). When link information is available, and we try to use it, there are two sources of error. The first source of error, is that not all linked pages belong to the same category as the page to classify (see Table 15). However, when we classify pages that have links using only the link information, there are some cases where the accuracy on these pages is well above the accuracy of the complete set. To obtain our test data we have used both incoming and outgoing links, which means that almost half of the pages do not belong to the same category as the page to classify. Secondly, we only know the real categories of the pages in the training data, which is only 10% of all data. For all pages in the test data, we estimate the

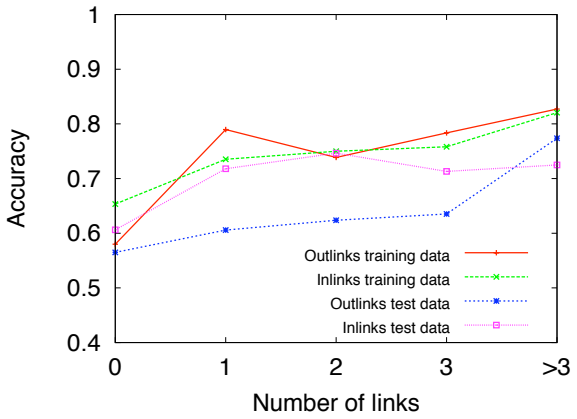


Fig. 3: Accuracy vs. number of links

Table 15: Statistics of training and test data

	# pages				links	% links with same cat.		
Data	total	with inlinks	with outlinks	with links	/page	inlinks	outlinks	links
Training	11,437	2,627 (23%)	5,288 (46%)	5,925 (52%)	0.7	76.8%	41.1%	45.8%
Test	113,366	88,174 (77%)	103,781 (91%)	107,742 (94%)	5.6	77.2%	53.4%	59.0%

probability of each category belonging to that page. With a classification accuracy of almost 70%, this means we introduce a large additional source of error.

7.4 Findings

It is difficult to use link information to improve classification accuracy. A standard Naive Bayes model achieves an accuracy of almost 70%. While link information may provide supporting evidence for the pages that are easy to classify, for the difficult pages link information is either not available or contains too much noise.

8 Discussion and Conclusions

In this paper, we documented our efforts at INEX 2008 where we participated in six tracks: Ad hoc, Book, Entity Ranking, Interactive, Link the Wiki, and XML-Mining.

For the *Ad Hoc Track*, we investigated the effectiveness of combining article and element retrieval methods. We found that the ArtRank method, where the article run determines the article ranking, and the element run determines which part(s) of the text is returned, gives the best results for the Focused Task. For the Relevant in Context Task, the Multiplication method is slightly better than ArtRank and CombSUM, but for the CAS runs, where we filter on a pool of target elements based on the entire topic set, the CombSUM method gives the best performance overall. The combination methods are not effective for the Best in Context Task. The standard article retrieval run is far

superior to any focused retrieval run. With many short articles in the collection, all focused on very specific topics, it makes sense to start reading at the start of the article, making it hard for focused retrieval techniques to improve upon traditional document retrieval. The CAS pool filtering method is effective for all three tasks as well, showing consistent improvement upon the non CAS variants for all measures.

For the *Book Track*, we experimented with the same run combination methods as in the Ad Hoc Track. We also combined a standard content based book retrieval run with external evidence from Wikipedia. Using the idea that Wikipedia covers many of both the topics found in books and searched for by users, and the dense link graph between Wikipedia pages, we use the link distance between pages matching the query and pages matching the content of the books in the collection as measure of topical closeness. The topic assessment phase has yet to start, so we cannot report any results in this paper, but we aim to investigate the effectiveness of combination methods for document and focused retrieval techniques in a book retrieval setting, where the documents are far larger than in the Wikipedia collection. With entire books, providing a good access point to a specific topic might show focused retrieval techniques to be much more effective than traditional document retrieval.

For the *Entity Ranking Track*, we have presented our entity ranking approach where we use category and link information. Category information is the factor that proves to be most useful and we can do more than simply filtering on the target categories. Category information can both be extracted from the category titles and from the contents of the category. Link information can also be used to improve results, especially early precision, but these improvements are small.

For the *Interactive Track*, we found that our test persons spent more time on completing the research task in comparison to fact finding task. We also found that test persons regarded the research task easier, were more satisfied with the search result and found more relevant information for the research task. This is plausibly related to the task type, where test persons regard more information as relevant or useful when searching for a more open-ended research task. Fact finding tasks require a more specific and precise answer, which may diminish the additional value of exploring a wide range of search results.

For the *Link the Wiki Track*, we continued with our experimentation with the Vector Space Model, conducted some tests with Language Modeling, and simple string processing techniques for detecting missing links in the Wikipedia. The link detection occurred in 2 steps: first, a relevant pool of foster (candidate) articles is collected; second, substring matching with the list of collected titles to establish an actual link. We used entire orphaned articles as query. Clearly, we showed the constraints of this approach, especially on the article level. Our Article-to-BEP runs are also dependent on this first step. However, we found that we can significantly improve the accuracy of the detection of our outgoing links by generating multiple BEPs for an anchor.

For the *XML-Mining Track*, our aim was to explore whether we can use link information to improve classification accuracy. Previous years of the XML mining track have explored the utility of using XML document structure for classification accuracy. Link structure has been found to be a useful additional source of information for other tasks such as ad hoc retrieval. Our experiments suggest that it is difficult to use link in-

formation to improve classification accuracy. A standard Naive Bayes model achieves an accuracy of almost 70%. While link information may provide supporting evidence for the pages that are easy to classify, for the difficult pages link information is either not available or too noisy to be promoting the classification accuracy.

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. Khairun Nisa Fachry and Junte Zhang were supported by NWO under grant # 639.072.601.

Bibliography

[1] M. Agosti, F. Crestani, and M. Melucci. On the use of information retrieval techniques for the automatic construction of hypertext. *Information Processing and Management*, 33: 133–144, 1997.

[2] J. Allan. Building hypertext using information retrieval. *Information Processing and Management*, 33:145–159, 1997.

[3] N. Craswell, S. Robertson, H. Zaragoza, and M. Taylor. Relevance weighting for query independent evidence. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 416–423. ACM Press, New York NY, USA, 2005.

[4] L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 40:64–69, 2006.

[5] 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.

[6] K. N. Fachry, J. Kamps, M. Koolen, and J. Zhang. Using and detecting links in wikipedia. In *Proceedings INEX 2007*, volume 4862 of *LNCS*, pages 388–403. Springer Verlag, Heidelberg, 2008.

[7] S. Fissaha Adafre and M. de Rijke. Discovering missing links in wikipedia. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 90–97. ACM Press, New York NY, USA, 2005.

[8] S. Geva. GPX: Ad-hoc queries and automated link discovery in the Wikipedia. In *Proceedings INEX 2007*, volume 4862 of *LNCS*, pages 404–416. Springer Verlag, Heidelberg, 2008.

[9] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, Center for Telematics and Information Technology, University of Twente, 2001.

[10] D. Hiemstra, S. Robertson, and H. Zaragoza. Parsimonious language models for information retrieval. In *Proceedings SIGIR 2004*, pages 178–185. ACM Press, New York NY, 2004.

[11] ILPS. The ILPS extension of the Lucene search engine, 2008. http://ilps.science.uva.nl/Resources/.

[12] K. Y. Itakura and C. L. A. Clarke. University of waterloo at INEX 2007: Adhoc and link-the-wiki tracks. In *Proceedings INEX 2007*, volume 4862 of *LNCS*, pages 417–425. Springer Verlag, Heidelberg, 2008.

[13] D. Jenkinson and A. Trotman. Wikipedia ad hoc passage retrieval and wikipedia document linking. In *Proceedings INEX 2007*, volume 4862 of *LNCS*, pages 426–439. Springer Verlag, Heidelberg, 2008.

[14] J. Kamps and M. Koolen. The importance of link evidence in Wikipedia. In *Advances in Information Retrieval: 30th European Conference on IR Research (ECIR 2008)*, pages 270–282, 2008.

[15] J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. The importance of morphological normalization for XML retrieval. In *Proceedings of the First Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, pages 41–48. ERCIM Publications, 2003.

[16] J. Kamps, M. Koolen, and B. Sigurbjörnsson. Filtering and clustering XML retrieval results. In *Comparative Evaluation of XML Information Retrieval Systems (INEX 2006)*, volume 4518 of *LNCS*, pages 121–136. Springer Verlag, Heidelberg, 2007.

[17] J. Kamps, M. Koolen, and M. Lalmas. Locating relevant text within XML documents. In *Proceedings SIGIR 2008*, pages 847–849. ACM Press, New York NY, USA, 2008.

[18] M. Koolen, G. Kazai, and N. Craswell. Wikipedia Pages as Entry Points for Book Search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining (WSDM 2009)*. ACM Press, New York NY, USA, 2009.

[19] W. Kraaij, T. Westerveld, and D. Hiemstra. The importance of prior probabilities for entry page search. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 27–34. ACM Press, New York NY, USA, 2002.

[20] Lucene. The Lucene search engine, 2008. http://lucene.apache.org/.

[21] D. Milne and I. H. Witten. Learning to link with Wikipedia. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge mining*, pages 509–518. ACM Press, New York NY, USA, 2008.

[22] N. Pharo, R. Nordlie, and K. N. Fachry. The interactive track at INEX 2008. In *This Volume*, 2008.

[23] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47, 2002.

[24] B. Sigurbjörnsson. *Focused Information Access using XML Element Retrieval*. SIKS dissertation series 2006-28, University of Amsterdam, 2006.

[25] 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.

[26] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-Based Approach to XML Retrieval. In *INEX 2003 Workshop Proceedings*, pages 19–26, 2004.

[27] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Mixture models, overlap, and structural hints in XML element retreival. In *Advances in XML Information Retrieval: INEX 2004*, volume 3493 of *LNCS 3493*, pages 196–210, 2005.

[28] 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.

[29] T. Tsikrika, P. Serdyukov, H. Rode, T. Westerveld, R. Aly, D. Hiemstra, and A. P. de Vries. Structured document retrieval, multimedia retrieval, and entity ranking using PF/Tijah. In *Proceedings INEX 2007*, volume 4862 of *LNCS*, pages 306–320. Springer Verlag, Heidelberg, 2008.

[30] A.-M. Vercoustre, J. Pehcevski, and J. A. Thom. Using wikipedia categories and links in entity ranking. In *Proceedings INEX 2007*, volume 4862 of *LNCS*, pages 321–335. Springer Verlag, Heidelberg, 2008.

[31] Wikipedia. The free encyclopedia, 2008. http://en.wikipedia.org/.

[32] K. Williams. Ai::categorizer - automatic text categorization. Perl Module, 2003.

[33] J. Zhang and J. Kamps. Link detection in XML documents: What about repeated links? In A. Trotman, J. Kamps, and S. Geva, editors, *Proceedings of the SIGIR 2008 Workshop on Focused Retrieval*, pages 59–66. University of Otago, Dunedin New Zealand, 2008.

UJM at INEX 2008: pre-impacting of tags weights

Mathias Géry, Christine Largeron and Franck Thollard

Université de Lyon, F-69003, Lyon, France
Université de Saint-Étienne, F-42000, Saint-Étienne, France
CNRS UMR5516, Laboratoire Hubert Curien
{mathias.gery, christine.largeron, franck.thollard}@univ-st-etienne.fr

Abstract. This paper[1] addresses the integration of tags in terms weighting function for focused XML retrieval. Our model allows to consider a certain kind of structural information: tags that represent logical structure (title, section, etc.) as well as tags related to formatting (bold font, centered text, etc.). We first take into account the tags influence by estimating the probability that tags distinguishes terms which are the most relevant. Then, these weights are impacted on terms weighting function using several combining schemes.
Experiments on a large collection during INEX 2008 XML IR evaluation campaign[2] showed that using tags leads to improvements on focused retrieval.

1 Introduction

We consider in this paper the problem of extending the classical probabilistic model [4] that aims at estimating the relevance of a document for a given query through two probabilities: the probability of finding a relevant information and the probability of finding a non relevant information. The model is extended using formatting tags (*e.g.* tag *strong*) and structuring tags (*e.g.* tag *section*). For a given tag we can estimate if it emphasizes terms in relevant documents or term in non relevant part of documents. This work has been presented in [2]. Although the recall was improved by the integration of the tag weights, the results were not that convincing. Even if the estimation of the tag weights must be carefully addressed, it appears that the way such weights are integrated into the final term weight is essential.

We consider in this work different ways of integrating the tag weights in a more general term weighting scheme. We experimented different term weighting functions (namely `ltn`, `ltc`, `BM25`...), together with different integration schemes. We now present the different integration schemes proposed. The experimental results follows.

[1] This work is supported by the project "Web Intelligence", Rhône-Alpes region, cf. http://www.web-intelligence-rhone-alpes.org
[2] INitiative for Evaluation of XML Retrieval. See http://www.inex.otago.ac.nz

2 Tags weights integration schemes: the TTF and CLAW strategies

Having tags weights estimated on a training set, we need now to compute a final score for ranking the elements which answer to a query Q. We first propose to make a linear combination between tags weights and the term weight. In order to take into account all the tags that mark the terms, we average their weights and then combine linearly the term and the tag weights. The first model called[3] f_{claw} is:

$$f_{claw}(m_j) = \sum_{t_{ik} \in m_j / t_i \in Q} w_{ji} \times \frac{\sum_{k/t_{ik}=1} w'_k}{|\{k/t_{ik}=1\}|} \quad (1)$$

where m_j is an XML element (cf. notations in [2]). w_{ji}, the weight of the term t_i in the element m_j is computed by one of the different ranking functions `ltn`, `ltc`, or `BM25`. The term w'_k represents the weight of the tag k. Therefore, the fraction $\frac{\sum_{k/t_{ik}=1} w'_k}{|\{k/t_{ik}=1\}|}$ represents the average of the tag weights that mark term considered.

Following Robertson and *al.* [5] we experiment another way of impacting the tags weights: as before, we compute the average tags weights for each term, and multiply this score at the term frequency step. More formally, the `Tagged Term Frequency`, noted ttf_{ji}, replaces the tf_{ji}, the term frequency of the term t_i in the element e_j in the ranking functions `ltn`, `ltc`, or `BM25`.

$$ttf_{ji} = tf_{ji} \times \frac{\sum_{k/t_{ik}=1} w'_k}{|\{k/t_{ik}=1\}|} \quad (2)$$

3 Experiments

We have experimented these models in a classic IR way (granularity: full articles) as well as in a focused IR way (granularity: XML elements). The experimental protocol is the one proposed in the INEX competition.

3.1 INEX collection

We have used for our experiments the english Wikipedia XML corpus [1] used by the INEX IR campaign since 2006, as it contains an important amount of structured XML data and relevance assessments.

This corpus contains 659,388 articles extracted from the Wikipedia encyclopedia in early 2006. The original Wiki syntax has been converted into XML, using both general tags of the logical structure (article, section, paragraph, title,

[3] f_{claw} stands for `Combining Linearly Average tag-Weights`.

list and item), formating tags (like bold, emphatic) and frequently occurring link-tags. The documents are strongly structured as they are composed of 52 millions of XML elements. Each XML article can be viewed as a tree which contains, on average, 79 elements for an average depth of 6.72. Moreover, whole articles (textual content + XML structure) represent 4.5 Gb while the textual content only 1.6 Gb. The structural information thus represents more than twice the textual one.

3.2 Experimental protocol

The corpus enriched by the INEX 2006 assessments on 114 queries has been used as the training set in order to estimate the tags weights w'_k. Then we have evaluated our approach using the 285 queries of INEX 2008 (135 queries from INEX participants and 150 queries from real users), on which only 70 have been evaluated yet. Our evaluation is based on the main INEX measures ($iP[x]$ the precision value at recall x, AiP the *interpolated average precision*, and $MAiP$ the *interpolated mean average precision* [3]). Note that the main ranking of INEX competition is based on $iP[0.01]$ instead of the overall measure $MAiP$, in order to take into account the importance of precision at low recall levels.

Each run submitted to INEX is a ranked list containing at most 1500 XML elements. Some runs retrieve all the relevant elements among the first 1500 XML returned elements, and some others retrieve only part of them. Note that a limit based on a number of documents (instead of *e.g.* a number of bytes) allows to return more information and therefore favors runs composed by full articles. In order to better consider this fact in recall/precision curves, we have calculated $R[1500]$, the recall at 1500 elements, and S[1500], the size of these 1500 elements (in Mbytes).

3.3 Tags weighting

We have manually selected 14 tags (*title*, *table*, *caption*, *article*, *body*, *section*, *numberlist*, *definitionitem*, *normallist*, *th*, *td*, *tr*, *p*, *row*) in order to define the XML elements to consider. These logical structure tags will be considered during the indexing step and therefore those will define the elements the system will be able to return.

Regarding the other tags (namely the formatting tags), we first selected the 61 tags that appear more than 300 times in the 659,388 documents. We then manually removed 6 tags: article, body (they mark the whole information), br, hr, s and value (considered not relevant).

The weights of the 55 remaining tags were computed according to equation w'_k in [2]. Table 1 presents the top 6 tags and their weights, together with the

weakest 6 ones and their weights. Their frequencies in the whole collection is also given.

Table 1. Weight w'_k of the 6 strongest and 6 weakest tags

Top strongest weights			Top weakest weights		
tag	weight	freq.	tag	weight	freq.
h4	12,32	307	emph4	0,06	940
ul	2,70	3'050	font	0,07	27'117
sub	2,38	54'922	big	0,08	3'213
indentation1	2,04	135'420	em	0,11	608
section	2,01	1'610'183	b	0,13	11'297
blockquote	1,98	4'830	tt	0,14	6'841

4 Results

We have submitted 9 runs to INEX 2008 Ad-Hoc, i.e. 3 runs in each task: focused, relevant in context and best in context.

4.1 Parameters

The parameters of the chosen weighting functions (namely `BM25`) were tuned in order to improve classic retrieval (articles granularity) and focused retrieval (elements granularity). Among the parameters studied to improve the baseline, we can mention the use of a stoplist, the optimization of BM25 parameters ($k_1 = 1.1$ and $b = 0.75$), etc. Regarding the queries, we set up a better "andish" mode and consider *or* and *and*, *etc* .... Some specific parameters (*e.g.* the minimum size of the returned elements) were also tuned for focused retrieval.

Our baseline and all other 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* (structured part of the topics).

4.2 Focused task: baseline

The focused task asks systems to return a ranked list of non-overlapping elements or passages to the user. Our aim was firstly to obtain a strong baseline, secondly to experiment focused retrieval (i.e. elements granularity) against classic retrieval (i.e. full articles granularity), and thirdly to experiment the impact of tags weights in the BM25 weighting function. Our 3 runs submitted to focused task are presented in table 2.

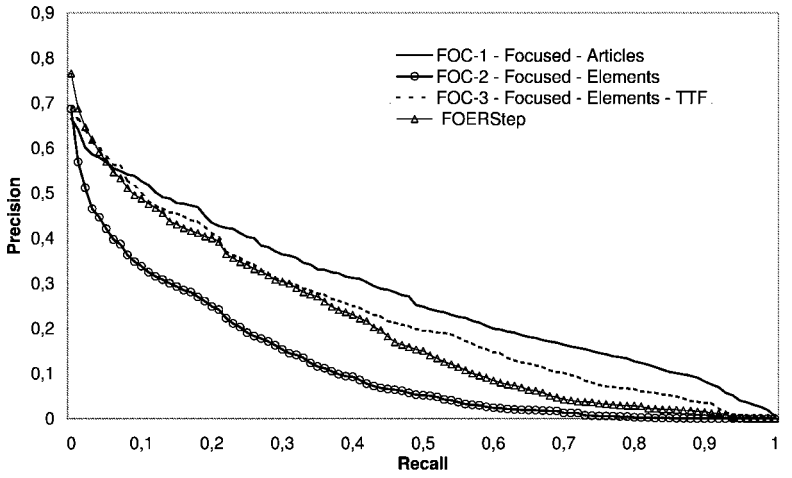
Table 2. Our 3 runs submitted to INEX 2008 Ad-Hoc, focused task

Run (name)	Type	Granularity	Tags weights
Foc-1 (JMU_expe_136)	Focused	articles	-
Foc-2 (JMU_expe_141)	Focused	elements	-
Foc-3 (JMU_expe_142)	Focused	elements	TTF 2006

4.3 INEX ranking: $iP[0.01]$

Our system gives very interesting results compared to the best INEX systems. Our runs are compared on the figure 1 against $FOERStep$, the best run submitted to INEX 2008 according to $iP[0.01]$ ranking, on 61 runs yet evaluated in the focused task. This run outperforms our runs at very low recall levels. Our run $Foc-1$ gives the best results at recall levels higher than 0.05. This is also shown by the $MAiP$ presented in table 3.

Fig. 1. Recall / Precision of 3 runs on 61 runs yet evaluated in the focused task



4.4 Articles versus elements

Our second aim was to compare classic retrieval of full articles versus focused retrieval of XML elements. We therefore indexed either the whole articles or the elements, and the parameters of the system were tuned also for focused retrieval.

Table 3. Evaluation of 61 runs yet evaluated in the focused task

Run	iP[0.01]	Rank	$MAiP$	Rank	R[1500]	S[1500]
FOERStep	**0.6873**	1	0.2071	27	0.4494	**78**
Foc-1	0.6412	13	**0.2791**	6	**0.7897**	390
Foc-2	0.5688	37	0.1206	45	0.2775	**51**
Foc-3	**0.6640**	7	0.2342	19	0.6110	234

It is interesting to notice that the BM25 model applied on full articles ($Foc-1$) outperforms our focused retrieval results ($Foc-2$) considering $MAiP$, despite the fact that BM25 parameter `ndl` is designed to take into account different documents lengths and thus documents granularities. Classic IR weighting functions, indexing and querying process, are undoubtedly not well adapted to focused retrieval. However, this is consistent with other results obtained during the INEX 2007 campaign where some top ranked systems only consider (and therefore return) full articles.

On the other hand, the focused run $Foc-2$ returns a smallest quantity of information. Indeed, the total size of the 1500 XML elements returned (for each query) is reduced to 51 Mb instead of 390 Mb for classic retrieval of full articles.

4.5 Pre-impacting of tags weights on terms weights

Finally, our third aim was to experiment the impact of tag weights in term weighting function in a focused retrieval scheme. In order to understand the pro and cons of our structured model, the weighting functions and the same parameters used for the baseline runs were also used with our structured model.

The figure 1 shows that our TTF strategy ($Foc-3$) improves dramatically the focused retrieval at low recall levels (from 0.5688 to 0.6640 following iP[0.01] ranking). However, it does not improve focused retrieval enough to reach better results than classic retrieval.

These results confirm also that, according to Robertson and *al.* [5], it is important to keep the non linearity of the BM25 weighting function by "pre-impacting" term position in the structure of document (in other terms, tags weights) on the terms frequencies (strategy TTF) instead of "post-impacting" it directly on the terms weights (strategy CLAW, cf. [2]).

Last but not least, the results with the measures R[1500] and S[1500] are synthesized in table 3. We can see that our runs obtain very good results according to R[1500] (*i.e.* the run exhaustivity): $Foc-1$ and $Foc-3$ outperform the best INEX run. But we don't have all the results yet, and we don't know the R[1500] score of other runs.

4.6 Relevant in context task

The "relevant in context" task asks systems to return non-overlapping elements or passages clustered per article to the user. Our 3 runs submitted to focused task are presented in table 4.

Table 4. Our 3 runs submitted to INEX 2008 Ad-Hoc, "relevant in context" task

Run (name)	Type	Granularity	Tags weights
RIC-1 (JMU_expe_148)	Relevant in Context	elements	-
RIC-2 (JMU_expe_150)	Relevant in Context	elements	TTF 2006
RIC-3 (JMU_expe_156)	Relevant in Context	elements	TTF 2007

Our 3 runs are focused runs. Thus we have experimented a classic focused retrieval ($RIC-1$), and a focused retrieval which integrates tag weights early in the term weighting function (TTF strategy). Both runs are based on BM25. Our run $RIC-2$ integrates tags weights estimated with INEX 2006 collection, and our third RIC run $RIC-3$ integrates tags weights estimated with INEX 2007 collection.

Our "relevant in context" runs have not been evaluated yet by INEX campaign.

4.7 Best in context task

The "best in context" asks systems to return articles with one best entry point to the user. Our 3 runs submitted to focused task are presented in table 5.

Table 5. Our 3 runs submitted to INEX 2008 Ad-Hoc, "best in context" task

Run (name)	Type	Granularity	Tags weights
BIC-1 (JMU_expe_149)	Best in Context	elements	-
BIC-2 (JMU_expe_151)	Best in Context	elements	TTF 2006
BIC-3 (JMU_expe_157)	Best in Context	elements	TTF 2007

Our 3 best in context runs are similar to our 3 relevant in context runs. All of them are focused runs. We have also experimented a classic focused retrieval ($BIC-1$), and a focused retrieval which integrates tag weights early in the term weighting function (TTF strategy). Both runs are based on BM25. $BIC-2$ integrates tags weights estimated with INEX 2006 collection, and $RIC-3$ integrates tags weights estimated with INEX 2007 collection.

Our "best in context" runs have not been evaluated yet by INEX campaign.

5 Conclusion

We proposed in [2] a new way of integrating the XML structure in the classic probabilistic model. We consider both the logical structure and the formatting structure. The logical structure is used at indexing step to define elements that correspond to part of documents. These elements will be indexed and potentially returned to the user. The formatting structure is integrated in the document model itself. During a learning step using the INEX 2006 collection, a weight is computed for each formatting tag, based on the probability that this tag distinguishes relevant terms. During the querying step, the relevance of an element is evaluated using the weights of the terms it contains, but each term weight is modified by the weights of the tags that mark the term.

The baselines are rather strong as the score of the BM25 run on article (run $FOC-1$) is ranked seven of the competition according to the $iAP[0.01]$ ranking.

Our experiments show the same results than Robertson and *al.* [5], *i.e.* "pre-impacting" term frequencies lead to better results than "post-impacting" BM25 weights. Actually, TTF changes significantly the performances of the methods when considering the $iP[0.01]$ or the $MAiP$ measure.

References

1. Ludovic Denoyer and Patrick Gallinari. The wikipedia XML corpus. In *SIGIR forum*, volume 40, pages 64–69, 2006.
2. Mathias Géry, Christine Largeron, and Franck Thollard. UJM at INEX 2007: document model integrating XML tags. In *Proc. of INitiative for the Evaluation of XML Retrieval (INEX), Dagstuhl*, 2008.
3. J. Kamps, J. Pehcevski, G. Kazai, M. Lalmas, and S. Robertson. INEX 2007 evaluation measures. In N. Fuhr, M. Lalmas, A. Trotman, and J. Kamps, editors, *Focused access to XML documents, 6th INEX Workshop*, 2007.
4. S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Sciences*, 27(3):129–146, 1976.
5. Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple BM25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, New York USA, 2004.

A Novel XML Fragment Retrieval Method based on Statistical Analyses

Kenji Hatano[1], Jun Miyazaki[2], and Atsushi Keyaki[1]

[1] Faculty of Culture and Information Science, Doshisha University
1–3 Tatara-Miyakodani, Kyotanabe, Kyoto 610–0394, Japan
[2] Graduate School of Information Science, Nara Institute of Science and Technology
Keihanna Science City, Ikoma, Nara 630-0192, Japan

Abstract. In this paper, we propose a novel XML fragment retrieval method based on deeply statistical analyses. In conventional XML search engines, the statistics like term-frequencies, document sizes, and so on are usually used directly for ranking retrieved XML fragments; however we believe that these statistics should be deeply analyzed to extract new knowledge of XML documents. We are now implementing two types of statistical analyses for the deeply developing high-precision XML search engines.

1 Introduction

XML (Extensible Markup Language) [1] has now become a universal data format of electronic documents. That is, almost all documents created by many applications are compiled as XML. This situation is very similar to the widespread HTML documents in the early 1990's; consequently, it is now a given that developing XML search engines has drawn attention an important research field for exploring existing electronic documents.

It is known that the XML search engines have various kinds of features that Web search engines do not have. They are able to handle XML fragments as a search result, to utilize the structure of XML documents, and to investigate the relationships among XML fragments extracted from XML documents, for example. It was true that the retrieval accuracies of the XML search engines could be improved using such features based on path-based term-weighting scheme [2], document-size-based data cleaning technique [3], and so on. However, it is also an undeniable fact that the retrieval accuracies themselves are still low for end-users, so that we still have so much research work to deal with. Therefore, we have to extract a lot more information from XML documents and users' queries for finding every feature of XML search engines which can be expected.

In this paper, we propose a method for utilizing extracted another information from XML documents to improve retrieval accuracies of XML search engines. Another information means that our approach does not directly utilize the statistics like term and path frequencies, document sizes, and so on, for improving retrieval accuracies of XML search engines, but discovers a new knowledge by investigating such statistics. We call this new knowledge *the secondary*

qualitative characteristics, because the knowledge is created from the statistics. With conventional term weighting approaches, it is difficult to find the *best entry point* which is the starting point of the most important document position associated with a given query, because the term weighting approaches capture only single information of each term in a document. However, for example, analyzing changes of term weight in a document which is one of the secondary qualitative characteristics could help us to detect the best entry point. Therefore, we believe that the secondary qualitative characteristics are helpful if the statistics contribute to improving retrieval accuracies.

2 Secondary Qualitative Characteristics

In this section, we explain the secondary qualitative characteristics in details.

In order to utilize the secondary qualitative characteristics, we firstly analyze the statistics extracted from XML documents. Currently, we usually use the number of terms and XPaths containing a term in an XML fragment as the statistics extracted from XML documents. The secondary qualitative characteristics are created from these statistics, so that data mining techniques are usually used to create such a new knowledge of XML documents.

Now, we consider that there are two types of the secondary qualitative characteristics because of the differences of our approaches. One approach is that extracting relationships between terms and XPaths of XML fragments, and then, constructing a conceptual connection between the terms and document structures in XML documents. This type of secondary qualitative characteristics is constructed by using conventional data mining techniques including clustering techniques, decision trees/rules, neural networks, and so on. That is, it is regarded as the knowledge of the target XML documents, which are the relationships between terms and document structures in XML documents.

Once we can extract the knowledge between terms and document structures, we are also able to construct another knowledge among terms and document structures in XML documents. These knowledges help to improve retrieval accuracy of XML search engines. However, this type of secondary qualitative characteristics is well-known in many research areas, so that it seems to be effective for improving retrieval accuracies of XML search engines, though it is not a novel one.

Therefore, we focus on the other one that uses deeply analyzed statistics extracted from XML documents. The deeply analyzed statistics do not directly utilize statistics extracted from XML documents like frequencies of terms and XPaths in XML documents and document-sizes. Currently, we propose the following two approaches using the deeply analyzed statistics.

2.1 Capturing Weight of Terms

We consider changes and variations of certain statistics. The data in the deeply analyzed statistics are captured as continuous values in a whole XML document

like sequence data, e.g., changes of TF-IPF score of a term in each text node, while those in conventional statistics like frequencies of terms and document-sizes are individual values in every XML fragment. If there is substantial change of the weight of a term in an XML document, it may indicate the existence of an important point of the XML document related with the term (see Fig. 1).

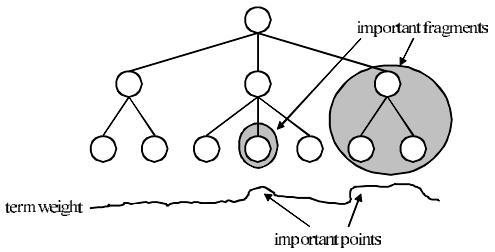


Fig. 1. Relationship between An XML document and A Term-weight

To determine the most important point of an XML document is to find the point that the continuous values increase most drastically. We can detect such a point by analyzing the difference between two adjacent values or its derived function. However, we have to take noise into consideration carefully. Even if we can detect the point that the difference between two adjacent values is the largest, it is not an important one when it is under a noise level. Therefore, the most important point of an XML document must be over a noise level. Once a noise level can be determined, we can find not only the best entry point, but also any relevant partial documents in XML documents. Therefore, this approach has an impact on general XML information retrieval. Unfortunately, since the determination of a noise level is not easy, the determination is currently one of the system tuning tasks.

We can regard an XML document as a string, so that the XML fragment including such important points is valuable about the term, in particular, for *Best-In-Context*. Of course, this secondary qualitative characteristics can be combined with one obtained from data mining. They can be utilized at the same time, so that further improvement of retrieval accuracies can be expected.

These ideas are based on not using statistics but utilizing their analyses. In other words, we have to focus on many aspects having XML documents, to extract them using ever more data mining techniques for attaining high-precision XML search engines. The secondary qualitative characteristics are recently required for implementing high-precision XML search engines, so that we have to apply them actively as important techniques in this research area.

2.2 Capturing Query Keywords

Our XML search engine firstly extracts all XML fragments whose root node is an intermediate node of XML documents, and then, calculates their scores based on both keywords and structures in user's query. This approach is different from conventional XML search engines' approaches, because our XML search engine has to calculate scores of all XML fragments extracted from XML documents. In other words, it takes a lot of time to calculate scores of the XML fragments; however, our XML search engine can retrieve XML fragments related to the user's query more accurately than conventional XML search engines.

It is said that our approach is one of the information extraction techniques. Employing our approach to developing an XML search engine, it is, thus, important to consider the following three issues:

query keyword connection: In conventional XML search engines, the XML fragments returned as a search result must contain all query keywords. In our XML search engines, however, it is not ensured it. Therefore, we have to consider how much the text nodes containing query keywords are connected to the root node of each XML fragment.

query keyword homogeneity: In information retrieval research area, the score of an XML fragment is large if it contains query keywords with large weights. In our XML search engine, the most important thing is to extract a valuable part of an XML fragment. That is to say, the valuable part should contain homogeneously-distributed query keywords. Therefore, we have to take into account the indicator reflected the homogeneity of query keywords in each XML fragment.

discarded information: In our previous research [3], we found that documents usually contain unuseful information. In other words, even if an XML fragment contains the query keywords with large weights, it may also contain large-sized but non-valuable information. Therefore, we have to make consideration of such discarded information in each XML fragment.

In this section, we propose the three types of scores to cope with the issues described above. These scores are calculated on each XML fragment, so that the number of calculated scores in an XML document is as same as that of its element nodes.

Edge Score The document structure of an XML document tends to express the structure of its context. In short, if the distance between the text nodes containing query keywords is long, these connection is weak. If so, these text nodes should not be contained in one XML fragment. Consequently, the XML fragment should not be retrieved as a search result. Therefore, we calculate the distance between the text nodes containing query keywords, and reflect on the score of the XML fragment. The distance between the text nodes containing query keywords can be calculated as the number of edges between the root node of the XML fragment and the text nodes. We call the distance *edge score* in this paper.

The larger the edge score is, the smaller the size of XML fragment is. This fact is satisfied with a requirement of the information extraction technologies described in [4]. Moreover, the larger the edge score is, the smaller the distance between the root node and the text node is; so that such XML fragment should be one of a search result due to its size. Considering above mentioned points, in this paper, the edge score is defined as follows:

$$s_e(n_i) = \frac{1}{1 + m(n_i)} \tag{1}$$

where n_i is the root node of an XML fragment, $m(n_i)$ is the number of edges from the root node to the text node with query keywords in the XML fragment. Let us calculate the edge scores using a sample XML document shown in Fig. 2. If a user issues a keyword query, k_1, k_2, and k_3 to the XML document, the element node n_3 has four edges to its descendant text nodes containing query keywords; so that $s_e(n_3) = \frac{1}{1+4} = 0.2500$. In the same way, we can calculate all edge scores, $s_e(n_2) = 0.1667$, $s_e(n_4) = s_e(n_6) = s_e(n_{11}) = s_e(n_{14}) = 0.5$, $s_e(n_9) = \frac{1}{1+7} = 0.125$, and $s_e(n_{10}) = s_e(n_{13}) = 0.333$.

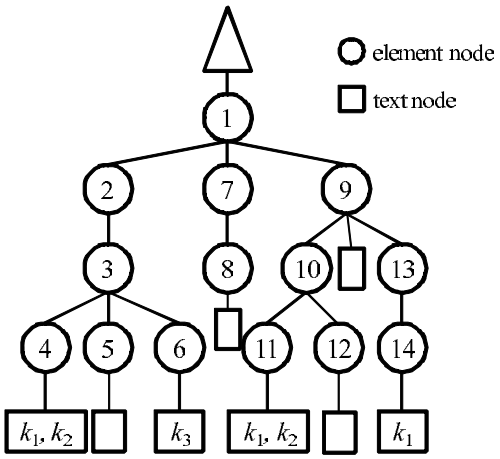


Fig. 2. A Sample XML document

Homogeneity Score As was mentioned at the beginning of this section, it is highly possible that an XML fragment is appropriate for returning as a search result if the XML fragment contains query keywords with large weights. However, we believe that the XML fragment should also contain all kinds of query keywords in itself, so that the XML fragment containing homogeneously-distributed

query keywords should be given a large score. Therefore, we have to take into account the indicator reflected the homogeneity of query keywords in each XML fragment.

In this paper, the indicator, *homogeneity score*, which is calculated by counting the number of query keywords and their varieties. Naturally, the more the XML fragment contains a wide variety of query keywords, the larger the homogeneity score of the XML fragment becomes. The homogeneity score is defined as follows:

$$s_h(n_i) = \frac{l(n_i)}{j} \tag{2}$$

where $l(n_i)$ is the number of text nodes containing query keywords in the XML fragment whose root node is n_i, and j is the variety of keywords in user's query. The homogeneity score is also satisfied with a requirement of the information extraction techniques described in [4]. Using the sample XML document shown in Fig. 2, we can calculate all homogeneity scores as follows: $s_e(n_2) = s_e(n_3) = \frac{3}{3} = 1$, $s_e(n_4) = s_e(n_9) = s_e(n_{10}) = s_e(n_{11}) = \frac{2}{3} = 0.667$, and $s_e(n_6) = s_e(n_{13}) = s_e(n_{14}) = \frac{1}{3} = 0.333$.

Contribution Score The edge and the homogeneity scores of an XML fragment are calculated by using the positions of the text nodes containing query keywords. However, they do not consider the text nodes not containing query keywords. In short, this fact is indicated that these scores of the XML fragment may be large if it is composed almost entire of discarded information. Such a XML fragment is not appropriate as a result XML fragment, so that the score which indicates containing valuable information in each text node should be needed.

This score, called *contribution score*, is defined while focusing on both the number of all text nodes and that of the text nodes containing query keywords in an XML fragment as follows:

$$s_c(n_i) = \frac{l(n_i)}{L(n_i)} \tag{3}$$

where $L(n_i)$ is the number of text nodes in the XML fragment whose root node is n_i, and $l(n_i)$ is the same as in equation (2). The contribution score is also satisfied with a requirement of the information extraction techniques described in [4]. As in the case of two scores above, we can calculate all contribution scores using a sample XML document shown in Fig. 2 as follows: $s_c(n_2) = s_e(n_3) = \frac{2}{3} = 0.667$, $s_e(n_4) = s_e(n_6) = s_e(n_{11}) = s_e(n_{13}) = s_e(n_{14}) = \frac{1}{1} = 1$, $s_e(n_9) = \frac{2}{4} = 0.5$, and $s_e(n_10) = \frac{1}{2} = 0.5$.

3 Expected Outcomes

Our main contribution of this paper is to develop a concept of the secondary quantitative characteristics to achieve high-precision XML search engines. Using both statistics and their deep analyses, XML search engines will be able to utilize

much more information to retrieve XML fragments related to users' information needs, and then their retrieval accuracies become better. Currently, we are now implementing our approaches using the secondary quantitative characteristics, so that we can evaluate them using the INEX 2008 test collection compared with our conventional approach.

Moreover, our statistical analyses can be used for developing not only high-precision but also high-performance XML search engines. Using the deeply statistical analyses, we can also classify XML fragments into categories. Therefore, XML search engines can narrow down the focus to some categories related to users' information needs. As a result, we can reduce query processing costs of XML search engines. If we have a time to evaluate the performance of our XML search engine, we try to evaluate it.

4 Related Work

The application of information retrieval techniques in searching XML fragments has become an important research area in recent years. Especially, the researchers in the INEX project have proposed many types of scoring algorithms using the statistics like the number of terms and document structures for XML search engines [5]. Over the years, it becomes clear that refining the level of granularity at which document structure is taken into account in pre-computing individual term weights either in the vector space model or the probabilistic model has increased retrieval accuracy.

For example, Fuhr et al. proposed a method for propagating scores of XML fragments leaf-to-root along the XML document tree [6]. However, although XIRQL, their proposed language, enables queries with a mix of conditions on both structure and keywords, only keywords are scored using conditions on document structure. Other scoring methods also use conditions on document structure to apply length normalization between query paths and data paths [7], to compute term weights based on element tags [2, 8]. It was reported that these methods were valuable for searching XML fragments [9]; however, such methods did not use the secondary qualitative characteristics described above.

At the same time, some researchers have been proposed to find minimal subtrees in XML documents containing all query keywords efficiency in database research area, which is called *SLCA* [10–13]. These approaches assume that the XML fragments whose root nodes are the LCAs should be answer XML fragments, so that their research purposes focused only on efficient finding the LCA nodes in XML documents. These researches did not consider the effectiveness of XML search engines; however, they are also important because the efficiency of XML search engines should be considered to search XML fragments. These studies are close to our method. However, ours can return the results which are not SLCA. In other words, our idea does not greatly depend on the SLCA concept, but makes use of it. Our approach which uses both tree-based scoring and the secondary qualitative characteristics would help us to identify which part of documents is the most important for given query keywords more accurately.

In this paper, therefore, we would like to utilize not only these statistics, but also the deeply analyzed one extracted from XML documents for searching XML fragments related to users' information needs for developing high-precision XML search engines. The statistics are valuable for effective searching of XML fragments, so that the secondary qualitative characteristics must be important for improving retrieval accuracies of XML search engines.

5 Conclusion

Conventional XML search engines directly utilize the statistics like the TF-IDF family to retrieve XML fragments effectively. It is natural to use the statistics in XML search engines; however such statistics should be analyzed more deeply to extract new knowledge of XML documents. Based on this consideration, we proposed methods described in Section 2 using the secondary qualitative characteristics for developing high-precision XML search engines.

In the near future, we evaluate our methods using the INEX 2008 test collection compared with our conventional approach.

References

1. Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., Yergeau, F.: Extensible Markup Language (XML) 1.0 (Fourth Edition). `http://www.w3.org/TR/xml` (September 2006) W3C Recommendation 16 August 2006, edited in place 29 September 2006.
2. Grabs, T., Schek, H.J.: 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., Springer (September 2003) 100–117
3. Hatano, K., Kinutani, H., Amagasa, T., Mori, Y., Yoshikawa, M., Uemura, S.: Analyzing the properties of xml fragments decomposed from the inex document collection. In: Advances in XML Information Retrieval. Volume 3493 of Lecture Notes on Computer Science., Springer-Verlag (July 2005) 168–182
4. Manning, C.D., Raghavan, P., Schutze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
5. Amer-Yahia, S., Lalmas, M.: XML Search: Languages, INEX and Scoring. SIGMOD Record **35**(4) (December 2006) 16–23
6. Fuhr, N., Großjohann, K.: XIRQL: An XML Query Language based on Information Retrieval Concepts. ACM Transactions on Information Systems **22**(2) (April 2004) 313–356
7. Carmel, D., Maarek, Y.S., Mandelbrod, M., Mass, Y., Soffer, A.: Searching XML Documents via XML Fragments. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (July/August 2003) 151–158
8. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: A Semantic Search Engine for XML. In: Proceedings of 29th International Conference on Very Large Data Bases. (September 2003) 45–56

9. Kazai, G., Lalmas, M.: INEX 2005 Evaluation Metrics. In: Advances in XML Information Retrieval and Evaluation. Volume 3977 of Lecture Notes on Computer Science., Springer-Verlag (June 2006) 16–29
10. Liu, Z., Chen, Y.: Identifying Meaningful Return Information for XML Keyword Search. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. (June 2007) 329–340
11. Li, Y., Yu, C., Jagadish, H.V.: Schema-Free XQuery. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases. (August/September 2004) 72–83
12. Xu, Y., Papakonstaninou, Y.: Efficient Keyword Search for Smallest LCAs in XML Databases. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. (June 2005) 527–538
13. Sun, C., Chan, C.Y., Goenka, A.K.: Multiway SLCA-based Keyword Search in XML Data. In: Proceedings of the 16th International Conference on World Wide Web. (2007) 1043–1052

University of Lyon3 and University of Avignon at INEX 2008: Ad Hoc Track

Fidelia Ibekwe-SanJuan[1] and Eric SanJuan[2]

[1] ELICO, University of Lyon3, France
ibekwe@univ-lyon3.fr
[2] LIA, Université d'Avignon, France
eric.sanjuan@univ-avignon.fr

Abstract. This paper describes the joint participation of University of Lyon 3 and LIA-University of Avignon in the INEX 2008 Ad-Hoc Retrieval track. In this first participation, we tested two ideas: (i) evaluate the performance of standard IR engines used in full document retrieval for XML element retrieval; (ii) evaluate the effectiveness of multiword terms as query terms. The second idea, more labor intensive, involved manual multiword term gathering from the Wikipedia corpus following an initial query. Different search strategies involving query expansion and term weighting were then designed and submitted to the Indri query language.

1 Introduction

INEX XML retrieval aims to evaluate systems performance in retrieving relevant document components (e.g. XML elements) rather than whole documents. It is therefore a natural assumption that participating systems would make use of the XML structure in their query and that they would locate xml elements or passages rather than full articles. This assumption is however not a mandatory requirement and systems retrieving full articles are welcomed as a means of comparing full article vs element retrieval. The corpus used in the 2008 edition is the same as in previous years, it is the 2006 version of the English Wikipedia comprising 659,388 articles without images [1]. From this corpus, participants were asked to submit topics corresponding to real life information need. A total of 133 such topics were built. A topic consists of four fields: Content Only field (<CO> or <Title>) with a multiword term expression of the topic; a Content Only + Structure version of the topic (<CAS>) which is the title with indication of xml structure where the relevant elements may be found; a description field which is a slightly longer version of the title field; and a <narrative> field comprising a summary with more details about the expected anwsers. Typically, the narrative would indicate things to eliminate from relevant answers and draw boundaries on relevant articles that can be geographic, spatial, genre or historical in nature. Some title fields contained booleen operators that required systems to explicitly exclude (-) or include (+) certain terms in the relevant answer elements.

The Ad Hoc track is divided into 3 subtasks: Focused retrieval, Relevant-in-Context (RiC), Best-in-Context (BiC). The focused task requires systems to return a ranked list

of relevant non-overlapping elements or passages. This is called the "fetching phase". The Relevant-in-Context task builds on the results of the focused task. Systems are asked to select, within relevant articles, several non-overlapping elements or passages that are specifically relevant to the topic. This is called the "browsing phase". The Best-in-Context task is aimed at identifying the best entry point (BEP) to read a relevant article. This tests the capacity of a system to select one and only one best starting point for reading a relevant article.

As this is our first participation in INEX, we tested a strategy based on two ideas: (i) evaluate the performance of state-of-art IR engines used in full document retrieval; (ii) evaluate the effectiveness of using multiword terms as query terms. The first strategy, which can be perceived as the "principle of least effort" tested several features of the Indri engine to implement different search strategies. The second strategy, more labor intensive, consisted in manually term gathering from Wikipedia corpus from an initial query. The two strategies were combined. Two types of results were computed by the track organizers: (i) xml element or passage retrieval and (ii) full article retrieval. Our runs obtained nice performances in the xml retrieval and very good performances in full article retrieval. Indeed, one of our runs took the first position in two tasks (Focused task, Relevant-in-Context task) and 2nd position in Best-in-Context task for full document retrieval. This tends to support the relevance of retrieving full articles in Ad-hoc XML retrieval.

2 Corpus preparation

No pre-processing was performed on the corpus. In particular, no lemmatization was performed. An inverted Indri index was constructed on the xml file of the corpus. A nice feature of the Indri index is that word occurrences and positions in the original texts are also recorded. This is a significant difference with other IR engines such as Lucene. The corpus was then loaded onto the TermWatch system, a text analysis platform which we designed [2]. In this experiment, only TermWatch's interface was used to in order to construct Indri queries and view returned articles as html files.

3 Query formulation

We performed a baseline run using the original sequence of text in the title field without stop word removal and without attempting to extract query words or terms. The other runs were based on a multiword query term construction process which can be described by the following steps:

1. we first searched the wikipedia corpus using terms from the three fields title, description and narrative of each topic. We did not use the CAS field and we ignored the "+, -" operators in the other fields. The multiword terms collected from these three fields were formulated as Indri queries. The terms were combined by one or two Indri belief operators (band, combine) or proximity operators (odN). For more details on the Indri query language, see [3].

[3] http://www.lemurproject.org/lemur/IndriQueryLanguage.php

2. from the retrieved articles, we manually explored the top 20 for more terms with which to expand the initial set of terms in (1). This led to acquiring synonyms, abbreviations, hypernyms, hyponyms and associated terms. At this stage, we can have anything from 2-20 multiword terms per topic, expressed as subsets of query terms linked by one or two Indri operators.

3. this second set of expanded query terms was combined with the #or operator.

4. different search strategies were then tested on the expanded set of terms using other features of Indri search engine such as query expansion (QE) and weighting. The precise parameters for each run will be detailed hereafter.

The multiword term gathering phase was the most labor-intensive, requiring roughly one to two hours per topic.

4 Search strategies

We submitted only full article retrieval for the three subtasks (focused, RiC, BiC). On the whole, we devised five different search strategies for the three tasks (15 runs altogether). We first tested a baseline strategy using basic Indri without multiword query terms, then progressively built more complex search strategies by adding IR mechanisms such as query expansion and term weighting to multiword query terms linked by Indri belief operators.

Table 1. Table 1. Ad-hoc runs for the three tasks: Focused, RiC, BiC.

RunID	Approach
ID92_manual_indri01	multiword term with Indri #or operator
ID92_auto_indri02	automatic one word query with Indri #combine operator
ID92_manualQE_Indri03	multiword term with Indri query expansion (QE)
ID92_manual_weighting_Indri04	multiword term with Indri term weighting (TW)
ID92_manual_weightingQE_Indri05	multiword term with Indri TW and QE

4.1 Baseline bag-of-word search

Run ID: ID92 _auto _indri02
This is an automatic run using only the query input from the title field of the topic, without stopword removal. The constituent words were linked by the "#combine" operator and submitted to Indri search engine. The idea is to test the performance of Indri basic engine without additional enhancements.

4.2 Multiword term search

Run ID: ID92 _manual _indri01
In this run, the multiword terms gathered during the process described in section 3 are

linked by the "or" operator. No other features were used. This run is a counterpart to the baseline run. We test a basic automatic bag-of-word search in section 4.1 against manually acquired multiword query terms.

4.3 Multiword term search with Query expansion (QE)

Run ID: ID92 _manualQE _Indri03
This run is based on manual_indri01 run with Indri query expansion mechanism using the following parameters. The number N of added terms is 50. These terms were all extracted from the $D = 4$ top ranked documents using the original query. Finally, in the expanded query, the original query was weigthed to $w = 10\%$. These three parameters $(D = 4, N = 50, w = 10)$ were learned based on TREC Enterprise 2007 track results on the CSIRO website corpus. Thus from a totally different corpus than the one used in INEX.

4.4 Weighted multiword term search

Run ID: ID92 _manual _weighting _Indri04
This run takes the multiword query terms in manual_indri01 and converts them into a weighted bag of words. Each word w occurring in at least one query term is used. Its weight is set to $c + 0.1 \times m$ where c is the number of query terms with w as head word (noun focus) and m the number of terms where it appears as a modifier word. We then used the Indri operator "weight" to combine these words and their weights.

4.5 Weighted multiword term search with Query expansion

Run ID: ID92 _manual _weightingQE _Indri05
This run combines the two preceding strategies, i.e multiword query terms with term weighting and QE.

5 Results

INEX Ad-Hoc evaluations are carried out at different levels of precision and recall. For the focused task, interpolated precision (iP) is calculated at 0.00, 0.01, 0.05 and 0.10 recall levels with iP[0.01] being the official measure. For the Relevant-in-Context and Best-in-Context task, the measure is mean average generalised precision (MAgP) at early ranks (5, 10, 25, 50). The same search strategies were submitted for all three tasks. We present the results obtained by our five runs for the three tasks ranked by xml retrieval, then by full document retrieval.

5.1 Rank by XML element retrieval

Although we submitted only full articles, our runs were ranked by xml retrieval. We present scores obtained for the three subtasks.

Table 2. Results for the Focused task. (Total runs submitted: 61)

RunID	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP	Rank at iP[0.01]
ID92_manualQE_Indri03	0.66	0.66	0.61	0.55	0.30	6th
ID92_manual_indri01	0.65	0.64	0.58	0.51	0.24	12th
ID92_manual_weightingQE_Indri05	0.62	0.62	0.59	0.55	0.28	23rd
ID92_manual_weighting_Indri04	0.59	0.58	0.56	0.54	0.25	32nd
ID92_auto_indri02	0.56	0.56	0.52	0.45	0.24	38th

Focused search A total of 61 runs were submitted by 17 institutions.

Four of our runs are ranked in the first half of all submitted runs with the "ID92 _manualQE _Indri03 run" being ranked at the 6th position. Not surprisingly, the automatic baseline (ID92 _auto _indri02) running a bag-of-word query approach showed the worst performance.

Figure 1 shows all levels of precision of our runs. It appears that at iP[0.15] all srategies but the baseline outperformed simply manual run (ID92_manual_indri01). At iP[0.50] only strategies with query expansion (QE) outperform the Indri baseline (ID92_auto_indri02).

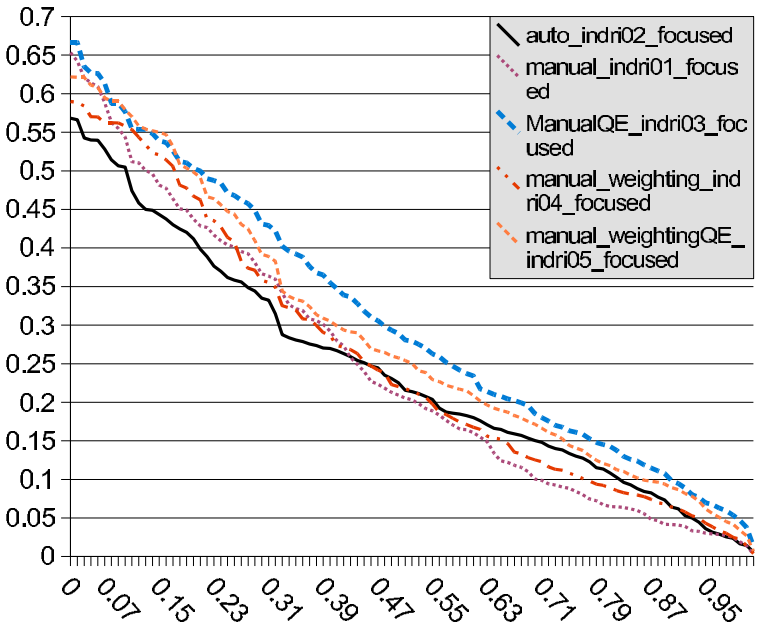


Fig. 1. Precision of Lyon 3 & University of Avignon runs on focused task

Relevant-in-Context A total of 40 runs were submitted for this task by all participants. The table below shows the scores obtained by our five runs at different precision levels, their MagP and overall ranks.

Table 3. Results for Relevant-in-Context task. (Total runs submitted: 40)

RunID	gP[1]	gP[2]	gP[3]	gP[5]	MAgP	Rank
ID92_manualQE_Indri03	0.55	0.50	0.46	0.41	0.20	4th
ID92_manual_weightingQE_Indri05	0.52	0.47	0.44	0.37	0.18	12th
ID92_auto_indri02	0.44	0.40	0.37	0.32	0.17	19th
ID92_manual_indri01	0.54	0.48	0.45	0.39	0.15	24th
ID92_manual_weighting_Indri04	0.45	0.45	0.40	0.35	0.14	32nd

Here again, the "ID92_manualQE_Indri03 run" outperformed the others although the order of performance is somewhat different. Surprisingly, the baseline approach (ID92_auto_indri02) outperformed both the manual multiword term approach (ID92 _manual _indri01) and the same approach with weighting (ID92 _manual _weighting _Indri04) whereas these two runs had higher precisions at early recall levels.

Best-in-Context Our runs basically conserve the order or performance as in RiC task with all runs moving forward to higher ranks. Particularly noticeable is the good performance of the ID92_manualQE_Indri03 which is ranked 2nd out of 35 submitted runs.

Table 4. Results for Relevant-in-Context task. (Total runs submitted: 35)

RunID	gP[1]	gP[2]	gP[3]	gP[5]	MAgP	Rank
ID92_manualQE_Indri03	0.56	0.49	0.45	0.40	0.21	2nd
ID92_manual_weightingQE_Indri05	0.47	0.44	0.42	0.38	0.19	6th
ID92_auto_indri02	0.39	0.37	0.36	0.32	0.17	10th
ID92_manual_indri01	0.55	0.47	0.45	0.40	0.16	14th
ID92_manual_weighting_Indri04	0.40	0.43	0.37	0.34	0.15	17th

5.2 Ranking by full article retrieval

Here, our runs obtained even better rankings than in the xml element ranking. Our "ID92_manualQE_Indri03 run" ranked 1st for both the Focused and Relevant-in-Context tasks and 2nd for the Best-in-Context task. Table 5 below give more details of these results. For each task, we indicate the run's MAP and rank out of all submitted runs.

As we can see, the performance order of our runs do not change in the three tasks and the scores remain the same because we submitted the same search strategies irrespective of subtasks.

Table 5. Rankings for full document retrieval. Focused(**Foc**), Relevant-in-Context (**RiC**), Best-in-Context task (**BiC**) tasks. Total runs submitted: 76, 49 and 38 resp.

RunID	MAP	Foc	RiC	BiC
ID92_manualQE_Indri03	0.36	1st	1st	3rd
ID92_auto_indri02	0.32	12th	8th	10th
ID92_manual_weightingQE_Indri05	0.32	17th	9th	11th
ID92_manual_indri01	0.26	48th	32nd	23rd
ID92_manual_weighting_Indri04	0.25	56th	39th	30th

6 Discussion

Based on a survey of the results obtained in INEX 2007 ad-hoc task, it appeared that retrieving full articles instead of specific xml elements or passages was not a ridiculous strategy. Indeed, in the 2007 edition, systems that submitted runs with only full article retieval featured among the top 10 best runs in each task: 8th in focused retrieval, 3rd in RiC and 1st in RiC ([3]). This can be explained by several factors most of which have been analysed in previous INEX conferences [4]:

1. assessment of INEX topics are performed by participants themselves. For INEX 2006 & 2007, these assessments showed that for the BiC task, the best entry point (BEP) to begin reading a relevant article is situated not far from the beginning of the article. Thus a system retrieving full articles is not seriously handicapped for this task.
2. depending on the length of a given wikipedia article, the relevant portion can correspond to the full article rather than to sub-elements as these may be judged less self-contained by assessors. This point will require further investigation. However, from our summarily observations, the length of most wikipedia articles is at least a full A4 paper.
3. the scoring metrics appear not to be unfavorable to full article retrieval especially for the BiC task, hence the good performance of systems retrieving full articles.

References

1. L. Denoyer, P.G.: The wikipedia xml corpus. In: SIGIR Forum. (2006) 6
2. SanJuan, E., Ibekwe-SanJuan, F.: Text mining without document context. Information Processing and Management **42** (2006) 1532–1552
3. Thom, J., Pehcevski, J.: How well does best in context reflect ad hoc xml retrieval? In: PreProceedings of the 14th Text Retrieval Conference (INEX 2007), Dagstuhl, Germany (17-19th December 2007) 124–125
4. Fuhr, N., Kamps, J., Lalmas, M., Malik, S., Trotman, A.: Overview of the inex 2007 ad hoc track. In: PreProceedings of the 14th Text Retrieval Conference (INEX 2007), Dagstuhl, Germany (17-19th December 2007) 1–22

University of Waterloo at INEX2008: Adhoc, Book, 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. In this paper, we describe University of Waterloo's approaches to the Adhoc, Book, and Link-the-Wiki tracks. For the Adhoc track, we submitted runs for all the tasks, the Focused, the Relevant-in-Context, and the Best-in-Context tasks. The preliminary results show that we ranked first among all participants for each task, by the simple scoring of elements using Okapi BM25. In the Book track, we participated in the Book retrieval and the Page-in-Context tasks, by using the approaches we used in the Adhoc track. The Book track has yet to produce preliminary results. In the Link-the-Wiki track, we submitted runs for both File-to-File and Anchor-to-BEP tasks, using PageRank [1] algorithms on top of our previous year's algorithms that yielded high performance. The preliminary results indicate that our baseline approaches work best, although other approaches have rooms for improvements.

1 Introduction

In 2008, University of Waterloo participated in the Adhoc, the Book, and the Link-the-Wiki tracks. In the Adhoc track, we implemented both passage and element retrieval algorithms to compare the relation between the best passages and the best elements. This is in contrast to our 2007 runs [3] that compared the passage-*based* element retrieval algorithm against the simple element retrieval algorithm. We scored elements and passages using a *biased* BM25 and language modeling [5], in addition to Okapi BM25 [6] to see the effect of scoring functions in retrieval results.

In the Book track, we implemented our tried and true element retrieval algorithm with Okapi BM25 to retrieve best books and pages.

In the Link-the-Wiki track, we added PageRank algorithm [1] in addition to using anchor density [3, 7] for the numbers of links to return for each topic.

This paper is organized as follows. In Section 2, we describe our approaches to the Adhoc track, and in Section 3, we describe our approaches to the Book track. In Section 4, we describe our approaches in the Link-the-Wiki track. We conclude this paper with directions for future work in Section 5.

2 Ad hoc Track

In the Adhoc track, as in the past year, the basic retrieval algorithms used are element retrieval and passage retrieval.

The only difference between element retrieval and passage retrieval is the retrieval unit. In element retrieval, we only scored the following elements in corpus,

<p>, <section>, <normallist>, <article>, <body>, <td>, <numberlist>, <tr>, <table>, <definitionlist>, <th> ,<blockquote>, , <li>, <u>.

In passage retrieval, we scored passages of any word-lengths.

To score an element or a passage, we converted each topic into a disjunction of query terms without negative query terms. We located positions of all query terms and XML tags using Wumpus [2]. We then used two versions of Okapi BM25 [6] to score passages and elements. The score of an element/passage P using Okapi BM25 is defined as follows.

$$s(P) \equiv \sum_{t \in Q} W_t \frac{f_{P,t}(k+1)}{f_{P,t} + k(1 - b + b\frac{pl_P}{avgdl})} \quad , \tag{1}$$

where Q is a set of query terms, W_t is an IDF value of the term t in the collection, $f_{P,t}$ is the sum of term frequencies in a passage P, pl_P is a passage length of P, and $avgdl$ is an average document length in Wikipedia collection to act as a length normalization factor.

Using a biased Okapi BM25, the first α words' term frequency is multiplied by β. The reason behind this is that because in Best-in-Context task it appears that the closer the best entry point is to the beginning of an article, the more relevant it is judged, we thought that the closer the terms are to the beginning of an element the more relevant they are.

We tuned parameters using INEX2007 Adhoc track evaluation scripts distributed via email by the organizers. Our tuning approach was such that the sum of all relevance scores, ip[0.00], ip[0.01], ip[0.05], ip[0.10], and MAiP are maximized. However, by looking at the training results, the choice of the parameters did not seem much different if we had chosen the official metrics, ip[0.01] for the Focused task, and MAiP for Relevant-in-Context and Best-in-Context task.

2.1 Focused Task

In the focused task, after scoring all elements and passages, we eliminated overlaps and returned the top 1500 elements and passages. There are four runs we submitted. For element retrieval, we submitted three runs using Okapi BM25 with parameters $k = 4$ and $b = 0.8$, the biased Okapi BM25 with $\alpha = 10$, $\beta = 2$, $k = 3$, and $b = 0.8$. For passage retrieval, we submitted a run using Okapi BM25 with $k = 4$ and $b = 1.2$.

Our biased Okapi BM25 approach ranked first amongst 19 participants and the 61 runs. Table 1 shows the results of our individual runs.

Although our biased Okapi BM25 approach performed better than the simple BM25, it did not give substantial improvement. However, it may give an

insight into efficient scoring by possibly only scoring the first words of an element/passage and ignoring the rest. This would not only reduce the length of elements/passages to score, but also the number of elements/passages to score because many of them overlap at the beginning. A look into the scores of all elements/passages that start at the same character is necessary.

The passage run was disqualified because of overlaps. While training with correct overlap elimination, the evaluation scores of the passage run was somewhat lower than that of BM25 run. Re-evaluation of the correct run is under way. However, the overall impression is that Okapi BM25 is the best scoring function and that users prefer element results to passage results. One question left is, would users prefer ranges of elements over single elements or passages?

Run	Rank	ip[0.01]
Biased Okapi	1	0.68731056
Okapi BM25	2	0.68654649
Passage Retrieval	-	-

Table 1. Results of Waterloo's Runs in the Adhoc Track Focused Task

2.2 Relevant-in-Context Task

In the Relevant-in-Context task, the results of the top 1500 elements/passages using Okapi BM25 in the Focused task was grouped in two different ways. The first way is to rank the articles according to the score of the articles themselves with parameters $k = 2$ and $b = 0.8$, the second way is to rank the articles according to the scores of the highest scoring elements/passages the article contains with $k = 2$ and $b = 0.8$.

Our run with best element score ranked first amongst 11 participants and their 41 runs. Table 2 shows the results of our individual runs.

Because there was no substantial difference between ranking articles by the article scores and ranking articles by their best element scores, it may be as effective to fetch the articles with the highest articles score first, and then run element retrieval on the retrieved set.

Run	Rank	ip[0.01]
Best Element Score	1	0.22631027
Article Score	2	0.22523137

Table 2. Results of Waterloo's Runs in the Adhoc Track Relevant-in-Context Task

2.3 Best-in-Context Task

In the Best-in-Context task, we recorded the scores of the best element/passage in each article for biased and simple Okapi BM25, and returned the top 1500 elements/passages. The parameters used for element retrieval with Okapi BM25 are $k = 1.2$ and $b = 0.4$, for element retrieval with biased Okapi BM25 are $\alpha = 10$, $\beta = 2$, $k = 0.6$, and $b = 0.4$, and for passage retrieval using Okapi BM25 are $k = 1.4$ and $b = 0.8$.

Our simple Okapi BM25 based run scored the first amongst 13 participants and their 35 runs. Table 3 shows the results of our individual runs.

The performance of all the approaches are similar during the training phase and the results are fully what we had expected. As in the Focused task, the biased Okapi function did similarly well to a simple Okapi BM25, implying a possible efficiency improvement. The results of the passage run is not impressive as expected. This fact is quite alarming given that the only difference between the two approaches is the unit of scoring; the highest scoring passage must be quite far apart from the highest scoring element. This explains why our passage-*based* element retrieval run of INEX 2007 was not as effective as the simple element retrieval run.

Run	Rank	ip[0.01]
Okapi BM25	1	0.22065149
Biased Okapi BM25	3	0.20970581
Passage Retrieval	24	0.12374541

Table 3. Results of Waterloo's Runs in the Adhoc Track Best-in-Context Task

3 The Book Track

In the Book track, we employed the element retrieval algorithm with Okapi BM25 as described in Section 2. The only difference is the unit of scoring, which are document, page, region, and section. Since we had no training data available from previous years, we ran our algorithms with arbitrary parameters.

In the Book search task, the first run was obtained by ranking books according to the document scores, and the second run was obtained by ranking books according to their best element scores.

All runs for the Page-in-Context task differed in how the books were ranked. The first run ordered books by their best element scores, the second run ordered books by the books' score with manual query expansion. The manual query expansion was done by observing the query phrases and adding any extra query phrases that may be helpful to disambiguate the queries. For example, if the query phrase is just "mouse", but the description of the user's information need suggests that it pertains to the animal mouse, as opposed to the computer mouse,

the expanded query phrase would be "animal mouse". Query expansion normally increases precision, but it also increases the processing time. Since one of the goal of book search is to create an efficient search system, our assumption is that applying query expansion only on the whole book score maybe reasonable. Other effort to shorten search processing time was to make a distributed search. The problem with this approach was that in order to create a merged list of top scoring elements precisely, for each distributed node, it was necessary to compute and store all the element scores, which was costly. To see how much of top elements for each node could be cut-off without affecting the results, we made the above two runs with various cut-off values, including no cut-off.

4 Link the Wiki Track

Following the previous year's successful run, we decided to extend our basic approaches for both incoming and outgoing links by incorporating PageRank [1]. Our File-to-File runs mirror our Anchor-to-BEP runs by employing the same algorithms, but abstracting out to the article level. Therefore, in this section, we describe our approaches to Anchor-to-BEP runs.

Outgoing Links As in the last year, the basic ingredient to computing outgoing links is the following ratio, γ.

$$\gamma = \frac{\sharp \text{ of files that has a link from anchor } a \text{ to a file } d}{\sharp \text{ of files in which } a \text{ appears at least once}}$$

Because this year, we are allowed to specify multiple destinations for a given anchor phrase, for each anchor phrase that appear in the corpus, we computed γ for the most frequent destination, but kept up to four other destinations on the list. We sorted all anchor phrases by γ, and then for each topic file, we looked for the locations of the anchor phrase. Once the anchor phrases are located, we listed the destinations in the order of frequency in the corpus. We specified the best entry point as the beginning of the article.

For the second run, we computed the maximum number of anchor phrases that a topic file can contain using the size of the topic file. As in [3], we define the *anchor density* δ as

$$\delta = \frac{\#\text{ number of anchor strings in the file}}{\text{size of the file in bytes}}.$$

We computed that anchor density is linear and that there are 3.584 anchor per KB of a document in the corpus and set the number of anchor phrases in the topic files accordingly.

For the second run, instead of ordering the anchor phrases by the γ values, we ordered them by the PageRank value. For this run, there is no cut-off values as in the second run.

Preliminary results that compared the runs against the links in Wikipedia corpus indicate that both for File-to-File and Anchor-to-BEP runs, our first runs without an anchor density performed best among our submissions followed by our second runs with anchor density. The runs based on PageRank performed poorest among our submissions. Compared to other participants, both our runs at the Flie-to-File and the Anchor-to-BEP tasks ranked third.

The fact that our anchor density approach did not perform as well as our baseline approach is buffling. Maybe we needed to compute anchor density per word, instead of per file size. But more importantly, the anchor density may vary not only by size, but also by the type of articles.

Incoming Links

The first run of the incoming links is done exactly the same as in Waterloo's last year's run. For each topic file, we created a query that consists of the topic title, and looked for the files that contains the title. We did not differentiate between the articles that contain the query term, but we simply picked the first 250 articles in the corpus. The best entry point to the topic file was set to the beginning of the article.

For the second run, we used the same set of query terms, but applied the element retrieval algorithm in the Adhoc track to rank the article that contains the query terms according to its highest element score.

For the third run, we took the result of our third outgoing run to compute a topic oriented PageRank [4] and reranked all articles containing the query term by these values.

The preliminary results that compared runs against the links in Wikipedia corpus indicate that for File-to-File run, our PageRank based approach performed best, closely followed by our baseline approach, and for Anchor-to-BEP run, our baseline run performed slightly better than our PageRank run. In either case, our element retrieval based runs did not perform as well as our other runs. Compared to other participants, our best run in the File-to-File task ranked second among participants and our best run in the Anchor-to-BEP task ranked fifth among participants.

It is interesting to see that our baseline approach did equally well as our PageRank approach and did much better than our element retrieval approach. Our poor performance of element retrieval based approach is probably due to lack of training. The fact that PageRank was reasonably effective in finding incoming links, but not in finding outgoing links requires further investigation.

5 Conclusions and Future Work

This year, we extended our previous year's best performing algorithms to improve the performance. Unfortunately, our simple algorithms from the previous years not only did the best amongst all our runs, but also did best amongst all the runs in all the tasks in the Adhoc track. This may indicate the uselessness of the XML structure. On the other hand, since it seems that the passage

runs do not perform as well as our element runs, marking up elements do seem useful. Moreover, the effect of specifying ranges of elements as opposed to the current approaches of choosing the single elements or passages is a new area to investigate.

A very interesting area of future research is the effect of positioning within a document to the relevance. Maybe the poor performance of passage retrieval in the Best-in-Context task is because the highest scoring passage within a document is located further down in the document than the second highest scoring passage that starts close to the beginning of the document. Therefore combining the score of a passage with the positional/structured information seems promising.

References

1. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems, Proceedings of the Seventh International World Wide Web*, 30(1-7):107–117, 1998.
2. S. Büttcher. the Wumpus Search Engine. Accessible at `http://www.wumpus-search.org`, 2007.
3. K. Y. Itakura and C. L. A. Clarke. University of waterloo at INEX2007: Adhoc and link-the-wiki tracks. *Focused Access to XML Documents, LNCS4862*, pages 417–425, 2008.
4. 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.
5. J. M. Ponte and W. B. Croft. 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*, pages 275–281, New York, NY, USA, 1998. ACM Press.
6. S. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC-7: Automatic ad hoc, filtering, vlc and interactive track. *7th Text REtrieval Conference*, 1998.
7. J. Zhang and J. Kamps. Link detection in XML documents: What about repeated links. In *Proceedings of the SIGIR 2008 Workshop on Focused Retrieval*, pages 59–66, 2009.

Enhancing Keyword Search with a Keyphrase Index

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

[1] Department of Computer Science
P. O. Box 68 (Gustaf Hällströmin katu 2b)
FI–00014 University of Helsinki
Finland
{Miro.Lehtonen,Antoine.Doucet} @cs.helsinki.fi
[2] GREYC CNRS UMR 6072,
University of Caen Lower Normandy
F-14032 Caen Cedex
France
Antoine.Doucet @info.unicaen.fr

Abstract. Combining evidence of relevance coming from two sources — a keyword index and a keyphrase index — has been a fundamental part of our INEX-related experiments on XML Retrieval over the past years. In 2008, we focused on improving the quality of the keyphrase index and finding better ways to use it together with the keyword index even when processing non-phrase queries. We also updated our implementation of the word index which now uses a state-of-the-art scoring function for estimating the relevance of XML elements. Compared to the results from previous years, the improvements turned out to be successful in the INEX 2008 ad hoc track evaluation of the focused retrieval task.

1 Introduction

The interest in developing methods for keyphrase search has decreased recently in the INEX community partly because most of the queries are not keyphrase queries [1]. However, we believe that indexing interesting phrases found in the XML documents can be useful even when processing non-phrase queries. As the XML version of the Wikipedia is full of marked-up phrases, we have been motivated to work on the quality of the phrase index, as well, in order to capture those word sequences that document authors really intended to be phrases.

In the previous years, our ad hoc track results have not been at the same level with the best ad hoc track results. We believed that the reason lay in the keyword index and the tfidf scoring function because the top results were achieved with the probabilistic retrieval model. Lesson learned: we introduced BM25 as the new scoring function for the keyword index. The latest results of the INEX 2008 evaluation show great improvement from previous years. How much the improvement is due to the state-of-the-art scoring function and how much to the improved phrase index is still unclear, though.

This article is organised as follows. Section 2 describes our IR system as it was implemented in 2008. In Section 3, we show how the keyphrases are extracted from the document collection into a keyphrase index. Section 4 details the scoring methods for both the word index and the keyphrase index. Our results are presented in Section 5, and finally, we draw conclusions and directions for future work in Section 6.

2 System description

Documents, selection of indexed XML elements, term index, phrase index, and system architecture will be described here.

3 The anatomy of a keyphrase index

Building a keyphrase index starts from finding or detecting the word sequences that should be considered keyphrases. As we are indexing hypertextual XML documents, it is natural to use the characteristics of hypertext documents and the markup language in the analysis as we detect passages that are potentially indexed keyphrases. The analysis is followed by a text mining method for extracting Maximal Frequent Sequences.

3.1 Phrase detection and replication

Most of the XML markup in the Wikipedia articles describes either the presentation of the content or the hyperlink structure of the corpus, both of which show as mixed content with inline level XML elements. In these cases, the start and end tags of the inline level elements denote the start and the end of a word sequence that we call an *inline phrase*. These phrases include the anchor texts of hyperlinks as well as phrases with added emphasis, e.g., italicized passages. An exact definition for the XML structures that qualify was presented at the INEX 2007 workshop [2]. Intuitively, the inline phrases are highly similar to the multiword sequences that text mining algorithms extract from plain text documents. Therefore, the tags of the inline elements are strong markers of potential phrase boundaries. Because phrase extraction algorithms operate on word sequences without XML, we incorporate the explicit phrase marking tags into the word sequence by replicating the qualified inline phrases.

Considering the effect of replication, we only look at the character data as the tags and other XML markup are removed before phrase extraction. The most obvious effect is the increase in phrase frequency of the replicated inline phrases with a similar side effect on the individual words they compose of. Moreover, the distance between the words preceding and following the phrase increases, which makes the phrase boundaries more explicit to those phrase extraction algorithms that allow gaps in the multiword sequences.

Duplicating the inline phrases lead to a 10–15% improvement in the MAiP on the INEX 2007 topics [3], but more recent experiments where the phrases were

replicated three times have shown even further improvement when tested on the same topics. Note that these results depend on the phrase extraction algorithm and that other algorithms than ours may lead to different figures. Anyway, we chose to see if the triplication of the inline phrases works on the INEX 2008 topics, as well, and built the phrase index correspondingly.

3.2 MFS extraction

A frequent sequence is defined as a sequence of words that must occur in the same order more often than a given sentence-frequency threshold. MFSs are constructed by expanding a frequent sequence to the point where the frequency drops below the threshold. This way we obtain a compact phrasal description of a document collection [4].

More description to be added here.

4 Scoring XML fragments

The scoring function used with the word index was BM25 as implemented in the Lemur Toolkit [5]. Computation of the Retrieval Status Value for the MFS's will be described here.

5 Results

We submitted three runs for the ad hoc track task of focused retrieval. The configurations of the submitted runs were based on experiments on the ad hoc track topics of INEX 2007, according to which the best proportion of weight given to terms and phrases would be around 92:8–94:6. The weight is given to the word index component is part of the Run ID. The initial results are shown in Table 1.

Run ID	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
UHel-Run1-92	0.6920	0.6503	0.5540	0.4983	0.2239
UHel-Run2-93	0.7030	0.6617	0.5556	0.5013	0.2255
UHel-Run3-94	0.7109	0.6619	0.5532	0.5028	0.2251

Table 1. Evaluation of our three official runs submitted for the focused retrieval task.

What we learn from these results is still unclear but the figures are highly similar to those with the INEX 2007 topics.

6 Conclusion and future work

The biggest change in our system from 2007 took place in the scoring function that contributes over 90% of the total relevance score of each XML fragment. We discarded tfidf and replaced it with BM25 which assumes the probabilistic model for information retrieval. Thanks to that update, our results are now comparable with the best results overall.

A topic-specific analysis of the results is still to come.

References

1. Doucet, A., Lehtonen, M.: Let's phrase it: INEX topics need keyphrases. In: Proceedings of the SIGIR 2008 Workshop on Focused Retrieval. (2008) 9–14
2. Lehtonen, M., Doucet, A.: Phrase detection in the Wikipedia. In: Focused access to XML documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007. Volume 4862/2008 of Lecture Notes in Computer Science. (2008) 114–121
3. Lehtonen, M., Doucet, A.: XML-aided phrase indexing for hypertext documents. In: SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (2008) 843–844
4. Doucet, A., Ahonen-Myka, H.: Fast extraction of discontiguous sequences in text: a new approach based on maximal frequent sequences. In: Proceedings of IS-LTC 2006, "Information Society, Language Technology Conference". (2006) 186–191
5. Lemur: Lemur toolkit for language modeling and ir (2003)

CADIAL at INEX

Jure Mijić[1], Marie-Francine Moens[2], and Bojana Dalbelo Bašić[1]

[1]Faculty of Electrical Engineering and Computing, University of Zagreb,
Unska 3, 10000 Zagreb, Croatia
{jure.mijic, bojana.dalbelo}@fer.hr
[2]Department of Computer Science, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, 3001 Heverlee, Belgium
sien.moens@cs.kuleuven.be

Abstract. Structured document retrieval is becoming more popular with the increasing quantity of data available in XML format. In this paper, we describe a search engine model for structured document retrieval that uses language modelling and smoothing at the document and collection levels for calculating the relevance of each element of the query. Element priors, CAS query constraint filtering, and the +/- operators are also used in the ranking procedure. We also present the results of our participation in the 2008 INEX ad hoc track.

Key words: Focused retrieval, Index database, Language model, Search engine

1 Introduction

Information retrieval has become a part of our everyday lives. With the growing amount of available information, it has become challenging to satisfy a specific information need. We expect the retrieval procedure to find the smallest and most relevant information unit available, especially if our information need is very specific. Information retrieval procedures usually return whole documents as a result of a user query, but with the increasing number of structured XML data sources, the information unit size can be varied from whole documents to sections, paragraphs, or even individual sentences. The choice of an appropriate information unit size is left to the retrieval procedure, which determines which portions of a document are considered relevant. If the search procedure is returning parts of a document, it is necessary to eliminate overlapping content so that the user does not have to inspect duplicate content. This reduces the time it takes for the user to browse through the results.

The outline structure of the documents we are searching could also be known, so the user could specify additional structural constraints in the query, i.e., to return only relevant paragraphs or images. Such queries are called content-and-structure (CAS) queries, as opposed to content-only (CO) queries, which do not have those structural constraints and contain only the keywords from the query. Using CAS queries, the user can generate a more specific query that could improve the retrieved results.

The structure of the document could also be exploited in the ranking of the document elements. The location of the returned element in the document could indicate its potential relevance. For example, elements at the beginning of the document could be considered to be more relevant in most cases. Also, elements that are nested deep in the document structure could be considered less relevant.

In Section 2, we give an overview of our search engine and how it is modelled. In Section 3, we describe the ranking method used in our search engine and the data model that the method requires. In Section 4, we present and discuss our experimental results, and in Section 5, we give our concluding remarks.

2 System overview

Our search engine [3] was developed for the CADIAL project. The search engine provides access to a collection of Croatian legislative documents and has built-in support for morphological normalization. All of the documents have a similar structure, which consists of a title, introduction, body, and signature. Furthermore, the body is divided into articles, and each article into paragraphs. This document structure can prove to be useful, as it can be exploited by the retrieval procedures. This was the main motivation for our participation in the INEX ad hoc track, as the documents from the Wikipedia collection used in that track are also structured and written in XML format, with document structure tags such as article, body, section, paragraph, table, and figure.

For the purpose of text processing, we use the Text Mining Tools (TMT) library [5]. The most basic text processing operation is tokenization, which is implemented for use with the UTF-8 character set that we use for internal text representation. Input documents are in XML format, and any part of the document can be indexed. The search engine can also use an inflectional lexicon for morphological normalization, but we did not have a lexicon built for the English language, so we instead used stemming, specifically the Porter stemmer.

At the core of our search engine is an index database containing all words found in the document collection, along with their respective positions in the documents. Words are stored in their normalized form if morphological normalization is used, or stems of words are stored if stemming is used. The index database also contains additional statistical data needed for the new ranking method we implemented for the INEX ad hoc track: see remark further, as well as the structure of the elements for each document in the collection. The list of document elements to index is defined during the process of indexing, so we can choose which elements to index, i.e., article, body, section, paragraph, and figure, or to index the documents without their structure, i.e., only the article root tag. A document collection index database is built using an index builder tool, and than saved to a file in binary format. Serialization and deserialization procedures used are also implemented in the TMT library.

3 Ranking method and underlying data model

We implemented a new ranking method in our search engine that can support the document structure and be relatively straightforward and efficient to use on a large document collection. We also added support for CAS queries and the +/- keyword operators.

3.1 Language model

For our ranking method, we used language modelling [4]. The basic idea behind this method is to estimate a language model for each element, and then rank the element by the likelihood of generating a query with the given language model. Therefore, we can calculate the relevance of every element e to the specified query Q:

$$P(e|Q) = P(e) \cdot P(Q|e), \tag{1}$$

where $P(e)$ defines the probability of element e being relevant in the absence of a query; and $P(Q|e)$ is the probability of the query Q, given an element e. We estimated the element priors in the following way:

$$P(e) = \frac{1}{1 + e_{location}} \cdot \frac{1}{1 + e_{depth}}, \tag{2}$$

where $e_{location}$ is the local order of an element, ignoring its path; and e_{depth} is the number of elements in the path, including e itself. For example, for an element /article[1]/body[1]/p[5], the location value is 5, and its depth is 3. A similar formula for calculating element priors was used in previous work by Huang et al. [1]. We experimented with this formula, and found that changing the coefficients in the formula does not improve the results any further. The formula, in this simple form, yields noticeable improvements in the retrieval performance.

For a query $Q = (q_1, q_2, ..., q_m)$, assuming the query terms to be independent, $P(Q|e)$ can be calculated according to a mixture language model:

$$P(Q|e) = \prod_{i=1}^{m} (1 - \lambda_d - \lambda_c) P_{elem}(q_i|e) + \lambda_d P_{doc}(q_i|D) + \lambda_c P_{col}(q_i|C), \tag{3}$$

where λ_d is the smoothing factor for the document level; λ_c is the smoothing factor for the collection level; and $P_{elem}(q_i|e)$, $P_{doc}(q_i|D)$, $P_{col}(q_i|C)$ are probabilities of the query term q_i given the element, document, and collection, respectively. The smoothing is done on two levels: the document and the collection levels, with the restriction that $\lambda \in [0, 1]$ and $\lambda_d + \lambda_c < 1$. Wang et al. [6] found that smoothing on both the document and collection levels produced significantly better results than just smoothing on the whole collection. They used the Two-Stage smoothing method, and compared it to the Dirichlet priors and Jelinek-Mercer smoothing method. We chose to use our smoothing method

because the values of the smoothing factors λ_d and λ_c have a very intuitive meaning. Although we considered additional smoothing at the section level, we did not implement it, because the section elements could be nested in each other, so we would not have a constant number of smoothing factors.

The probabilities of the query term q_i given the element, document, or a collection are calculated in the following way:

$$P_{elem}(q_i|e) = \frac{tf(q_i|e)}{length(e)}, \tag{4}$$

$$P_{doc}(q_i|D) = \frac{tf(q_i|D)}{length(D)}, \tag{5}$$

$$P_{col}(q_i|C) = \frac{tf(q_i|C)}{length(C)}, \tag{6}$$

where $tf(q_i|e), tf(q_i|D), tf(q_i|C)$ are the term frequency of the query term q_i in the element, document, and collection, respectively; $length(e), length(D), length(C)$ are the length of the element, document, and collection, respectively, in terms of the number of words.

3.2 Ranking the elements

In the ranking procedure, other factors may influence the scores for each element. Elements are first scored using the language model formula 1, and then filtered according to the structural constraints from the CAS query, if there are any. For example, if the CAS query specifies that the user wants to find a figure, then elements that contain the element figure in their XPath are promoted to the top of the rank. The promotion of these elements is done sequentially from top to bottom, so the order of relevance for these elements is preserved.

We also implemented the +/- keyword operators, meaning that keywords from the query marked with the plus operator must be contained in the returned element, and keywords marked with the minus operator must not be contained in the element. For performance reasons, this operation is integrated in the calculation of the probabilities in the language model, so elements that do not satisfy the constraints of these operators, i.e., those that do not contain keywords marked with the plus operator or do contain keywords marked with the minus operator, are automatically assigned a score of zero.

Finally, when all the retrieved elements are ranked, we have to eliminate overlapping elements so that the ranking procedure does not return duplicate content. This is done simply by iterating through the results from top to bottom and eliminating elements whose XPath is fully contained in any of the previous elements' XPath, or if any of the previous elements' XPath is fully contained in the XPath of the element currently being analyzed.

Table 1. Tag set of indexed elements

No.	Tag name
1	article
2	body
3	section
4	p
5	table
6	figure
7	image

3.3 Index database

The index database is the backbone of our search engine. In the database, we store the positions of words in every element being indexed. Along with word indices, some other statistical data is also stored for use in language modelling. Each element is represented with its own language model, so some data must be stored separately for every element in the collection, e.g., term frequency for that element and the element length. The size of the index database is therefore dependent on the number of elements we want to index, so we chose to index only elements that are most likely to be relevant, as shown in Table 1.

The document as a whole is also considered as an element. Other data that is not stored for every element includes the term frequency for the entire collection, number of elements containing each term, unique term count, total term count, and total element count.

Collection information is also stored in the index database. Information such as the structure of the elements being indexed, i.e., the parent child relations of the elements in the document, needs to be stored in order for the language model to perform the necessary smoothing operations at the document and collection level, and also to reconstruct the proper XPath for every retrieved element.

4 Ad hoc results

Results for our runs are given in Table 2 and are sorted by the interpolated precision measure at 1% recall, i.e., iP[0.01], which is the official measure of the focused retrieval task in the INEX ad hoc track. Other measures include interpolated precision at other early levels of recall, i.e., iP[0.00], iP[0.05], iP[0.10], and mean average interpolated precision over 101 standard levels of recall, i.e., MAiP. The best result for each measure is marked in boldface. The name of the run contains the type of query used, i.e., CO for content-only and CAS for content-and-structure query. It also contains the returned information unit, i.e., document or element, and the smoothing factors used, i.e., *ld* for document level and *lc* for collection level. Note that the smoothing factors are in the range from 0.0 to 1.0 with the restriction that $\lambda_d + \lambda_c < 1$, so for the run *cas-element-ld5-lc4* the smoothing factors are $\lambda_d = 0.5$ and $\lambda_c = 0.4$.

Table 2. Official results for our runs

No.	Run	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
1	co-document-lc6	0.6389	**0.5949**	**0.5051**	**0.4699**	**0.2551**
2	cas-element-ld5-lc4	0.6684	0.5530	0.4048	0.3248	0.1440
3	co-element-ld2-lc5	**0.6907**	0.5417	0.4007	0.2920	0.0994
4	co-element-ld2-lc1	0.6718	0.5241	0.3922	0.2963	0.0929
5	cas-element-ld2-lc5	0.6494	0.5203	0.3569	0.2593	0.1134
6	cas-element-ld1-lc6	0.6642	0.5063	0.3652	0.2610	0.1133

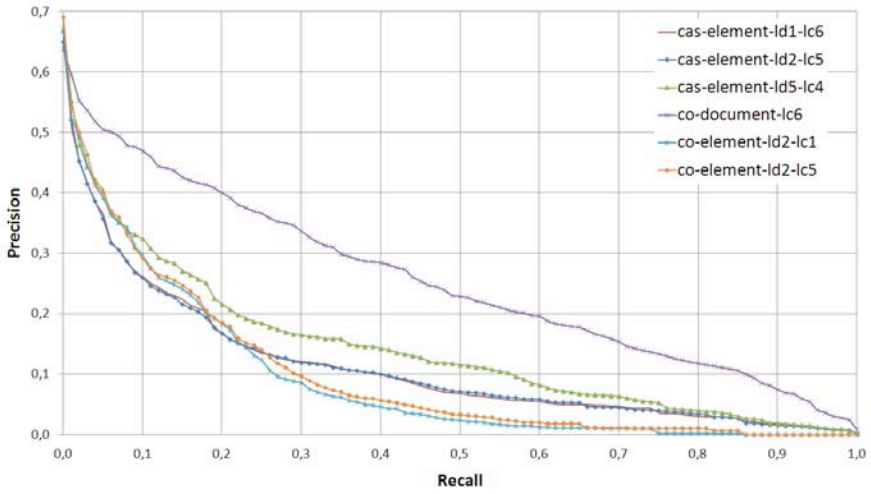


Fig. 1. Recall precision graph for our runs

Immediately from the results, we can see that the retrieval of the whole document gives better performance at higher levels of recall than element retrieval, as can be seen from the precision-recall graph in Fig. 1. Only at very low levels of recall, i.e., iP[0.00], does element retrieval outperform document retrieval. Similar results, where the document retrieval outperforms element retrieval at iP[0.01] and at higher levels, were seen in previous years and in some of the systems in this year's ad hoc track. This could, perhaps, be a consequence of the way in which we perform relevance assessments, where fully relevant documents are assigned to most of the topics. Another problem could be that the articles in the Wikipedia collection are very specific to their content, and the topics are usually not very specific. This leads to a situation where many articles are marked as fully relevant, and only a few have some specific relevant elements.

Smoothing factors also had a significant impact on the retrieval performance. As we mentioned previously, retrieving whole documents outperformed element retrieval at higher levels of recall, so it is reasonable to expect that higher smoothing at the document level would yield better results. This can be seen in our

cas-element-ld5-lc4 run in Fig. 1, where higher smoothing at the document level contributes to significantly better performance at higher levels of recall than other runs with lower smoothing at the document level, e.g., *cas-element-ld2-lc5* and *cas-element-ld1-lc6*. Liu et al. [2] also found that the document score greatly influenced the retrieval performance. They implemented a separate document and element index, and combined the document and element score.

The use of CAS query constraint filtering did improve retrieval performance overall, especially at midrange levels of recall. At low levels of recall the difference is not significant, and even at iP[0.00], the performance is slightly worse than the run using CO queries. Perhaps more complex processing of CAS queries could yield some improvement at low levels of recall, although most of the topics did not use the structural features of CAS queries.

Although we did not do a direct comparison on the influence of the +/- keyword operator and element priors on the retrieval performance, we did notice during development that using both the operators and element priors did in fact improve performance slightly.

5 Conclusion

We developed a search engine for structured document retrieval and implemented a simple ranking method that uses language modelling and smoothing at two levels: the document and the collection level. Retrieving whole documents performed better than element retrieval at higher levels of recall, which could perhaps be attributed to the nature of the topics. Element retrieval performed better than document retrieval only at the lowest level of recall, i.e., iP[0.00]. Filtering of elements' structural path to the CAS query constraints contributed to the improvement in retrieval performance, as well as the higher smoothing factor at the document level. We have also used element priors and implemented the +/- keyword operators, which we noticed tend to improve the retrieval performance, but we did not investigate their impact on performance in detail.

We developed our system from the ground up, putting an emphasis on simplicity, efficiency, and effectiveness. Language modelling proved to be very effective, and yet relatively simple. This was our first year participating in the INEX ad hoc track, and we are pleased with the results. There is much room left for improvements, e.g., relevance feedback and incorporating link evidence, but we will leave that for future years.

Acknowledgments. This work was performed at Katholieke Universiteit Leuven, during the first author's stay as a visiting scholar. This work has been jointly supported by the Ministry of Science, Education and Sports, Republic of Croatia and the Government of Flanders under the grant No. 036-1300646- 1986 and KRO/009/06 (CADIAL).

References

1. F. Huang. The role of shallow features in XML retrieval. In *INEX 2007 Workshop Proceedings*, pages 33–38, 2007.
2. J. Liu, H. Lin, and B. Han. Study on reranking XML retrieval elements based on combining strategy and topics categorization. In *INEX 2007 Workshop Proceedings*, pages 170–176, 2007.
3. J. Mijić, B. Dalbelo Bašić, and J. Šnajder. Building a search engine model with morphological normalization support. In *ITI 2008 Proceedings of the 30th International Conference on Information Technology Interfaces*, pages 619–624, 2008.
4. Jay M. Ponte and W. Bruce Croft. 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*, pages 275–281, New York, NY, USA, 1998. ACM.
5. A. Šilić, F. Šarić, B. Dalbelo Bašić, and J. Šnajder. TMT: Object-oriented text classification library. In *ITI 2007 Proceedings of the 29th International Conference on Information Technology Interfaces*, pages 559–566, 2007.
6. Q. Wang, Q. Li, and S. Wang. Preliminary work on XML retrieval. In *INEX 2007 Workshop Proceedings*, pages 70–76, 2007.

Indian Statistical Institute at INEX 2008 Adhoc track

Sukomal Pal, Mandar Mitra, Debasis Ganguly, Samaresh Maiti, Ayan Bandyopadhyay, Aparajita Sen, and Sukanya Mitra

Information Retrieval Lab, CVPR Unit,
Indian Statistical Institute, Kolkata
India
{sukomal_r, mandar, samaresh_t, ayan_t, aparajita_t, sukanya_t}@isical.ac.in,
debasis@synopsys.com

Abstract. This paper describes the work that we did at Indian Statistical Institute towards XML retrieval for INEX 2008. Beside the Vector Space Model (VSM) that we have been using since INEX 2006, this year we implemented Language Model (LM) in our text retrieval system (SMART) to retrieve XML elements against the INEX Adhoc queries. Like last year, we considered Content-Only (CO) queries and submitted three runs for the FOCUSED sub-task. Two runs are from Vector Space Model and one using Language Model. One run from each model is at document level, while the third run at the element level using VSM approach. We applied blind feedback for both the runs from VSM approach. For the run with LM approach it was our first attempt to implement LM in the SMART system and then to customize it for XML retrieval. Though this run was of preliminary nature without any feedback, its performance is quite satisfactory, better than VSM runs. In general, relative performance of our document-level runs are respectable, and still much way ahead of the element level retrieval run. Our immediate next task is therefore to focus on how to improve element 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 also permitted as retrievable units, besides the usual XML elements. A retrieved passage can be a sequence of textual content either from within an element or spanning a range of elements. Since INEX 2007 also classified the adhoc retrieval task 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.

Each of the three subtasks can be based on two different query variants: Content-Only(CO) and Content-And-Structure(CAS) queries. In the CO task, the user poses the 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.

This year we submitted three adhoc focused runs, two from Vector Space Model (VSM) based approach and one from Language Modelling (LM) approach. 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.

In LM, probability of a document generating the query terms are taken as the measure of similarity between the document and the query. However the difference lies in the fact that queries are explicitly modelled as a sequence of query terms, not a set of query terms. The relative positions of the query terms in the query matters during the probability calculations.

We used our modified SMART system for the experiments at INEX 2008. Earlier we had customized the SMART text retrieval system for XML element retrieval based on VSM approach. We submitted two such runs, one at the document retrieval level, and the other at the element level. For both these runs we used blind feedback as post-processing of initial document retrieval.

This year we implemented LM into SMART as well and equipped it with XML retrieval. Our third run was based on this implementation at the document level.

All three runs was for the *FOCUSED* sub-task of the Adhoc track considering CO queries only.

In the following section we describe our general approaches for all these runs, and discuss results and further work in Section 3.

2 Approach

Like last year, to extract the useful parts of the given documents, we short-listed about thirty tags that contain useful information: <*p*>, <*ip1*>, <*it*>, <*st*>, <*fnm*>, <*snm*>, <*atl*>, <*ti*>, <*p1*>, <*h2a*>,<*h*>, <*wikipedialink*>, <*section*>, <*outsidelink*>, <*td*>, <*body*>, 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 [3]. Stopwords were removed

in two stages. First, we removed frequently occurring common words (like *know*, *find*, *information*, *want*, *articles*, *looking*, *searching*, *return*, *documents*, *relevant*, *section*, *retrieve*, *related*, *concerning*, etc.) from the INEX topic-sets. Next, words listed in the standard stop-word list included within SMART were removed from both documents and queries. Words were stemmed using a variation of the Lovin's stemmer implemented within SMART. Frequently occurring word bi-grams (loosely referred to as phrases) were also used as indexing units. We used the N-gram Statistics Package (NSP)[1] on the English Wikipedia text corpus 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* [4] term-weighting formula. For each of 135 adhoc queries(544-678), we retrieved 1500 top-ranked XML documents or non-overlapping elements.

2.1 Document-level Run

We submitted two runs at the document level retrieval. One was based on LM approach and the other one on VSM approach.

LM approach: The run *LM-nofb-0.20* was based on language modelling approach. We implemented a language modelling framework within the SMART system [5].Since, the SMART system was designed specifically for VSM, we had to cast it in terms of VSM for smooth integration within SMART. Here we briefly introduce the key concepts of LM, its relationship with VSM and our experiments with the LM implementation based on the work of Hiemstra [6].

The model uses the following definitions.

Let D be a discrete random variable denoting the document which the user has in mind. The sample space of D is the finite set comprising of all documents in the collection $\{d^1, \ldots, d^N\}$.

Let I_i be a discrete random variable denoting the "importance of the ith query term" over the sample space 0, 1, where 0 stands for unimportant and 1 for important.

Let T_i be a discrete random variable denoting "the ith query term", which sample space contains a finite number of points $\{t^{(1)}, \ldots, t^{(m)}\}$ each referring to an actual term in the collection, where m refers to the no. of words in the dictionary built on the given corpus.

The joint probability $P(D, I_1, \ldots, I_n, T_1, \ldots, T_n)$ completely defines the information retrieval problem for a query of length n [6]. A query is generated by first selecting a document d with probability $P(D = d)$. Given that d is the document the user has in mind, tossing for importance and selecting the query terms is done independently for each query term t_i with probabilities $P(I_i)$ and $P(T_i|I_i, D)$ respectively. Hence,

$$P(D, I_1, \ldots, I_n, T_1, \ldots, T_n) = P(D) \prod_{i=1}^{n} P(I_i) P(T_i|I_i, D) \quad (1)$$

[1] http://www.d.umn.edu/~tpederse/nsp.html

Distributing sums over the products we get

$$P(D, T_1, \ldots, T_n) = P(D) \prod_{i=1}^{n} \sum_{k=0}^{1} P(I_i = k) P(T_i | I_i = k, D) \quad (2)$$

Ranking the documents by (2) will in fact rank the documents in decreasing order of the probability that the document is relevant given the query.

The probabilities are defined by using the number of documents in the collection and the term frequencies of a term in a document. Let $tf(t, d)$ denote the term frequency of term t in document d.

From the urn model, it is quite straightforward to see that:

$$P(D = d) = \frac{1}{\#\ of\ documents} \quad (3)$$

$$P(T_i = t_i | I_i = 1, D = d) = \frac{tf(t_i, d)}{\sum_t tf(t, d)} \quad (4)$$

$$P(T_i | I_i = 0) = \frac{\sum_k tf(t_i, k)}{\sum_{t,k} tf(t, k)} = \frac{cf(t_i)}{cf(t)} \quad (5)$$

where $cf(t_i)$ is the collection frequency of the term t_i and $cf(t)$ is the collection size.

Alternatively, the collection frequencies can be replaced with document frequencies to maintain similarity with VSM, which uses document frequencies:

$$P(T_i) = \frac{df(t_i)}{df(t)} \quad (6)$$

To make the notations simpler: let $P(I_i = 1)$ be denoted by λ_i and hence $P(I_i = 0)$ would become $(1 - \lambda_i)$. Let $P(T_i | I_i = 1, D)$ be denoted by $P(T_i | D)$ and $P(T_i | I_i = 0)$ be replaced with $P(T_i)$.

As a result (2) becomes

$$P(D, T_1, \ldots, T_n) = P(D) \prod_{i=1}^{n} ((1 - \lambda_i) P(T_i) + \lambda_i P(T_i | D)) \quad (7)$$

From LM to VSM

Dividing (7) by $\prod_{i=1}^{n} (1 - \lambda_i) P(T_i)$ wouldn't affect the ranking because λ_i and $P(T_i)$ have the same value for each document. Infact any monotonic transformation of the document ranking function will produce the same ranking of the documents. Instead of using the product of weights, beacuse of the obvious diasdvantage of decrease in the probability values after successive multiplications, the formula can be implemented by using the sum of logarithmic weights. Doing so and replacing $P(D)$, $P(T_i | D)$ and $P(T_i)$ by the definitions in equations (3), (4) and (6) results in:

$$P(D_j = d_j, T_1 = t_1, \ldots, T_n = t_n) = \sum_{i=1}^{n} log(1 + \frac{\lambda_i tf(t_i, d) \sum_t df(t)}{(1 - \lambda_i) df(t_i) \sum_t tf(t, d)}) \quad (8)$$

Also we can associate non uniform prior probabilities to document relevance resulting in

$$P(D_j = d_j, T_1 = t_1, \ldots, T_n = t_n) = log(\sum_t tf(t,d)) + \sum_{i=1}^{n} log(1 + \frac{\lambda_i tf(t_i,d) \sum_t df(t)}{(1-\lambda_i) df(t_i) \sum_t tf(t,d)}) \quad (9)$$

For *nnn* query weights i.e. $q_k = tf$ and

$$d_k = log(1 + \frac{tf \times (sum\ of\ dfs)}{df \times document\ length} \times \frac{\lambda_k}{1-\lambda_k}) \quad (10)$$

($k = 1, \ldots, m$).and $\lambda_k = \lambda$ for all k).

the vector product of the two, d_k and q_k is identical to the rhs of equation (9) and hence is an useful estimation of $P(D_j = d_j, T_1 = t_1, \ldots, T_n = t_n)$ acting as measure for similarity score.

So, for a system with the vector data structure implemented in it, reweighting the document vectors and using simple term frequencies as weights of the query vectors would give the LM scores with the dot product operation on vectors. Our LM run was only at the document level with no feedback using $\lambda_k = \lambda = 0.20$ for all k.

VSM approach: For run, *VSMfb*, we retrieved whole documents only using blind feedback. 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 $\{t_l, \ldots, t_m\}$ 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

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

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 subsequent match is deprecated depending on how strongly this match was

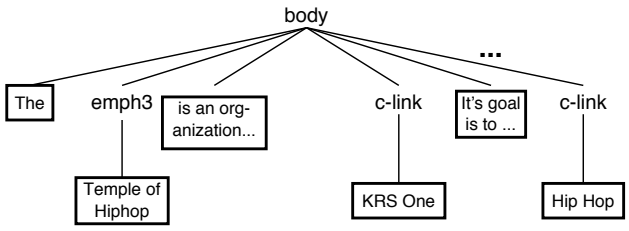


Fig. 1. Parse tree for a fragment of a wikipedia document

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:

$$\begin{aligned} \text{number of words} &= 20 \\ \text{number of phrases} &= 5 \\ \text{Rocchio } \alpha &= 4 \\ \text{Rocchio } \beta &= 4 \\ \text{Rocchio } \gamma &= 2. \end{aligned}$$

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

2.2 Element-level Run

For the element-level retrieval, we adopted a 2-pass strategy. In the first pass, we retrieved 1500 documents for each query using query expansion and blind feedback.

In the second pass, these documents were parsed using the libxml2 parser, and leaf nodes having textual content were identified. Figure 1 shows a fragment of a file from the wikipedia collection. The leaf nodes that have textual content are enclosed in rectangles in the figure. The total set of such leaf-level textual elements obtained from the 1500 top-ranked documents were then indexed and compared to the query as before to obtain the final list of 1500 retrieved elements.

As is clear from figure 1, permitting only leaf-level textual elements to be retrieved has an obvious disadvantage: nodes such as <p> or <body> are very often not considered retrievable elements, because of the occurrence of nodes like

<emph3> and <collectionlink> under the <p> or <body> node. It is not surprising, therefore, that the *VSMfbElement* run performs poorly.

We incorporated within SMART the capability to retrieve elements at intermediate (i.e. non-leaf) levels of the XML tree. Here the query is compared to all elements that contain text, instead of only the leaf-level textual nodes. In order to avoid any overlap in the final list of retrieved elements, the nodes for a document are sorted in decreasing order of similarity, and all nodes that have an overlap with a higher-ranked node are eliminated.

3 Results

The results as reported in the inex website over relevance judgment for 70 topics are shown in Table 1.

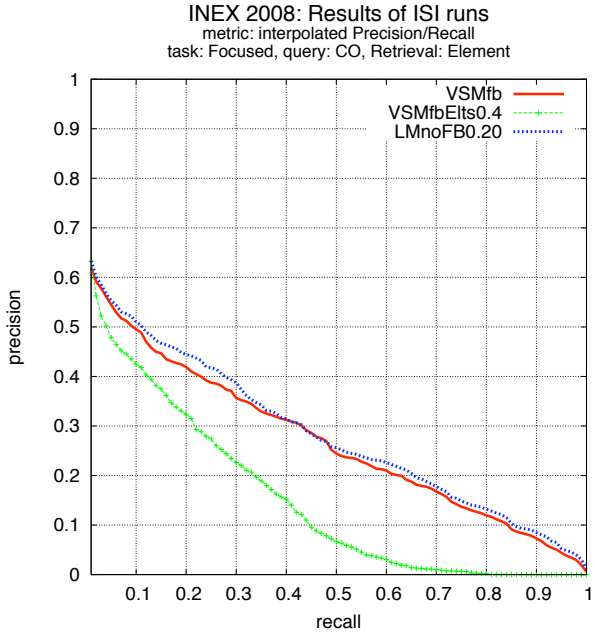
Table 1. Metric: Interpolated Precision/Recall task: FOCUSED , CO retrieval unit: Element

Run Id	iP@0.00	iP@0.01	iP@0.05	iP@0.10	MAiP	Official Rank
VSMfb	0.6318	0.6197	0.5451	0.4950	0.2708	24/61
VSMfbElts0.4	**0.7148**	0.6325	0.4782	0.4248	0.1531	18/61
LM-nofb-0.20	0.6830	**0.6337**	0.5537	0.5100	**0.2847**	**17/61**
Best Run (FOERStep)	n/a	**0.6873**	n/a	n/a	n/a	1

For the top-performing run, the scores other than the official one (iP@0.01) is not available (n/a) at the moment. But interestingly, if we consider precision at the first retrieval unit (iP@0.00), our element run *VSMfbElts0.4* is the best. Though if we consider overall performance, our document-level runs (*VSMfb* and *LM-nofb-0.20*) are far more better. One possible reason is the use of pivoted length normalization [8] without having the knowledge of pivot and slope for wikipedia collection [9]. For the VSM-based runs we considered $slope = 0.4$ and $pivot = 80$, which was seen to be yielding best result for early precision.

4 Conclusion

This was our third year at INEX. Our main objective this year was to see the performance of LM approach vis-a-vis VSM. Therefore we implemented LM into SMART retrieval system. Its performance is certainly assuring as it fares the best among our runs. However the retrieval was at the document-level only. We need to extend it to element-level retrieval. For the VSM runs, document-level run is quite satisfactory. Though for the element-level run, there is plenty of room for improvement. We need to study the effect of document length normalization in-depth for XML collection and adopt a suitable strategy for wikipedia corpus.



There is also enough scope of studying and thereby proposing effective term-weighting scheme for different element-tags in the XML tree. We hope these will be an exciting exercises which we plan to do in the coming days.

References

1. : INEX: Initiative for the Evaluation of XML Retrieval (2008) http://www.inex.otago.ac.nz.
2. W3C: XPath-XML Path Language(XPath) Version 1.0 http://www.w3.org/TR/xpath.
3. Salton, G.: A Blueprint for Automatic Indexing. ACM SIGIR Forum **16**(2) (Fall 1981) 22–38
4. 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). Volume NIST Special Publication 500-238. (1997)
5. Ganguly, D.: Implementing a language modeling framework for information retrieval. Master's thesis, Indian Statistical Institute (2008)
6. Hiemstra, D.: Using language models for information retrieval. PhD thesis, University of Twente (2001)
7. Mitra, M., Singhal, A., Buckley, C.: Improving automatic query expansion. In: SIGIR 98, Melbourne, Australia, ACM (1998) 206–214
8. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference

on Research and development in information retrieval, New York, NY, USA, ACM (1996) 21–29
9. 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

Search for Fast and High-precision XML Retrieval

Hiroki Tanioka

Innovative Technology R&D, JustSystems Corporation,
108-4 Hiraishi-Wakamatsu Kawauchi-cho Tokushima-shi Tokushima, Japan
hiroki.tanioka@justsystems.com

Abstract. This paper reports experimental results our approach using the vector space model for retrieving large-scale XML data, to improve the retrieval precision on the Evaluation of XML Retrieval (INEX) 2008 Adhoc Track. The vector space model is commonly used in the information retrieval community. Last year, for the Evaluation of XML Retrieval (INEX) 2007 Adhoc Track, we developed a system using a relative inverted-path (RIP) list and a Bottom-UP Scheme (BUS) approach. The system made it possible to search faster than ever for XML data. However the system has a room for improvement in terms of the retrieval precision. To improve the retrieval precision, the system tries two methods, using CAS title and using Pseudo Relevance Feedback (PRF).

1 Introduction

There are two approaches for XML information retrieval (IR): one based on database models, the other based on information retrieval models. Our system is based on the vector space model[3] from information retrieval. In the field of information retrieval, the retrieval unit returned by IR systems is typically a whole document or a document fragment, such as a paragraph in passage retrieval. Traditional IR systems based on the vector space model compute a postings file as term vectors for each retrieval unit, and calculate similarities between the units and the query. Specifically, the postings file maps each XML node from words, and the query consists of some words.

Our system uses keywords (multi-word terms, single words) as the query and separates XML [1] documents into two parts: content information (the keywords) and structural information. XML nodes correspond to retrieval units, and nodes that include query terms can be quickly retrieved using an inverted-file list. For very large XML documents, all XML nodes are indexed to each term directly included in the node itslef, but not the node's children or more distantly related nodes. During the retrieval phase, the score of a retrieved node is calculated by merging the scores from its descendant nodes. To merge scores while identifying parent-child relationships, the system employs a relative inverted-path list (*RIP* list) that uses nested labels with offsets to save the structural information.

In the INEX 2008, our experiment targets the CO Task and the CAS Task. The system accepts CO queries, which are terms enclosed in <title> tags. Therefore, the system can accept CAS queries as constrained condition, which are XPath[2] representations enclosed in <castitle> tags. For improving the retrieval precision, the system adopts Pseudo Relevance Feedback (PRF).

The rest of this article is divided into three sections. In section 2, we describe the IR model for XML documents. In section 3, we describe experimental results. And in section 4, we discuss results and future work.

2 XML Information Retrieval

Our system uses a TF-IDF scoring function for retrieval. TF-IDF is additive, therefore a node score can be easily calculated by merging the scores of its descendant nodes. The TF-IDF score L_j of the jth node is composed of the term frequency tf_i of the ith term in the query, the number of nodes f_i including the ith term, and the number of all the nodes n in the XML collection.

$$L_j = \sum_{i=1}^{t} tf_i \cdot \log(\frac{n}{f_i}) \tag{1}$$

However, if the node score is the sum of the scores of its descendants, there is the problem that the root node always has the highest score in the document. Therefore, the score R_j of the jth node is composed of the number T_j of terms contained in the jth node, the score L_k of the kth descendant of the jth node, and the number T_k of terms contained in the kth node.

$$R_j = \sum_{k\ childrenof\ j} D \cdot L_k \tag{2}$$

$$T_j = \sum_{k\ childrenof\ j} T_k \tag{3}$$

where $D(= 0.75)$ is a decaying constant, which is as shown in the following equation. Then the TF-IDF score R_j is normalized by the number of terms T_j.

$$R'_j = \frac{R_j}{T_j} \tag{4}$$

Then s_j is the occurrence of terms included in both the query and jth node,

$$s_j = count(\delta_j), \qquad \delta_j = \bigcup_{k\ childrenof\ j} \gamma_k,$$

where α is the set of terms included in the query, and β_j is the set of terms included in the jth node. The conjunction, $\gamma_j = \alpha \cap \beta_j$, is the set of query terms included in the jth node.

After that, the score RSV_j of jth node is conposed of the TF-IDF score R'_j and the S_j, which is one of the heuristic scores we call a leveling score. If a node contains all terms in the query, the leveling score is the highest,

$$RSV_j = R'_j + \frac{Q \cdot s_j}{q} \tag{5}$$

where $Q(= 30)$ is a constant number and q is the number of terms in the query.

Then the retrieved results are chosen from the node list, which is sorted in descending order of RSV scores. All the thresholds are determined from a few topics of the Focused task of the INEX 2007[4].

In addtion, the PRF method considers the 10 top-ranked XML nodes retrieved by the baseline run, and chooses the 20 top-ranked terms in the TF-IDF score as additional query terms.

3 Experimental Results

3.1 INEX 2008 Adhoc Track

To index the XML 2008 Adhoc Track document collection, the system first parses all the structures of each XML document with an XML parser and then parses all the text nodes of each XML document with an English parser. The size of the index containing both content information and structure information is about 8.32 GB. Thereafter, the system uses the same index in every experiment.

Our experiment targets the CO Task and the CAS Task. The system accepts CO queries, which are terms enclosed in <title> tags. Therefore, the system can accept CAS queries with XPath as constrained condition, which are XML tags enclosed in <castitle> tags.

For each query set, there is the Focused task, the Relevant in Context task, and the Best in Context task in the INEX 2008 Adhoc Track. All the tasks are as same tasks as in the INEX 2007 Adhoc Track. Hence the system parameters are tuned for the Focused task on topics from INEX 2007.

The system is installed on the same PC used on INEX 2006 and INEX 2007 for comparison. The PC has 2GHz CPU, 2GB RAM, and 300GB SATA HDD, and the system is implemented in Java 1.6.0_07. The time it takes to parse and load the 659,388 files on the PC is about 8.17 hours excluding file-copying time. The database size is about 3.18 GB on HDD.

3.2 Experimental Results

Table 1 shows results for the three Adhoc tasks using the main evaluation measure for INEX 2008. Unfortunately, There are some bugs in our system, so the listed results include Topic 544–650 for the Forcused task only.

Table 1. Top 10 of 29 participants on Focused

Run ID	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	AiP
RIP_01	0.4156	0.2947	0.2257	0.1719	0.0599
RIP_02	0.4192	0.3076	0.2477	0.1975	0.0686
RIP_03	0.3502	0.2961	0.2280	0.1814	0.0635

RIP_01 was a baseline using TF-IDF, RIP_02 was using CAS titles, and RIP_03 was effected by PRF method.

4 Conclusions

From the results RIP_03 was greater than RIP_01 as AiP, but lesser than RIP_01 as iP[0.00]. Because RIP_03 was effected by PRF method, so RIP_03 was slightly up to RIP_01. RIP_02 was the best score using CAS titles, there was ranked 41th with 0.5535 in the official ranking.

This paper reports the effectiveness of both the CAS Titles and the PRF method. Hence we have to invest reasons of bugs and should evaluate the effectiveness regarding both using the CAS Titles and using the PRF method.

References

1. Extensible Markup Language (XML) 1.1 (Second Edition). *http://www.w3.org/TR/xml11/*
2. XML Path Language (XPath) Version 1.0. *http://www.w3.org/TR/xpath*
3. Salton, G., Wong, A. and Yang, C. S.: *A vector space model for automatic indexing*, Communications of the ACM, **18**, pp. 613–620 (1975).
4. Tanioka, H.: A Fast Retrieval Algorithm for Large-Scale XML Data, *Focused Access to XML Documents*., LNCS, Vol. 4862, Springer-Verlag, pp. 129–137 (2008).

Using collectionlinks and documents as Context for INEX 2008

Delphine Verbyst[1] and Philippe Mulhem[2]

[1] LIG - Université Joseph Fourier, Grenoble, France
Delphine.Verbyst@imag.fr
[2] LIG - CNRS, Grenoble, France
Philippe.Mulhem@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 2008 Ad Hoc Track. We study here the use of non structural relations between document elements (doxels) in conjunction with document/doxel structural relationships. The non structural links between doxels of the collection come from the *collectionlink* doxels. We characterize the non structural relations with relative exhaustivity and specificity scores. Results of experiments on the test collection are presented. Our best run is in the top 5 for iP[0.01] values for the Focused Task.

1 Introduction

This paper describes the approach used for the Ad Hoc Track of the INEX 2008 competition. Our goal here is to show that the use of non structural links and the use of structural links lead to high quality results for an information retrieval system on XML documents. We consider that handling links between doxels in a "smart" way may help an information retrieval system, not only to provide better results, but also to organize the results in a way to overcome the usual simple list of documents. For INEX 2008 runs, we obtained very good results for low recall values (0.00 and 0.01).

The use of non structural links, such as Web links or similarity links has been studied in the past. Well known algorithms such as Pagerank [1] or HITS [3] do not integrate in a seamless way the links in the matching process. Savoy, in [6], showed that the use of non structural links may provide good results, without qualifying the strength of the inter-relations. In [7], Smucker and Allan show that similarity links may help navigation in the result space. Last year, for our first participation at INEX [9], our approach using non-structural links and a vector space model outperformed runs without using the links. This year, we go further by refining the non structural relationships used, and by integrating during the matching phase the RSV of the document that contains a relevant doxel.

In the following, the non structural relations between doxels will be referred to as the *non structural context* of the doxels. Our assumption is that document parts are not only relevant because of their content, but also because they are related to other document parts that answer the query. The document that contains a doxel d will be referred as the *structural context* of d. In some way, we revisit the *Cluster Hypothesis* of van Rijsbergen [8], by considering that the relevance value of each document is impacted by the relevance values of related documents.

In our proposal, we first build inter-relations between doxels, and then characterize these relations using relative exhaustivity and specificity at indexing time. We also build explicit relationships between the doxels and their container document. These two elements are used by the matching process.

The five officially submitted runs by the LIG for the Ad Hoc track integrate such non structural and structural links. We submitted three runs for the Focused task, and two runs for the Relevant in Context task.

The rest of this paper is organized as follows: we describe the non structural links that were used in our experiments in part 2, the structural links used in part 3, then the doxel space is described in detail in section 4, in which we propose a document model using the context. Section 5 introduces our *matching in context* process. Results of the INEX 2008 Ad Hoc track are presented in Section 6.

2 Non structural context

The idea of considering non structural neighbours was proposed in [10], in order to facilitate the exploration of the result space by selecting the relevant doxels, and by indicating potential good neighbours to access from one doxel.

The INEX 2008 collection contains several links between documents, like *unknownlinks*, *languagelinks* and *outsidelinks* for instance. We considered existing relations between doxels with the *collectionlink* tag, because these links denote links inside the collection. The most important attribute for such tags for us is $xlink : href$, that indicates the target of the link. The targets of such links are only whole documents, and not documents parts. That is why we extended the initial links doxel/document by doxel to doxels links according to the following process, by considering a collectionlink doxel c with a document target ct:

- assume that c is composing a doxel d of type Td,
- we compute the similarity (VSM, cosine based) between d and all components of ct of type Td,
- we keep all the components above with the matching value greater than a threshold T,
- we generate all of these components of ct in the context of c.

– we keep the initial link going from c to the document ct.

We do not consider the links between collectionlinks, because collectionlinks may be too small. Initially, there are 17 013 512 collectionlinks in the INEX 2008 collection. With the process described above we generated 115 million of non structural links.

3 Structural links

We describe now the use of the structural links as they are used in our proposal. These links come only from the transitive closure of the compositional links between doxels, filtered to keep only the document links. When there exists a compositional link going from a document D to a doxel d, we generate a structural link going from the doxel d to the document D . We generated then 28 million of structural links.

4 Doxel space

4.1 Doxel content

The representation of the content of doxel d_i is a vector generated from a usual vector space model using the whole content of the doxel: $d_i = (w_{i,1}, ..., w_{i,k})$. Such a representation has proved to give good results for structured document retrieval [2]. The weighting scheme retained is a simple $tf.idf$, with idf based on the whole corpus and with the following normalizations: the tf is normalized by the max of the tf of each doxel, and the idf is log-based, according to the document collection frequency. To avoid an unmanageable quantity of doxels, we kept only doxels having the following tags: article, p, collectionlink, title, section, item. The reason for using only these elements was because, except for the collectionlinks, we assume that the text content for these doxels are not too small. The overall number of doxels considered by us here is 29 291 417.

4.2 Characterizing non structural doxel context

To characterize the non structural relations between doxels, we propose to define relative exhaustivity and relative specificity. These features are inspired from the definitions of specificity and exhaustivity proposed at INEX 2005 [4], and are supposed to define precisely the nature of the link. Consider a non compositional relation from the doxel d_1 to the doxel d_2:

– The relative specificity of this relation, noted $Spe(d_1, d_2)$, denotes the extent to which d_2 focuses on the topics of d_1. For instance, if d_2 deals only with elements from d_1, then $Spe(d_1, d_2)$ should be close to 1.

– The relative exhaustivity of this relation, noted $Exh(d_1, d_2)$, denotes the extent to which d_2 deals with all the topics of d_1. For instance, if d_2 discusses all the elements of d_1, then $Exh(d_1, d_2)$ should be close to 1.

The values of these features are in $[0, 1]$. We could think that these features behave in an opposite way: when $Spe(d_1, d_2)$ is high, then $Exh(d_1, d_2)$ is low, and vice verse.

Relative specificity and relative exhaustivity between two doxels are extensions of the overlap function [5] of the index of d_1 and d_2: these values reflect the amount of overlap between the source and target of the relation. We define relative specificity and relative exhaustivity on the basis of the non normalized doxel vectors $w_{1,i}$ and $w_{2,i}$ (respectively for d_1 and d_2) as follows.

We estimate values of the exhaustivity and the specificity of d_1 and d_2, based on a vector where weights are $tf.idf$

$$Exh(d_1, d_2) = \frac{\sum_{i|w_{2,i}\neq 0} w_{1,i}^2}{\sum_i w_{1,i}^2} \quad (1)$$

$$Spe(d_1, d_2) = \frac{\sum_{i|w_{1,i}\neq 0} w_{2,i}^2}{\sum_i w_{2,i}^2} \quad (2)$$

These two values scores are in $[0, 1]$ if we assume that no doxel is indexed by a null vector.

5 Matching in context

As we have characterized the doxel context, the matching process should return doxels relevant to the user's information needs regarding both content and structure aspects, and considering the context of each relevant doxel.

We define the matching function as a linear combination of a standard matching result without context, a matching result based on relative specificity and exhaustivity, and a matching coming from the rank of the documents according to the query processed. The relevant status value $RSV(d, q)$ for a given doxel d and a given query q is thus given by:

$$RSV(d, q) = \alpha * RSV_{content}(d, q) + (1 - \alpha) * RSV_{context}(d, q) \quad (3)$$
$$+rank(RSV_{content}(Doc(d), q)),$$

where $\alpha \in [0, 1]$ is experimentally fixed, $RSV_{content}(d, q)$ is the score without considering the set of neighbours $\mathcal{V}_d$ of d (i.e. cosine similarity),

$$RSV_{context}(d,q) = \sum_{d' \in \mathcal{V}_d} \frac{\beta * Exh(d,d') + (1-\beta) * Spe(d,d')}{|\mathcal{V}_d|} RSV_{content}(d',q) \quad (4)$$

where $\beta \in [0,1]$ is used to focus on exhaustivity or specificity, and $rank(RSV_{content}(Doc(d),q))$ the rank of the matching result for each document of the collection. According to what is done now, the part of the fomula above weighted by α and $1-\alpha$ is in [0, 1], so adding the ranking of the documents induces a grouping by document of the doxels.

6 Experiments and results

The INEX 2008 Adhoc track consists of three retrieval tasks: the Focused Task, the Relevant In Context Task, and the Best In Context Task. We submitted 3 runs for the Focused Task, and 2 runs for the Relevant In Context Task. For all these runs, we used only the *title* of the INEX 2008 queries as input for our system: we removed the words prefixed by a '-' character, and we did not consider the indicators for phrase search. The size of the vocabulary we used is 210 000.

First of all, we have experimented our system with INEX 2008 collection to tune the α and β parameters and the number of non structural neighbours used. The best results were achieved with $\alpha = 0.5$ and $\beta = 0.0$, which means that the non structural context is as important as the context of the doxels, and that only the exhaustivity is considered. We considered 4 non structural neighbours. For the use of the structured context, we tested a ranking based on a Vector Space Model (similar to what was described earlier), namely VSM, and a ranking based on a Language Model using a Dirichlet smoothing, namely LM.

6.1 Focused Task

The INEX 2008 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.

For the runs LIG-ML-FOCRIC-4OUT-05-00 and LIG-VSM-FOCRIC-4OUT-05-00, as explained by the matching formula, the results are grouped by document, so results are somewhat similar to RIC results (except for the ordering of the doxels in one document).

The last run, namely, FOC-POSTLM-4OUT-05-00 is a bit different in nature: we generated a binary value (1 for relevant document and 0 for non relevant document) for the document matching, and we filter the doxels belonging to relevant documents, without changing their matching value.

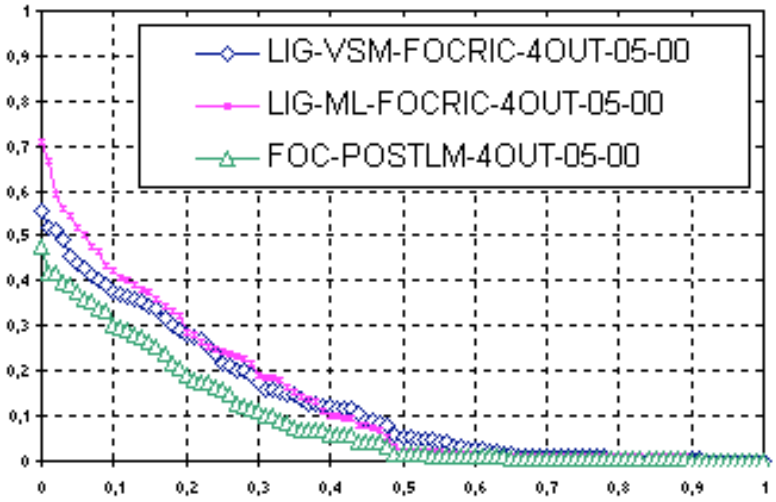
We present our results for the focused task in Table 1 showing precision values at given percentages of recall, and in Figure 1 showing the generalized precision/recall curve. These results show that runs based on the use of the

Language Model outperform at recall values lower than 0.2 the VSM document based context. Between recall values of 0.3 and 0.5, the VSM gives better results than the LM. The post processing using language model document matching does not give good results. Each of these curves drop sharply, which leads to a MAiP low compared to other participants runs.

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

Run	precision at 0.0 recall	precision at 0.01 recall	precision at 0.05 recall	precision at 0.10 recall
$LIG-ML-FOCRIC$ $MAiP = 0.1441$	0.7114	0.6665	0.521	0.4216
$LIG-VSM-FOCRIC$ $MAiP = 0.1339$	0.5555	0.5187	0.4402	0.3762
$FOC-POSTLM$ $MAiP = 0.0958$	0.4754	0.4188	0.3741	0.3035

Fig. 1. Interpolated Precision/Recall - Focused Task LIG Ad Hoc



6.2 Relevant In Context Task

For the Relevant In Context Task, we take "default" focused results and re-ordered 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.

We submitted two runs :

- $LIG - VSM - RIC - 4OUT - 05 - 00$: a run similar to the LIG-VSM-FOCRIC-4OUT-05-00, except that the doxels are ordered by their apparition in each document. In this run, we set $\lambda = 0.5$ and $\beta = 0.0$ and four neighbours;
- $LIG - LM - RIC - 4OUT - 05 - 00$: a run similar to the LIG-LM-FOCRIC-4OUT-05-00, except that the doxels are ordered by their apparition in each document. In this run, we set $\lambda = 0.5$ and $\beta = 0.0$ and four neighbours.

For the relevant in context task, our results in terms of non-interpolated generalized precision at early ranks $gP[r], r \in \{5, 10, 25, 50\}$ and non-interpolated Mean Average Generalized Precision $MAgP$ are presented in Table 2, and the interpolated Recall/Precision curve is presented in Figure 2. In these results, we see that the use of language model document ranking for doxels retrieval always outperforms the use of vector space based (+55% for MAgP). Similarly to our Focused runs, our results for the Retrieval in Context drop sharply when recall values increase, but the precision is above 62% for a recall of 0, which means that our best approach gives accurate results for the first query results.

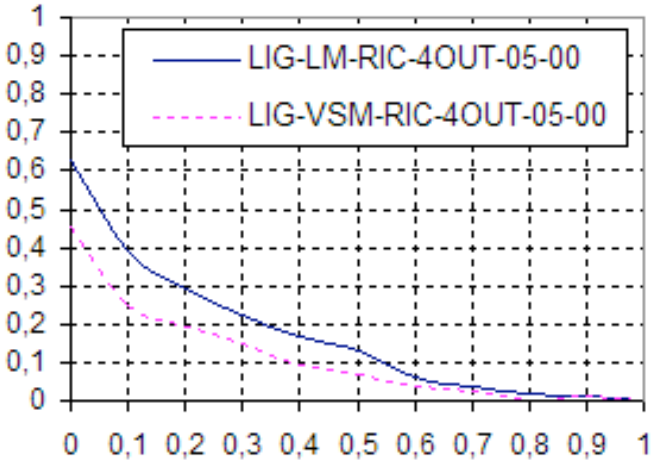
Table 2. Relevant In Context Task for INEX2007 Ad Hoc.

Run	gP[5]	gP[10]	gP[25]	gP[50]
$LIG - VSM - RIC - 4OUT - 05 - 00$ $MAgP = 0.0961$	0.2444	0.2023	0.1756	0.1360
$LIG - LM - RIC - 4OUT - 05 - 00$ $MAgP = 0.1486$	0.3595	0.3069	0.2303	0.1708

7 Summary and Conclusion

In the INEX 2008 Ad Hoc track, we intergrated two contexts (the four most similar doxels in a collectionlink document target and the full document rank) with the RSV of doxels. We submitted runs implementing our proposals for the Focused and Relevant in Context Ad Hoc tasks. For each of these tasks, we showed that combining content and context leads to good results, especially when considering full document language models. One explaination is that such models are well adapted for full documents, and using the context of doxel in a second step is a good approach. For our second participation to INEX, our best runs are ranked in the top 10 runs of participants systems at least in the Focused task at iP[0.01]. However, we plan to improve our baseline to obtain better results in the following directions:

Fig. 2. Interpolated Recall/Precision - Relevant in Context Task LIG Ad Hoc



– In the current results, we used a vector space model for the doxels and a language model for the full documents. We will focuse in the future on proposing a more consistent approach that relies only on language models.
– Refining also the non structural context of the doxel has to be done, by studying of other features that can be used to characterize the inter doxel relationships.

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. F. Huang, S. Watt, D. Harper, and M. Clark. Robert Gordon University at INEX 2006: Adhoc Track. In *INEX 2006 Workshop Pre-Proceeding*, pages 70–79, 2006.
3. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.
4. B. Piwowarski and M. Lalmas. Interface pour l'evaluation de systemes de recherche sur des documents XML. In *Premiere COnference en Recherche d'Information et Applications (CORIA'04)*, Toulouse, France, march 2004. Hermes.
5. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval - Chapter 6, page 203*. McGraw-Hill, Inc., New York, NY, USA, 1986.
6. J. Savoy. An extended vector-processing scheme for searching information in hypertext systems. *Inf. Process. Manage.*, 32(2):155–170, 1996.
7. M. D. Smucker and J. Allan. Using similarity links as shortcuts to relevant web pages. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 863–864, New York, NY, USA, 2007. ACM Press.

8. C. van Rijsbergen. *Information retrieval, Second edition - Chapter 3*. Butterworths, 1979.
9. D. Verbyst and P. Mulhem. Lig at inex 2007 ad hoc track : Using collectionlinks as context. In *Focused Access to XML Documents*, pages 138–147, Berlin/Heidelberg, Germany, 2007. Springer.
10. D. Verbyst and P. Mulhem. Doxels in context for retrieval: from structure to neighbours. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, New York, NY, USA, 2008. ACM.

University of Frankfurt at INEX2008 – An Approach for Distributed XML-Retrieval

Judith Winter[1] and Oswald Drobnik[1]

[1] J.W.Goethe University, Institute for Informatics, Frankfurt, Germany
{winter, drobnik}@tm.informatik.uni-frankfurt.de

Abstract. INEX provides a platform for a variety of successful and innovative XML-Retrieval approaches some of which are quite effective. However, all current solutions are centralized search engines. They do not consider distributed scenarios where it is not wanted or not possible to store the whole collection on one single machine. Such scenarios include search in large-scale collections, where the load of computations and storage consumption exceed the capacity of a single machine. Other systems consist of heterogeneous collections owned by different document providers. Those owners might be willing to share their collections but they may not want to give up full control by uploading them on a central server. Access control, privacy, and security are further reasons to distribute information among P2P systems, e.g. to avoid attacks or censorship.
Search in distributed systems has been a research area for the last few years – in regard of text documents or multimedia retrieval of images and music. With INEX and the progress of techniques to take advantage of XML-structure, it is now time to investigate distributed XML-Retrieval, too!
In this paper, we present a distributed search engine for XML-documents that is based on a Peer-to-Peer network. Our motivation for participating in INEX is to establish our system as a search engine that provides a variety of XML-Retrieval features leading to IR quality comparable to those of centralized XML-Retrieval systems. Furthermore, we would like to propose to the INEX community the idea of extending XML-Retrieval to distributed settings.

Keywords: Information Retrieval, Peer-to-Peer, XML Information Retrieval, XML-Retrieval, Distributed Search, Distributed XML-Retrieval, INEX

1 Introduction

Systems participating in INEX include a variety of XML-Retrieval approaches quite effective in terms of retrieval quality. To the best of our knowledge all current solutions are centralized search engines, though. This is sufficient for the current INEX test collection, consisting of 659.338 Wikipedia articles in XML format for the main ad-hoc track. In practice, collections such as those stored in digital libraries are much more voluminous and do not fit on one computer. Also, search should not be limited to one collection only, but a search engine should have access to a wide variety of

collections. This can include those offered by public institutions, such as universities, or the private collections of willing to share them.

Thus, distributed XML-Retrieval systems should be investigated in order to search large-scale or federated collections. Additionally, more powerful search engines can be built by pooling computers together – the load of computations and storage capacity can be shared by all participating nodes.

However, the cooperation of nodes implies communication costs which cannot be neglected. Distributed search engines hence have to consider efficiency as well as effectiveness. In terms of XML-Retrieval, a distributed search engine has to take care of an appropriate distribution of content information as well as structural information. On the other hand, structure can be used to perform distributed search more effectively and more efficiently, e.g. when selecting postings during routing.

To the best of our knowledge, none of the content-oriented XML-Retrieval solutions so far considers distributed aspects. In this paper, we present the first distributed XML-Retrieval system that performs content-oriented search for XML-documents. Our system is based on a structured Peer-to-Peer (P2) system.

P2P networks are emerging infrastructures for distributed computing where autonomous peers are pooled together. Other than in classical client/server systems, there is no central control but a high degree of self-organization, such that P2P systems are very flexible, adapt to new situations (such as joining and leaving of nodes), and thus may scale to theoretically unlimited numbers of participating nodes. A P2P network can be organized as a structured overlay network where a logical structure is laid on top of a physical network. Distributed hash tables (DHT) can be used to route messages to a peer storing an object specified by a unique identifier (ID), without knowing the peer's physical address [6]. The data structure for the routing table, and thus the logical structure of the overlay network, depends on the DHT algorithm used. For example, the Chord protocol maintains a DHT ring and maps peers and keys to an identifier ring [8]. In general, DHT-based algorithms structure an overlay network such that efficient lookup of objects can be achieved.

Among the growing amount of objects shared by P2P applications there is an increasing number of XML-documents, especially since public and private institutions, such as museums, start to share their Digital Libraries. There are a number of search engines for P2P networks, for example the DHT-based systems [2] and [5]. However, none of these approaches supports XML-Retrieval techniques. Schema-based P2P-networks [3], on the other hand, consider structure for the routing but existing solutions (such as [1]) do not provide IR techniques for content-oriented search.

In this paper, we propose a distributed XML-Retrieval system. It provides features such as allowing CAS queries, weighting different XML-elements differently, and element retrieval. On the one hand, this system takes advantage of XML-structure to perform the search for XML-documents more precise, i.e. structure is used to achieve a system more effective than current P2P-IR solutions. On the other hand, XML-structure can be involved in the routing to make the search process more efficient, e.g. by selecting posting list and postings based on evidence from both content and structure. To demonstrate the proposed approach as an XML-Retrieval system, we participated in INEX 2008. This paper describes motivation, research questions, technical details, submitted runs for three tasks of the ad-hoc track, and preliminary evaluation results of our system.

2 Spirix – a P2P Search Engine for XML-Documents

The following section presents Spirix, a search engine for P2P Information Retrieval (IR) of XML-documents. First, we present the architecture and system design of Spirix and especially focus on the techniques to extract and store structured information during indexing. Second, we outline where and how structure can be used in the process of answering a given INEX query.

2.1 System Design – How XML-Structure Is Distributed during Indexing

The smallest object indexed by Spirix is what we denote by XTerm: a tuple consisting of a stemmed non-stopword term and its structure. A term's structure is denoted as the path from the document root element to the term element in the XML-document tree expressed with XPath but without element numbers. To reduce the variety of different structures, we apply methods such as mapping of syntactically different but semantically similar tags, as well as stopword removal and stemming of tags.

For each indexed collection, a temporary inverted index for all extracted XTerms is build locally, where the vocabulary consists of keys and where a key combines all XTerms with the same content but each XTerm keeps its own posting list (see figure 1). Each posting refers to a document but its score is computed based on evidence from the document's weigth (tf, idf), from its element weights, and from the peer assigned to the document (peerScore). Details about this impact ordering are described in [9].

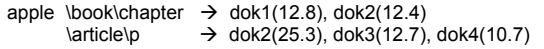


Fig. 1. Example of a key in the inverted index consisting of two XTerms.

As part of this parsing phase, we build a temporary statistic index (document index and element index) by collecting the statistics for each document and for selected elements which are treated as independent documents. So far, only fixed tags (paragraphs and sections) are considered as potential relevant elements. We are currently extending the method to elements that have been found relevant in the past or were specially marked by the indexing user.

Finally, all temporary indexes are distributed over the network. For message transport, we integrated a P2P protocol named SpirixDHT that has been developed to support efficient transport of messages between peers when retrieving XML-documents. This protocol is based on Chord and adapted to special requirements of XML-Retrieval as described in [9]. That is, in a network of n peers, we can store and locate information identified by unique IDs with DHT techniques in $log(n)$ hops. The inverted index is distributed by storing the posting lists on different peers. As ID that is hashed to assign a posting list to its according peer, we use the posting list key's content. Therefore, all posting lists regarding to the same content are stored on the same peer because during routing they are most likely to be accessed together. The statistic index is distributed by hashing the unique ID of each document (which is calculated by the peer's IP-address, its port, and the local document name, or which is created by storing the exact number of global documents in the DHT). To ensure that element

statistics are stored on the same peer as their root document, the hashed IDs of elements are chosen to be identical with those of their root documents.

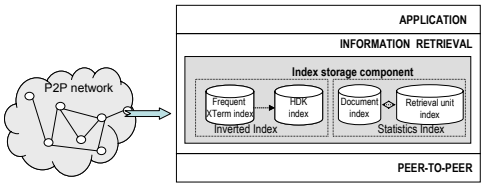


Fig. 2. Parts of the distributed global indexes that are stored on each peer.

By applying these distribution techniques, Spirix is based on global indexes (inverted index and statistics index) which are not stored centrally but distributed among all participating peers. If peers leave the network or new participants join, the indexes are automatically redistributed as the network organizes itself. For example, a new peer can take over some of the stored information that its direct neighbour has been responsible for. Figure 2 shows the different parts of the global indexes that are stored locally on each peer and that are managed by the index storage component as part of the IR complex on each peer.

2.2 System Use – Where XML-Structure Is Used during Querying

How is the distributed information used in the querying process, and especially where is XML-structure applied to improve performance?

Retrieval includes routing and ranking, and consists of two steps: the first step serves to locate the posting lists of all query terms, to merge them, and to select promising postings. Second, locate the statistics for the selected postings such that the referenced documents and elements can be ranked. In contrast to common P2P search engines that perform routing and ranking solely based on content evidence, Spirix includes XML-structure in both steps.

Routing: Especially for multi term queries, where posting lists have to be sent between peers for merging, not all postings can be selected because for large-scale collections with big posting lists this would significantly increase network traffic. For the selection of posting lists, we compute the similarity between the structural hint of a CAS query and the structure of an XTerm's posting list. Only those posting lists are considered where the structural similarity exceeds a threshold. Furthermore, only top k entries from the chosen posting lists are selected. For this purpose, the postings are ordered by impact based on evidence from document-, element-, and peer-level but also based on the structural similarity of the original posting list. Hence, postings of XTerms that match the CAS hint closely get a higher impact factor than less similar ones.

Ranking: For each selected posting, a ranking request is sent to the peer that is assigned to the referenced document. The relevance is then computed using an extension of the vector space model: query, document, and elements are represented as vectors where each component contains the weight of an XTerm. That is, the weight of a term is split into weights for its different structures. Second, the weight itself is

computed using structure: it is computed as the product of the XTerm's content's weight and the structural similarity between XTerm and CAS query term. Third, the XTerm's content's weight is computed with an adaption of BM25E [7] such that different elements can be weighted differently, e.g. "titles" can get higher weights than "links". Last, all relevance computations are performed not only for the selected documents but also for their potential relevant elements (whose statistics are stored on the same peer for better access) to achieve more focused and specific results. Similar to related work, e.g. [4], we assume a strong correlation between the relevance of an element and its parent document. Therefore, the parent document influences the ranking of elements: the score of an element is computed based on the stored statistics and smoothed by the score of its parent document.

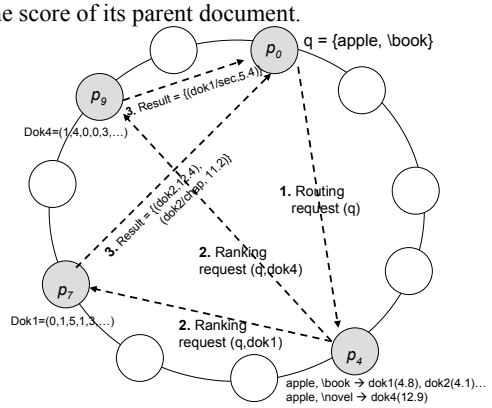


Fig. 3. Retrieval process for single term query *q = {apple,\book}* based on SpirixDHT.

Figure 3 displays the process of answering a query *q*. First, *q* is routed to peer p_4 which is assigned to the hash of *apple* and thus holds all posting lists for XTerms with content *apple*. On p_4, posting lists and postings are selected according to *q* by taking into account the postings' weights multiplied with the similarity to *\book*. Postings *dok1* and *dok4* are selected, and routing requests are sent to peers p_7 and p_9 which are responsible to hold the statistics of these documents and their elements. Peer p_7 and peer p_9 both receive the query, calculate results and send these back to the querying peer p_0.

3 Participating at INEX 2008

3.1 INEX Tracks

In 2008, University of Frankfurt participated in INEX for the first time. We chose two tracks: the ad-hoc track and the efficiency track. The participation at the efficiency track is described in [11]. In this chapter, we concentrate on the ad-hoc track.

3.2 INEX Tasks (Ad-hoc Track)

Our main task was the focused task, but we submitted the same results –filtered by the according rules – to the BestInContext task (BIC) and to the RelevantInContext task (RIC) as well to see how Spirix performs in these tasks.

Focused: To answer a topic, 5000 documents are chosen from the posting list of each query term. Note, that this is done on separate peers, i.e. before merging of posting lists. Their relevance plus the relevance of selected elements from these documents is then computed. Overlapping is filtered out by ordering the results by document ID and element offset. If two results have identical document IDs and one offset is the first part of the other one, only the result with the higher relevance is kept.

RelevantInContext (RIC): After removal of overlapping elements in the same manner as in the Focused task, the results are sorted into groups. Each group contains a document and all its relevant elements. The list of groups is sorted by root document relevance.

BestInContext (BIC): For this task, the overlap-filtering was modified such that all elements with identical document IDs were subject to the filter. Thus, only the best element of a document (or the document itself) survived the filter.

3.3 University of Frankfurt Runs (Ad-hoc Track)

We participated to demonstrate Spirix as an XML-Retrieval system. Furthermore, we were interested in a comparison of different methods implemented in our system. These include:

- CO versus CAS: do structural hints help to improve our ranking or routing?
- How do different ranking functions perform?
- How do different structural similarity functions perform (ranking and routing)?
- How does the amount of selected postings effect precision?
- Element versus document retrieval –do we perform better by retrieving elements?

For the official runs, we decided to compare different ranking functions, so we submitted the following runs:

Table 1. Runs submitted by University of Frankfurt.

Run	*Status*	*Task*	*Type*	*Name*
234	Evaluated	BIC	Element	006: BM25e_W1, 5000 postings, architect-similarity
235	Evaluated	BIC	Element	005: BM25e, 5000 postings, architect-similarity
236	Evaluated	BIC	Element	004: tf*idf-variant, 5000 postings, architect-similarity
242	Unchecked	Focused	Element	003: BM25E_W1, 5000 postings, architect-similarity
243	Unchecked	Focused	Element	002: BM25e, 5000 postings, architect-similarity
244	Evaluated	Focused	Element	001: tf*idf-variant, 5000 postings, architect-similarity
231	Evaluated	RIC	Element	007: tf*idf-variant, 5000 postings, architect-similarity
232	Evaluated	RIC	Element	008: BM25e, 5000 postings, architect-similarity
233	Evaluated	RIC	Element	009: BM25e_W1, 5000 postings, architect-similarity

For each run, 5000 postings were selected from the posting lists and the structural similarity function described in [10] was used. For each task, three runs were

submitted for three different ranking functions: a tf*idf-Baseline, a variant of BM25E, and a variant of BM25E with different weights for different elements.

3.4 Tuning the System – Balance between Effectiveness and Efficiency

As to the ad-hoc track, we aimed at high precision. Thus, for many parameters that influence the balance between effectiveness and efficiency, we chose values with respect to effectiveness. This includes for example global statistics. For early termination (selection of postings), we decided for a compromise of 5000 postings. Regarding storage of structural information, we used methods to shorten the structure length resulting in a much smaller and easier manageable index. However, this also reduces precision by approximately 5%.

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 eliminates 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.

Early termination (selection of 5000 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 (up to a collection specific point). Thus, the best 5000 documents were selected from each query term's posting list. Note, that this is done on separate peers and thus without merging – we loose precision for multi term queries when good documents are on positions > 5000.

4 Evaluation

For the focused task, we achieved interpolated precision at 1% (iP[0.01]) of 27% for our baseline run (tf*idf variant as ranking algorithm). Our systems thus came on rank 58 out of 61 participants in the preliminary evaluation.

Table 2. Performance at INEX2007 (unofficially) and INEX2008, focused task.

	INEX'07 (unofficially)	*INEX'08 (officially)*	*INEX'08 (re-evaluated)*
iP[0.01] Uni Frankfurt	37%	27%	52%
iP[0.01] best system	52% (Dalian)	68% (Waterloo)	68% (Waterloo)
Rang	57 out of 79	58 out of 61	ca. 47 out of 61

However, our runs with more sophisticated ranking methods (BM25E variants) were not considered in the official evaluation of INEX 2008 due to errors in the overlapping filter such that the runs had duplicates. After correcting this bug, we run the evaluation ourselves and got a precision of 51,97%. This would have meant rank 47 of 61 participants. Figure 4 shows precision for the officially evaluated run001 (Baseline) and for the re-evaluated run002 (BM25E-adaption) for the focused task.

Our precision for the BIC and RIC tasks was nearly 0, so we assume another error with our filtering methods as the result files themselves have been the same as submitted to the focused task. We are in the process of investigating this.

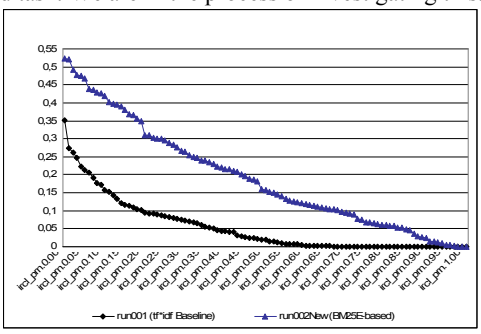


Fig. 4. Precision over all recall levels for run001 (baseline) and run002 (BM25E).

Figure 5 displays iP[0.01] and average interpolated precision (AiP) for run002 for each single topic. The variety over the topics is as wide as possible: the precision can be anything between 0% and 1%. We will have to investigate why for some topics Spirix did not return relevant documents or only few. This will be done in comparison to the results of other participants to identify easy and difficult topics.

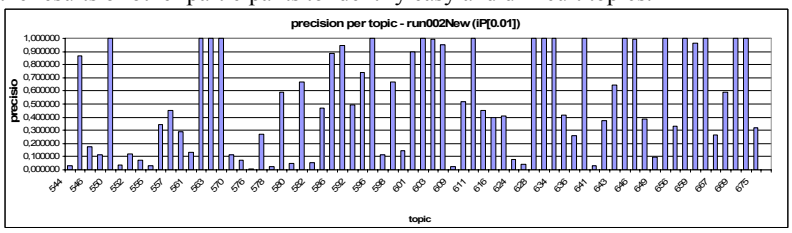


Fig. 5. iP[0.01] for each single topic (run002).

5 Conclusion

In this paper, the first distributed search engine Spirix has been proposed. It is based on a structured P2P system and offers a wide variety of XML-Retrieval features such as taking advantage of CAS queries, weighting different elements differently, and element retrieval. We participated in INEX 2008 to demonstrate Spirix as an XML-Retrieval system with an IR quality comparable to centralized XML-Retrieval. Due to the underlying architecture of a P2P network, we have to consider efficiency issues and are forced to make some short cuts which inevitably decreases precision. University of Frankfurt participated in INEX 2008 for the first time. Unfortunately, the implementation of our prototype has not been finished until one week before the run submission deadline. Thus, most of our runs failed due to bugs in our task filters.

The only officially evaluated run was our baseline run where we achieved a precision of 27%.

However, we have been able to fix the filters and did a re-evaluation after release of the new INEX 2008 tool. Our focused run now achieved iP[0.01] of 51,97 %, which could have been around rank 47 of 61 participants.

6 Acknowledgements

We would like to thank Andrew Trotman and Shlomo Geva very much for sharing their view of XML-IR, for countless constructive remarks concerning Spirix, and for endless hours of discussion about the challenges and opportunities of XML-P2PIR.

References

[1] Abiteboul, S.; Manolescu, I.; Polyzotis, N.; Preda, N.; Sun, C.: ***XML processing in DHT networks***. IEEE 24th Internat. Conference on Data Engineering (ICDE2008), Cancun, Mexico (2008)

[2] Bender, M.; Michel, S.; Weikum, G.; Zimmer, C.: ***The MINERVA Project - Database Selection in the Context of P2P Search***. BTW Conference 2005, Karlsruhe, Germany (2005)

[3] Koloniari, G.; Pitoura, E.: ***Peer-to-Peer Management of XML Data: Issues and Research Challenges***. SIGMOD Rec. Vol. 34, No. 2 (2005)

[4] Mass, Y.; Mandelbrod, M.: ***Component Ranking and Automatic Query Refinement for XML Retrieva***l. LNCS, Vol. 3493/2005, Springer-Verlag (2005)

[5] Podnar, I.; Rajman, M.; Luu, T.; Klemm, F.; Aberer, K.: ***Scalable Peer-to-Peer Web Retrieval with Highly Discriminative Keys***. In: Proc. of IEEE 23rd International Conference on Data Engineering (ICDE 2007), Istanbul, Turkey (2007)

[6] Risson, J.; Moors, T.: ***Survey of research towards robust peer-to-peer networks – search methods***. In: Technical Report UNSW-EE-P2P-1-1, Uni. of NSW, Australia (2004)

[7] Robertson, S.; Zaragoza, H.; Taylor, M.: ***Simple BM25 extension to multiple weighted fields***. In: Proc. of CIKM'04, ACM Press, New York, USA (2004)

[8] Stoica, I.; Morris, R.; Liben-Nowell, D.; Karger, D.; Kaashoek, F.; Dabek, F.; Balakrishnan, H.: ***Chord - A Scalable Peer-to-peer Lookup Protocol for Internet Applications***. IEEE/ACM Transactions on Networking, Vol. 11, No. 1 (2003)

[9] Winter, J.: ***Routing of Structured Queries in Large-Scale Distributed Systems.*** In: Workshop on Large-Scale Distributed Systems for Information Retrieval (LSDS_IR'08) at ACM CIKM 2008, Napa Valley, California, USA (2008)

[10] Winter, J.; Drobnik, O.: ***An Architecture for XML Information Retrieval in a Peer-to-Peer Environment.*** ACM PIKM2007 at ACM 16th Conference on Information and Knowledge Management (CIKM 2007), Lisbon, Portugal (2007)

[11] Winter, J.; Jeliazkov, N.: ***Recognition of Structural Similarity To Increase Performance***. In: Preproceedings of the 7th International Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2008), Dagstuhl, Germany (2008)

Overview of the INEX 2008 Book Track

Gabriella Kazai[1], Antoine Doucet[2], and Monica Landoni[3]

[1] Microsoft Research Cambridge, United Kingdom
gabkaz@microsoft.com
[2] University of Caen, France
doucet@info.unicaen.fr
[3] University of Lugano
monica.landoni@unisi.ch

Abstract. This paper provides an overview of the INEX 2008 Book Track, its current progress, participants, tasks, book corpus, test topics and relevance assessment collection system, as well as some results.

1 Introduction

Through mass-digitization projects and with the use of OCR technologies, the full texts of digitized books are becoming available on the Web and in digital libraries [1]. Examples include the Million Book project[4], the efforts of the Open Content Alliance[5] and the digitization project of Google[6].

The unprecedented scale of these efforts, the unique characteristics of the digitized material, as well as the unexplored possibilities of user interactions make full-text book search and eBook design and development an exciting area of research today. The growing interest in this area is reflected by the wide range of challenges and opportunities highlighted at the BooksOnline'08 Workshop [2].

Motivated by the need to foster research in this domain, the book track was launched in 2007 as part of the INEX initiative. INEX was chosen as a suitable forum as searching for information in a collection of books can be seen as one of the natural application areas of structured document retrieval (SDR), which has been investigated at INEX since 2002. For example, in focused retrieval a clear benefit to users is to gain direct access to parts of books (of potentially hundreds of pages in length) that are relevant to their information need.

The overall goal of the INEX Book Track is to study book-specific relevance ranking strategies for informational search tasks, investigate user interface issues and user behaviour both for online and e-reader based reading and information seeking experiences, exploiting book-specific features, such as back of book indexes and table of contents provided by authors, and associated metadata like library catalogue information.

In 2007, the track concentrated on identifying infrastructure issues, focusing on information retrieval tasks.

[4] http://www.ulib.org/
[5] www.opencontentalliance.org/
[6] http://books.google.com/

In 2008, the aim was to look beyond and bring together researchers and practitioners in Digital Libraries, Information Retrieval (IR), Human Computer Interaction, and eBooks to explore common challenges and opportunities around digitized book collections. Toward this goal, the track set up tasks to provide opportunities for investigating research questions around the following broad topics:

- IR techniques for searching full texts of digitized books, which are investigated in the Book Retrieval and Page in Context tasks,
- User's interactions with eBooks and collections of digitized books, which is the area that the Active Reading task is aimed at exploring,
- Mechanisms to increase accessibility to the contents of digitized books, which is the focus of the Structure Extraction task.

This paper provides an overview of the track, its current progress and results so far. Since, at the time of writing, the Active Reading task was still running and the relevance assessments for both of the search tasks (Book Retrieval and Page in Context) were in the process of being collected, the evaluation results for these are not reported here.

The paper is structured as follows. Section 2 gives a brief summary of the participating organisations. In Section 3, we describe the corpus of books that forms the main part of the INEX book test collection. The following three sections detail the four tasks defined in the track: Section 4 summarises the Book Retrieval and Page in Context retrieval tasks, Section 5 reviews the Structure Extraction task, and Section 6 discusses the Active Reading task. We close in Section 7 with a summary and further plans.

2 Participating organisations

A total of 54 organisations registered for the track (double of last year's 27), see Table 1. However, only 13 of them have actually taken part actively throughout the year (up from 9 last year). For active participants, the number of topics and runs they submitted are also listed. In total, 19 groups downloaded the book corpus, 11 groups contributed search topics, 2 have submitted runs to the Structure Extraction task (Microsoft Development Center Serbia, and Xerox Research Centre Europe), 4 to the Book Retrieval task (University of Amsterdam, University of California Berkeley, RMIT University, and University of Waterloo), 2 to the Page in Context task (University of Amsterdam, and University of Waterloo), and 2 are participating in the Active Reading task (University of California Berkeley, and Kyungpook National University).

3 Book corpus

The track builds on a collection of digitized books, provided by Microsoft Live Search and the Internet Archive (for non-commercial purposes only). It consists

ID	Organisation	ID	Organisation
Active participants			
6	University of Amsterdam Corpus; Topics: 51, 52, 65; Runs: 3 BR, 7 PiC	54	Microsoft Research Cambridge Corpus; Topics: 55, 56, 57, 58, 62, 63, 64, 70
14	University of California, Berkeley Corpus; Topics: 66, 67; Runs: 3 BR, ART	56	JustSystems Corporation Corpus; Topics: 53, 54, 59
30	CSIR, Wuhan University Corpus; Topics: 36, 38, 39, 42	62	RMIT University Corpus; Topics: 31, 37, 41, 43; Runs: 10 BR
31	Faculties of Management and Information Technologies, Skopje Corpus; Topics: 40, 46, 47, 48	78	University of Waterloo Corpus; Topics: 32, 33, 34, 35; Runs: 2 BR, 6 PiC
41	University of Caen Corpus; Topics: 60, 61	86	University of Lugano Corpus; Topics: 68, 69
43	Xerox Research Centre Europe Corpus; Runs: 4 SE	125	Microsoft Development Center Serbia Corpus; Runs: 3 SE
52	Kyungpook National University Corpus; Topics: 44, 45, 49, 50; ART		
Passive participants (Corpus download)			
4	University of Otago	17	University of Strathclyde
7	Oslo University College	42	University of Toronto
10	Max-Planck-Institut Informatik	116	University of the Aegean
Passive participants			
5	Queensland University of Technology	104	UCLV
8	University College London	107	University of Sci. and Tech. of China
9	University of Helsinki	112	Hitachi, Ltd.
15	Univeristy of Iowa	115	IIT
19	University of Ca Foscari di Venezia	117	Iran
21	MPP	118	M.Tech Student
27, 76	Univeristy at Albany	127	UNICAMP
29	Indian Statistical Institute	148	UEA
32	CUHK	158	George Mason University
39	University of New South Wales	160	Universite Jean Monnet
51	Suny-Albany	161	University of California, Santa Cruz
60	Saint Etienne University	164	Isfahan University
66	University of Rostock	165	Universidad de Oriente
88	Independent	166	Drexel University
91	Auckland University of Technology	171	Chinese University of Hong Kong
93	Wuhan Institute of Technology	174	Alexandria University
96	Cairo Microsoft Innovation Center	181	COLTEC
100	Seoul National University		
Total of 54 organisations			

Table 1. Participating groups at the Book Track in 2008 (BR = Book Retrieval, PiC = Page in Context, SE = Structure Extraction, ART = Active Reading Task)

of over 50,000 digitized out-of-copyright books from a wide range of genres, including reference works as well as poetry. Most of the corpus is made up of history books (mostly American history), biographies, literary studies, religious texts and teachings. There are also encyclopedias, essays, proceedings and novels.

The OCR content of the books has been converted from the original DjVu format to an XML format referred to as BookML, developed by the Document Layout Team of Microsoft Development Center Serbia. BookML provides richer structure information, including markup for table of contents entries. Most books also have an associated metadata file (*.mrc), which contains publication (author, title, etc.) and classification information in MAchine-Readable Cataloging (MARC) record format.

The basic XML structure of a typical book in BookML (ocrml.xml) is a sequence of pages containing nested structures of regions, sections, lines, and words:

```
<document>
 <page pageNumber=‘‘I-N’’ label=‘‘PT_CHAPTER’’ [...]>
  <region regionType=‘‘text’’ [...]>
   <section label=SEC_BODY’’ [...]>
    <line [...]>
     <word val=‘‘Moby’’ [...]/>
     <word val=‘‘Dick’’ [...]/>
    </line>
    <line [...]>
     <word val=‘‘Herman’’ [...]/>
     <word val=‘‘Melville’’ [...]/>
    </line>     [...]
   </section>   [...]
  </region>     [...]
 </page>        [...]
</document>
```

BookML provides a rich set of labels indicating structure information and additional marker elements for more complex texts, such as a table of contents. For example, the label attribute of a section indicates the type of semantic unit that the text contained in the section is likely to be a part of, e.g., a table of contents (SEC_TOC), a header (SEC_HEADER), a footer (SEC_FOOTER), or the body of the page (SEC_BODY).

The corpus was distributed on USB HDDs (at a cost of 70GBP), and a reduced version was made available for download. The original corpus totals 400GB, while the reduced version is only 50GB (and 13GB compressed). The reduced version was created by removing the word tags and their attributes (coordinates, etc.) and propagating the values of the val attributes as content into the parent line elements.

4 Information Retrieval Tasks

Two retrieval tasks were run: 1) the Book Retrieval (BR) task and 2) the Page in Context (PiC) task. Both these tasks build on the full corpus of over 50,000 books described in Section 3. The test topics, which were created by the participants, are also shared across the two tasks. This was motivated by the need to reduce the relevance assessment workload and to allow possible comparisons across the two tasks.

A summary of the two tasks, the test topics, and the online relevance assessment system are described in the following sections. Further details and the various submission run DTDs are available online in the track's Tasks and Submission Guidelines[7].

4.1 Book Retrieval 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 retrieved. 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. The list may be for entertainment, for research purposes, or in preparation of lecture materials, etc. Books on a reading list may be purchased or borrowed from libraries.

Participants of this task were invited to submit either single runs or pairs of runs. A total of 10 runs could be submitted. A single run may either be the result of generic (non-specific) or book-specific IR methods. A pair of runs had to contain both types, where the non-specific run serves as a baseline on which the book-specific run extends on by exploiting additional book-specific features (e.g., back-of-book index, citation statistics, book reviews, etc.) or specifically tuned methods.

At minimum, one automatic run (i.e., using only the topic title part of a test topic for searching and without any human intervention) was compulsory. Each run could contain, for each test topic, the top 1000 books (identified by its 16 character long bookId[8]) estimated relevant to the given topic, ranked in order of estimated relevance.

A total of 18 runs were submitted by 4 groups (University of Amsterdam (3); University of California, Berkeley (3); RMIT University (10); and University of Waterloo(2)), see Table 1.

4.2 Page in Context Task

The goal of this task is to investigate the application of focused retrieval approaches to a collection of digitized books. The task is thus similar to the INEX

[7] Available at http://www.inex.otago.ac.nz/tracks/books/taskresultspec.asp

[8] The bookId is the name of the directory that the book files are stored in within the corpus, e.g., A1CD363253B0F403

ad hoc track's Relevant in Context task, but using a significantly different collection and allowing for the ranking of book parts within a book.

The user scenario underlying this task is 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 of the book) or it may be the main focus of some other books. The user expects to be pointed directly to the relevant book parts. Following the focused retrieval paradigm, the task of a 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 books. Books needed to be ranked by decreasing order of relevance, which may be based on best or average passage/element score or some other document score. The book parts within a book were to be ranked in decreasing order of relevance. Both passage and element retrieval approaches were allowed.

As in the BR task, participants could submit up to 10 runs, where one automatic and one manual runs were compulsory. Each run could contain, for each topic, a maximum of 1000 books estimated relevant to the given topic, 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 cannot be mixed.

A total of 13 runs were submitted by 2 groups (University of Amsterdam (7); and University of Waterloo(6)), see Table 1. All PiC runs contained XML element results (i.e., no passage based submissions were received).

4.3 Test topics

The test topics are representations of users' informational needs, i.e, where the user is assumed to search for information on a given subject. As last year, all topics were limited to deal with content only aspects (i.e., no structural query conditions).

Participants were asked to create and submit topics for which at least 2 but no more than 20 relevant books were found during the collection exploration stage. To aid participants in their collection exploration task, they were provided with an online search system, called the Book Search System[9] developed at Microsoft Research Cambridge, which allowed them to search, browse and read the books in the test corpus.

A total of 40 new topics (topics 31-70) were contributed by 11 participating groups (see Table 1) in 2008, following the topic format described below. These were then merged with the 30 topics created last year (topics 1-30) and used as the test set for generating the retrieval runs. An example topic from the 2008 test collection is given in Figure 1.

[9] http://www.booksearch.org.uk

Topic Format. The topic format remained unchanged from 2007, each topic consisting of three parts, describing the same information need, but for different purposes and at different level of detail:

<title> represents the search query that is to be used by systems for the automatic runs. It serves as a short summary of the content of the user's information need.
<description> is a natural language definition of the information need.
<narrative> is a detailed and unambiguous explanation of the information need and a description of what makes an element/passage relevant or irrelevant. The narrative is taken as the only true and accurate interpretation of the user need. It consists of the following two parts:
<task> is a description of the task for which information is sought, specifying the context, background and motivation for the information need.
<infneed> is a detailed explanation of what information is sought and what is considered relevant or irrelevant.

```
<?xml version=‘‘1.0’’ encoding=‘‘ISO-8859-1’’?>
<!DOCTYPE inex_topic SYSTEM ‘‘bs-topic.dtd’’>
<inex_topic track=‘‘book’’ task=‘‘book-retrieval/book-ad-hoc’’
topic_id=‘‘62’’  ct_no=‘‘2008-37’’>
 <title> Attila the hun </title>
 <description> I want to learn about Attila the Hun’s character, his way of
     living and leading his men, his conquests, and rule.
 </description>
 <narrative>
  <task> I was discussing with some friends about Attila the Hun. What I
     found interesting was the difference in our perceptions of Attila: As a
     great hospitable king vs. a fearsome barbarian. I want to find out more
     about Attila’s character, his way of living as well as about his wars to
     better understand what he and his era of ruling represents to different
     nations.
  </task>
  <infneed> Any information on Attila’s character, his treatment of others, his
     life, his family, his people’s and enemies’ view on him, his ambitions,
     battles, and in general information on his ruling is relevant, and so is any
     information that can shed light on how he is perceived by different nations.
     Poems that paint a picture of Attila, his court and his wars are also relevant.
  </infneed>
 </narrative>
</inex_topic>
```

Fig. 1. Example topic from the 2008 test set.

4.4 Relevance Assessments

The Book Search System, developed at Microsoft Research Cambridge, is an online web service that allows participants to search and browse the books in the INEX 2008 Book Track corpus. It is available publicly at http://www.booksearch.org.uk.

The Book Search System also provides a complete relevance assessment module, which allows users to annotate books and pages inside the books, adding for example relevance labels. The relevance assessment system has been made available publicly as part of an online competition called the Book Explorers' Competition, where anyone interested may register and compete for prizes sponsored by Microsoft Research. The competition involves reading and marking relevant content inside books for which users are rewarded points. The competition is run between Dec 1-15, 2008, and is hoped to create sufficient relevance judgements to allow the evaluation of the submitted Book Retrieval and Page in Context runs.

Screenshots of the assessment system are shown in Figures 2 and 3. Figure 2 shows the list of books (assessment pool) to be judged for a given topic (selected by the user). The list was built by pooling the submitted runs (using a round robin process) and merging additional search results from the Book Search System itself. On accessing a book, the book is opened in the Book Viewer window (Figure 3). There, users can browse through the book and search inside it, or go through the pages listed in the Assessment tab, which were extracted from the Page in Context runs. Users can highlight text fragments on a page by drawing a highlight-box over the page image. They can also mark a whole page or a range of pages as relevant/irrelevant. Users are also asked to rate the relevance of the whole book. A detailed user manual and system description is available at http://www.booksearch.org.uk/BECRulesAndUserManual.pdf.

5 Structure Extraction Task

The goal of this task is to test and compare automatic techniques for extracting structural information from digitized books and building a hyperlinked table of contents (ToCs).

The task was motivated by the limitations of current digitization and OCR technologies to produce the full text of digitized books with only minimal structure information markup: Pages and paragraphs are usually identified, but more sophisticated structures, such as chapters, sections, etc., are currently not recognised.

Participants of the task were provided a sample collection of 100 digitized books 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 given. The books were selected to form a representative sample of different genre and styles. In addition to the DjVu XML files, participants were distributed the PDF of books or their JPEG image files (one per page).

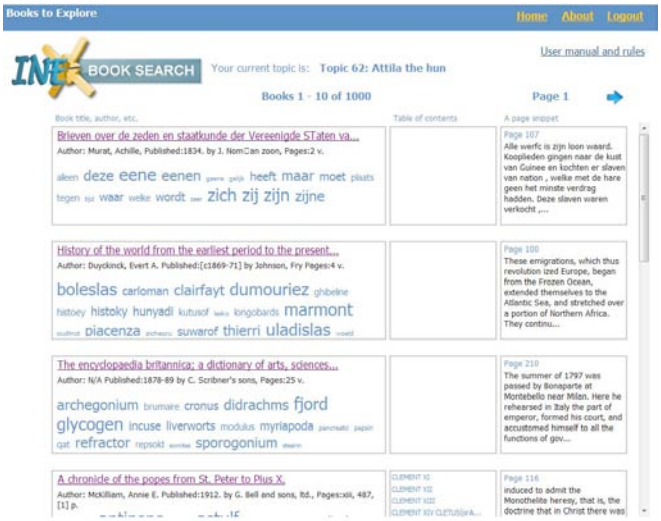


Fig. 2. Screenshot of the relevance assessment module of the Book Search System: List of books in the assessment pool for a selected topic.

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

A total of 7 runs were submitted from 2 institutions (Microsoft Development Center Serbia (MDCS), and Xerox Research Centre Europe (XRCE)). MDCS extracted ToCs by first recognizing the page(s) of a book that contained the printed ToCs, whereas the XRCE relied on title detection within the body of a book. Please refer to the corresponding papers in the proceedings for further details.

5.1 Evaluation and Results

The ToCs generated by participants were compared to a manually built ground truth, and evaluated using recall/precision like measures at different structural levels (i.e., different depths in the ToC). To collect ground truth data, a structure labeling tool was built which allowed assessors to attach labels to entries and parts of entries in the printed ToC of a book (using the PDF file). The tool was built by the Microsoft Development Center Serbia team and assessors were recruited and paid for their labeling work.

For the evaluation, precision and recall were defined as follows: Precision is the ratio of the total number of correctly recognized ToC entries and the total number of ToC entries; Recall is 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 is then 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.

RunID	Participant	F-measure (complete entries)
MDCS	MDCS	54,29%
MDCS_NAMES_AND_TITLES	MDCS	53,41%
MDCS_TITLES_ONLY	MDCS	24,06%
HF_ToC_prg_Jaccard	XRCE	9,62%
HF_ToC_prg_OCR	XRCE	9,47%
HF_TPF_ToC_prg_Jaccard	XRCE	9,46%
HF_ToC_lin_Jaccard	XRCE	5,33%

Table 2. Summary of performance scores for the Structure Extraction task

6 Active Reading Task

The main aim of the Active Reading Task (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 [3], 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 Requirements

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 who would 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 (see below) is provided to participants as a tool to define the user tasks they would 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.

ART crib sheet. A task crib sheet is a rich description of a user task that forms the basis of a given user study based on a particular scenario in a given context.

Thus, it aims to provide a detailed explanation of the context and motivation of the task, and all details that form the scenario of use. It should include a detailed explanation of how the task should be successfully performed, possible paths to solutions and expected outcomes. Information recorded in the task crib sheet must be clear and precise in order to unambiguously determine whether or not users have completed a task and expected results have been achieved. Precise recording of the task is also important for scientific repeatability. The task crib sheet, hence, had the following parts:

- Objectives: A summary of the aims and objectives of the task from the users' point of view, i.e. what is it that users are trying to achieve in this task. It is expected that a variety of tasks of different complexity will be produced by each participating group.
- Task: Description of the task
- Motivation: Description of the reasons behind running the task
- Context: Description of the context of the task in terms of time and resources available, emphasis and any other additional factors that are going to influence task performance
- Background: Describes any background knowledge required to accomplish the task
- Completion: Describes how to assess whether the task has been completed or not
- Success: Describes whether the task has been completed successfully.

6.2 User Studies

Participants are expected to run each of their proposed user studies with at least 15/20 test subjects and submit their findings, including satisfaction questionnaires provided by the organizers (which may be freely extended by participants) and completed by each subject in the study.

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 encourage direct collaboration with participants to help shape the tasks according to real/existing research needs. In fact one of the participants explained how English written material was not much use for their experiments as they were targeting Korean speaking users, so it was agreed that they would use their own book collection while still adopting the ART evaluation framework to ensure results were comparable at the end.

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.

Since ART is still ongoing, there is no data to be presented at this point.

7 Conclusions and plans

The Book Track this year has attracted a lot of interest and has grown to double the number of participants from 2007. However, active participation remained a

challenge for most participants due to the high initial set up costs (e.g., building infrastructure). Most tasks also require quite a lot of advance planning and preparations, e.g., for setting up a user study. This, combined with the late announcement and advertising of some of the tasks has greatly limited active participation this year. In particular, we received lots of expressions of interest for the Structure Extraction and the Active Reading tasks, but the deadlines prohibited most people from taking part. We aim to address this issue in INEX 2009 by raising awareness early on in the start of the INEX year and by ensuring continuity with the tasks established this year.

As a first step in this direction, we are proposing to run the Structure Extraction task both at INEX 2009 and at ICDAR 2009 (International Conference on Document Analysis and Recognition). This means an earlier start to the task (January 2009) and an earlier completion (July 2009 or earlier).

Both the Book Retrieval and Page in Context tasks will be run again in 2009, although we only expect to complete this year's cycle by the early part of 2009. The greatest challenge in running these tasks has been the collection of relevance assessments. Due to the huge effort required, we decided to depart from the traditional method of relevance assessment gathering (i.e., one judge per topic), and designed a system that aims to collect judgements from a distributed and diverse set of users, where multiple judges can assess the same topic. Implemented as an online game, users can contribute relevance labels for books, pages and passages for any topic they may be interested in and for any number of books for that topic. This way of collecting judgements is aimed to provide a more realistic expectation on the assessors, but it also comes with its own risks. Attracting a sufficiently large and dedicated user base is one of the risks, for example. Other risks include the question of the quality of the collected labels due to a mixture of expert and non-expert judges. To address the latter, we introduced a number of measures, such as requiring users to specify their familiarity with a given topic they selected to provide judgements on, as well as allowing users to quality check each other's work. The first pilot test of this methodology will complete on December 15, which will give us a better understanding of its feasibility.

We will either postpone this year's Active Reading task to 2009 or run it again. We found that the introduction of the ART task was a challenge for number of reasons:

- Because of its original approach to evaluation, which is quite far away from the classic TREC paradigm, and the relative difficulty in framing ART in a formal way, the task organisation has suffered few delays that have affected the availability of participants to get fully involved in it;
- User studies are per se risky and unpredictable and the idea of running a number of those in parallel in order to compare and combine results added an extra layer of uncertainty to the task, somehow discouraging participants that were used to a more stochastic approach to evaluation;
- The formalisation of the procedure and protocols to be followed when running user studies was designed on purpose to be flexible and unconstructive in order to accommodate for participants' specific research needs. This flex-

ibility, however, was interpreted by some as a negative feature, a lack in details that discouraged them from taking part.
- Opening up to different communities that were not yet involved in INEX required concentrated effort in order to advertise and raise awareness of what INEX's aims and objectives and in particular what ART's goals were. Some of this effort was simply to late for some interested parties.

The organisation of the ART task has proved a very valuable experience that has given us the opportunity to explore different research perspective while focusing on some of the practical aspects of the task. We believe that the effort that has gone into setting up and advertising ART this year will be rewarded by a more successful task next year. It has been a very enlightening experience and has given us a better understanding of how much participants' expectations could contribute to success of such an initiative.

Acknowledgements

The Book Track in 2008 was supported by the Document Layout Team of Microsoft Development Center Serbia. They contributed to the track by providing the BookML format and a tool to convert books from the original OCR DjVu files to BookML, which contains richer structural markup. 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. 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.
3. R. Wilson, M. Landoni, and F. Gibb. The web experiments in electronic textbook design. *Journal of Documentation*, 59(4):454–477, 2003.

Book Layout Analysis: TOC Structure Extraction Engine

Bodin Dresevic, Aleksandar Uzelac, Bogdan Radakovic, and Nikola Todic

Microsoft Development Center Serbia,
Makedonska 30, 11000 Belgrade, Serbia
{bodind,aleksandar.uzelac,bogdan.radakovic,nikola.todic}@microsoft.com
http://www.microsoft.com/scg/mdcs

Abstract. Scanned then OCRed documents usually lack detailed layout and structural information. We present a book specific layout analysis system used to extract TOC structure information from the scanned and OCRed books. This system was used for navigation purposes by live books search project. We provide labeling scheme for the TOC sections of the books, high level overview for the book layout analysis system, as well as TOC structure extraction engine. In the end we present accuracy measurements of this system on a representative test set.

Key words: book layout analysis, TOC, information extraction, TOC navigation, ocrml, bookml

1 Introduction

Book layout analysis as described in this paper is a process of extracting structural information from scanned and OCRed books. Main purpose of this work was to enable navigation experience for the live book search project, a clickable TOC experience, mid 2007. More precisely, the task was to isolate TOC pages from the rest of the book, detect TOC entries and locate the target page where TOC entries are pointing to. In this paper we shall focus on the second part, TOC structure extraction.

There are two aspects of the problem; first one is related to acquiring/extracting information from raw OCR (words, lines and bounding boxes), i.e. performing in depth analysis of TOC pages. On any TOC page there are several types of information which we shall roughly divide in two: TOC entries and other. Each TOC entry is a single smallest group of words with the same title target somewhere in the book (usually specified by the page number at the end of the entry). Everything other than TOC entries we shall ignore, implying other stuff does not yield significant information to be treated. With this defined – TOC Structure Extraction Engine is responsible for providing following information about TOC entries:

- Separating TOC entries from other less important content of the page
- Separating TOC entries among themselves

- Establishing TOC entry target page
- Establishing TOC entry relative significance
- Determining TOC entry internal structure

Second aspect of the problem is related to the presentation of extracted information. For presentation purposes we have devised a labeling scheme which supports all possible scenarios for TOC pages.

2 Labeling Scheme

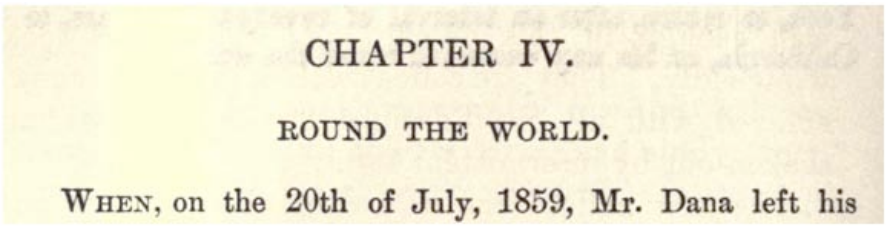
At this point we shall introduce several new ideas regarding the actual labeling of TOC pages. We have previously mentioned TOC entries in an informal way and now we shall define TOC entry in a more rigorous manner. TOC entry is a single referee to a target topic (part, chapter section, first line, first couple of words in some line...) somewhere in the book. The topic cannot begin in the middle of a line (we emphasize the case of a single word in the middle of the target line this is not considered to be TOC entry, but rather an index entry). In the same manner we shall disregard entries with a number of target topics (again, we would consider this entry to be an index rather than TOC).

To illustrate the point we have prepared an example, taken from Charles Francis Adams 1890 book "Richard Henry Dana", published by Houghton Boston. We took an excerpt from the table of content of that book. In Fig. 1 below we have presented three TOC entries. All three have the same structure; they all consist of a chapter number (IV, V, and VI) at the beginning of the line, then there is a title ("Round the world", and so on), a separator (a number of dots in each line), and finally a page number for each entry (178, 248, and 282).

Fig. 1. Example of three TOC entries

To ensure that each entry is in fact a TOC entry it is not sufficient to consider TOC page alone, target pages need be considered as well. As an illustration we shall observe page 178 and its content (shown in Fig. 2 below). After inspection it is clear that the first entry from Fig. 1 indeed targets page 178, i.e. there are both chapter number (the same as indicated in TOC entry) and chapter title (again, the same as indicated in TOC entry). We observe that the chapter number is of somewhat different structure than in the TOC entry on TOC page (namely, there is keyword "CHAPTER" in front of the actual chapter number, and the chapter name appears in the line preceding the actual title line), nevertheless the

target page is correct. Furthermore, both the chapter number and the chapter title are spatially separated from the rest of the text on that page (we shall refer to the rest of the page as a body), the font, although the same type and size, it is in all capital letters, which is yet another significant feature to further ensure that our target page is in fact correct.



CHAPTER IV.

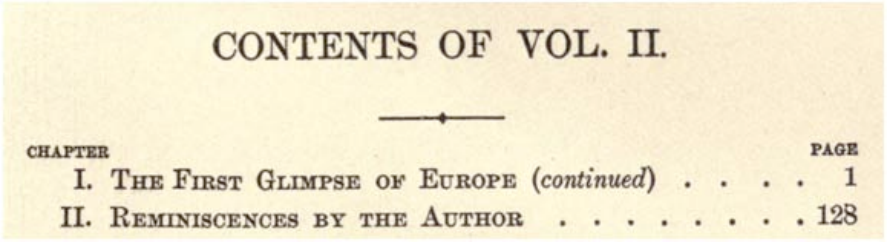
ROUND THE WORLD.

When, on the 20th of July, 1859, Mr. Dana left his

Fig. 2. Example of TOC entry target page

Given the simple example as a previous one, one can induce a minimal set of labels needed for labeling TOC entries. The most important label is clearly a title one, indicating a number of words that represent TOC entry title. In our labeling scheme we refer to this label as a TOC CHAPTER TITLE. Next in line of importance is the page number, providing information about the target page for each entry. In our labeling scheme we refer to this label as a TOC CHAPTER PAGE NUMBER. Then we have TOC CHAPTER NAME label, which in the example above refers to a chapter number IV; in general TOC chapter name label could consist of both keyword (like "CHAPTER" or "SECTION") and a number. At last there is TOC CHAPTER SEPARATOR label, which is self explanatory.

With the most important labels defined we are left with the duty of defining those less common/important. Again, we shall start with an example, excerpt from the same TOC page as in the previous example is taken and shown in Fig. 3 below.



CONTENTS OF VOL. II.

CHAPTER PAGE

I. The First Glimpse of Europe (*continued*) 1

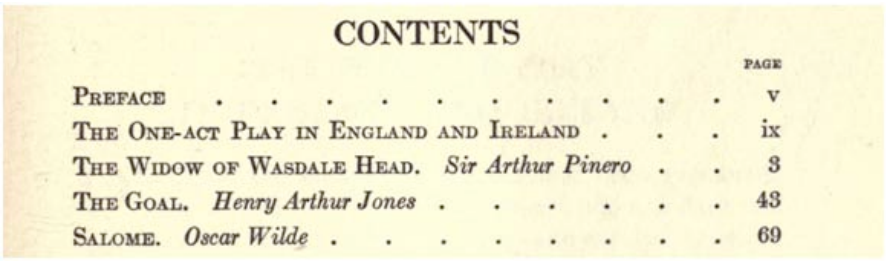
II. Reminiscences by the Author 128

Fig. 3. Example of TOC TITLE and NONE labels

Simple visual inspection is sufficient to ensure "CONTENTS OF VOL. II." should never be identified as a TOC entry, but rather as the title of the given TOC page. That is why we label this (and similar) group of words with a TOC TITLE label.

Another quick inspection is enough to rule out "CHAPTER" and "PAGE" words in the line preceding the first TOC entry; these two are surely not TOC entry (none of the two is targeting any title later in the book). To a viewer it is clear that the first word ("CHAPTER") stands for the horizontal position of chapter numbers, while the second word ("PAGE") stands for the horizontal position of page numbers for each TOC entry below. These two words are not of particular importance, especially compared to TOC entry labels, and that is why we introduce another label for negligible text, TOC NONE label. Because none of two words in question are actually linking to any target page, these words are not relevant for navigation nor for the indexing purposes that is how we validate our labeling decision. Furthermore, any other text that cannot possibly link to a target page is labeled the same way. A good example would be a page with severe errors in OCR, e.g. some text is recognized where no text exist.

Finally, there is one last type of less-important words that could be found on TOC pages. Observe the example in Fig. 4 (last three entries), where the author of the given title is specified in the same line with a chapter title, separator and page number. This example is taken from "Representative one-act plays" by Barrett Harper Clark, published by Brown Boston Little in 1921. It makes sense to distinguish a title from an author name; yet again, instead of an author name there could be any other text, and so far we observed several patterns, e.g. a year of publishing, a short summary of a chapter, rating, etc. All such text we label using TOC CHAPTER ADDITION label. We may decide to further fine sub-class addition label in the future, and use several labels instead.


CONTENTS

PAGE

PREFACE v

THE ONE-ACT PLAY IN ENGLAND AND IRELAND . . . ix

THE WIDOW OF WASDALE HEAD. *Sir Arthur Pinero* . . 3

THE GOAL. *Henry Arthur Jones* 43

SALOME. *Oscar Wilde* 69

Fig. 4. Example of TOC CHAPTER ADDITION labels

So far we have discussed labels for each word on TOC page, i.e. each word is labeled using one of the proposed labels. As a short summary we shall list them all here once more:

- TOC NONE label, for negligible text

- TOC TITLE label
- TOC entry labels:
  - TOC CHAPTER NAME label
  - TOC CHAPTER TITLE label
  - TOC CHAPTER SEPARATOR label
  - TOC CHAPTER PAGE NUMBER label
  - TOC CHAPTER ADDITION label

At this point we shall introduce another dimension to the labeling problem. It is clear that labeling of individual words is not sufficient. For instance by looking at Fig. 4 one can observe that the first two entries are somewhat different than the others; later entries are referencing one-act plays (as indicated by the book title), and the first two are giving preface and historical background for the plays in the book. On further inspection it can be observed that the first two entries have roman numerals for chapter page numbers, while the remaining TOC entries have arabic digits. This same pattern is observed to be common for most of the books we had access to. That is why we decided to introduce additional label/attribute for each TOC entry, where an entry is either of introductory or of regular type (with introductory entries being ones with roman numerals).

Third dimension of labeling problem is related to defining a TOC entry. There are cases of TOC pages where entries are in more than one line, e.g. second entry in Fig. 5 is in two lines (there are three entries in Fig. 5). In that and similar cases (where entries are in two or more lines) it is necessary to introduce yet another label for each word of each TOC entry, specifying whether any given word is a continuation of previous entry. If any given word is not a continuation of previous it is a beginning of a new TOC entry.

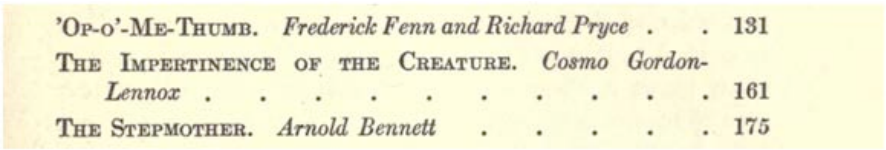


Fig. 5. Example of entry continuation

Last dimension of the labeling problem relates to establishing relative significance of each TOC entry on the TOC page. Relative significance of the entry is represented in form of a logical depth level. Observe structure presented in Fig. 6 (excerpt taken from "Managing and Maintaining a Microsoft Windows Server 2003 Environment for an MCSE Certified on Windows 2000" by Orin Thomas, published by Microsoft Press, 2005), there are five TOC entries presented in three structural/hierarchical groups, with a different relative significance assigned to each.

The entry in the first line is at highest hierarchical level, which is in this case indicated with keyword "CHAPTER", blue font color, bold characters, and

Fig. 6. Example of depth level

higher font size. Not to be forgotten, horizontal level of indentation is also a strong feature, especially for the following entries. The second and third TOC entries in picture 9 are one step below the level of the first one, resulting to level 2 in hierarchy. At last, TOC entries four and five are at level 3 in hierarchy.

We are now ready to summarize proposed labels for TOC page, four dimensions of the labeling problem are (categorized in hierarchical order):

- Hierarchical level of an entry
  - entry type (introductory vs. regular entries)
  - entry depth level
- Hierarchical level of a word
  - beginning vs. continuation of a word within an entry
  - word labels (name, title, separator, page number, addition, negligible text, toc page title)

3 Book Layout Engine

As previously stated, we consider raw OCR information at the input. This consists of words (strings), lines (list of words in each line) and bounding boxes of each word. Provided with this information we aim to detect and process TOC pages. There are a few prerequisites for the TOC Structure Extraction Engine, each of them listed below.

The first step in this process is to perform page classification and detect TOC pages. Next step would be detection of the page numbers, e.g. assigning each physical page of the book with unique logical page number. At last, each TOC page is processed to detect the scope of TOC section. These three steps are sufficient for the TOC Structure Extraction Engine to perform.

4 TOC Structure Extraction Engine

In the introduction we have specified the responsibilities of the TOC Structure Extraction Engine. Later on while discussing the labeling scheme we have specified the means of exposing this information. Here we shall discuss the engine itself.

It is worth noting that engine is developed on two sets of books, training and a blind test set, to prevent specialization. At last, the results presented later in

the text are measured against representative set (a third set, with no books from the previous two sets).

The first thing in the engine would be to distinguish between important and negligible portion of the TOC section. As specified by the labeling scheme, there are several possible cases of negligible text, the title of the TOC page, false positive OCR errors (usually words instead of the pictures), and random text. The engine is detecting each based on the pattern occurrences in the training set.

While it is feasible to separate entries among themselves without additional information, we have chosen to detect chapter names and chapter page numbers first, and only then proceed to the spatial entry detection. Again, this part of the engine is based on the pattern occurrences in the training set.

Once we have entries we can proceed to the linking, where a fuzzy search technique is used to locate entry title on the target page. It is worth noting that detection of page numbers (assigning each physical page of the book with unique logical page number) comes handy here, because it provides a significant hint where to look for the target title. Parameters of the fuzzy search are based on the pattern occurrences in the training set.

At last, relative significance of the entries is obtained after clustering, where each cluster represent a single level of significance. Parameters and features of the clustering are based on the training set.

5 Representative Set

In order to measure the accuracy of the TOC engine a fairly large test set needs to be selected due to the various layouts applied to books throughout history. Also, topics and publishers typically have specific templates for the book and TOC layout. Since storing large quantities of book data and performing tests on it is a costly process, a representative set has been created.

The representative set is a subset (200 books) of a much larger test set (180,000 books). It has 93% representativeness and gives a good picture of how the engine will perform in real-time situations.

6 Results

In this section we shall present some results on the representative set (blind set). Each of the four major parts of the TOC Structure Extraction are measured for precision and recall.

We shall start with the entry internal structure (word label) engine; chapter names are detected with 98.56% precision and 98.63% recall; chapter page numbers are detected with 99.97% precision and 99.07% recall; not surprising chapter titles and additions are detected with 99.36% precision and 99.80% recall. Total number of human labeled chapter page number words is 9,512, while total number of chapter name words is 7,206. Total number of chapter title and addition words is 52,571.

Spatial entry detection (we only consider entries which are entirely correct in a spatial sense – all words in an entry must be correct for the entry to be considered spatially correct) is with 92.91% precision and 95.45% recall. Total number of human labeled entries is 9535.

Linking detection is with 91.29% precision and 93.78% recall (98.25% conditional to entry being spatially correct).

Depth level 1 is detected with 80.51% precision and 84.58% recall (90.91% conditional to entry being spatially correct).

References

1. Ye, M. and Viola, P.: Learning to Parse Hierarchical Lists and Outlines Using Conditional Random Fields. In: Proceedings of the Ninth international Workshop on Frontiers in Handwriting Recognition, pp. 154–159. IEEE Computer Society, Washington, DC (2004)

XRCE Participation to the Book Structure Task (INEX 2008)

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 XRCE participation to the Structure Extraction task of the INEX Book track. After briefly explaining the method used for detecting table of contents and their corresponding entries in the book body, we will mainly discuss the evaluation and the main issues we faced, and eventually we will propose improvements for our method as well as for the evaluation framework/method.

1. Introduction

We present in this paper our participation to the Structure Extraction task of the INEX Book. Our objective was to assess a component, a table of contents detector, presented in [1], with the minimal effort. By minimal effort, we mean: the initial input (segmentation and text) was taken almost as provided. Especially, no preprocessing was used to improve it. But more important, no specific tuning was done with regard to the collection. This fact will be highlighted in the Evaluation Section, since some books do not comply with our assumptions about table of contents.

The rest of the article is structured as follows: we will explain the processing done for the collection. Then the method for detecting the Table of contents is sketched. We will explain the postprocessing and the different parameters used in our runs. Eventually we will discuss the results of the evaluation.

2. Pre-processing

The first step simply consists in reformatting the XML INEX format into our internal format, mostly renaming tag names and adding some internal attributes (such as unique IDs for each tag). This was performed using XSLT technology, with some difficulty for the largest books.

A second step consists in detecting pages headers and footers, which often introduce noisy for our table of contents detector (see [1]).

A heuristics has been used for run 4 in order to improve the ToC page detection based on the detected page headers and footers.

3. The ToC Detector

The method is detailed in [1] 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 theses hypotheses were not valid.

Three steps permit us to identify the area of the document containing the ToC text. Firstly, links are defined between each pair of text blocks in the whole document satisfying a textual similarity criterion. Each link includes a source text block and a target text block. The similarity measure we currently use is the ratio of words shared by the two blocks, considering spaces and punctuation as word separators. Whenever the ratio is above a predefined threshold, the similarity threshold, a pair of symmetric links is created. In practice, 0.5 is a good threshold value to tolerate textual variation between the ToC and the document body while avoiding too many noisy links. The computation of links is quadratic to the number of text blocks and takes most of the total computation time. However, searching for the ToC in the N first and last pages of the document leads to linear complexity without loss of generality.

Secondly, all possible ToC candidate areas are enumerated. A brute force approach works fine. It consists in testing each text block as a possible ToC start and extending this ToC candidate further in the document until it is no longer possible to comply with the five properties identified above. A ToC candidate is then a set of contiguous text blocks, from which it is possible to select one link per block so as to provide an ascending order for the target text blocks.

Thirdly, we employ a scoring function to rank the candidates. The highest ranked candidate table of contents is then selected for further processing. Currently, the scoring function is the sum of entry weights, where an entry weight is inversely proportional to the number of outgoing links. This entry weight characterizes the certainty of any of its associated links, under the assumption that the more links initiate from a given source text block, the less likely that any one of those links is a "true" link of a table of contents.

4. Post-Processing

This step mainly transforms the output of the ToC detector into the INEX format. Our component marks up the ToC entry and the body heading. From this information, the required page number was extracted. For the required title, we selected the title of the ToC entry, which, as we will see in the Evaluation section, will impact the evaluation.

5. The different runs

Several runs were conducted with different values for the main parameters, in particular the processing can be performed either at the line or paragraph level and the similarity measure can be the one described above, called Jaccard, or a dynamic time warping alternative (DTW). So we performed the following runs:

1. Paragraph level, Jaccard similarity: The Jaccard similarity consists in computing the ratio of the cardinal of the intersection to the union of normalized words of two text blocks, i.e. the paragraphs in this run.
2. Paragraph level, DTW similarity: the DTW consists in finding the best alignment of words of two blocks of text, the similarity between two words being established from an edit distance. The DTW similarity is more robust to OCR errors than the Jaccard but is computationally more intensive, usually twice more.
3. Line level, Jaccard similarity: here the considered blocks of text are the lines.
4. Paragraph Level, Jaccard similarity, with an additional heuristic to determine the ToC position.

We encountered memory issues with some of the largest documents and had to break our batches in several parts. Eventually we are not able to report accurately on the processing time.

Applying our standard ToC method prevented us from computing the level of the ToC entries, so we voluntarily set it to 1 for all entries despite this is clearly wrong.

6. Evaluation and Discussion

Let us review the results of the various runs, with a particular look at the first one because the other runs share many identical issues with it.

6.1. Run 1: Paragraph level, Jaccard similarity

We reproduce below the result of the Inex metric.

All books – run 1	Precision	Recall	F-Measure
Titles	25,98%	20,90%	22,13
Levels	14,43%	11,95%	12,45
Links	22,01%	18,16%	19,19
Complete entries	10,86%	9,30%	9,62
Entries disregarding depth	22,01%	18,16%	19,19

Table 1: Inex results for the run 1

These results appear overall quite bad. Actually, because the title matching between the run and the ground truth is critical to the evaluation method, any error on the title induces an error for all other criteria. For instance, a link with a valid destination but incorrect title will count as an error and a miss. In addition the conditions for title to match were quite strict, tolerating 20% of the shortest string as maximum edit distance with an additional condition on the first and last 5 characters. It turned out that an additional or missing word such as 'Chapter' as the beginning of the title suffices to discard entirely the title.

Under those conditions, 22% of precision at the link level with 26% correct titles, means in fact that among the entries with a correct title, 85% of them had a valid link. To examine this phenomenon further, we computed another link measure that ignores the title. We compare two links by comparing the page number they point to, so we consider a run output as a sequence of page numbers and compute the edit distance between the run and groundtruth sequences, which gives us a precision & recall measure. In other words, the Inex measure views an output as a unordered set of entries identified by their title, while our proposed complementary measure views an output as an ordered sequence of entries identified by the page pointed by each entry. Our measure, which we shall call 'hyperlink' focuses more on document navigation needs, and the quality of the extracted titles can be measured in a second step. Our 'hyperlink' measure is given Table 2 below.

All books	Precision	Recall	F-Measure
Hyperlinks (i.e. ignoring titles)	71%	40%	51

Table 2: 'Hyperlink' measure, which ignore title errors, for the run 1

This result is more conform to what we generally observe although the recall is particularly low. The histogram below shows an interesting aspect, where books tend to go either well or bad but more rarely in the middle. This behavior can be exploited thanks to automated quality assurance methods.

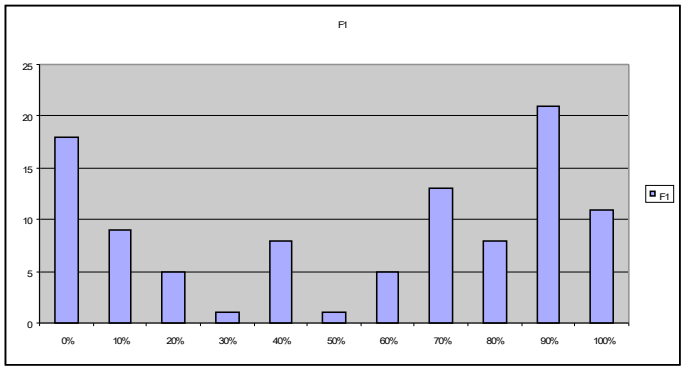


Table 3: Histogram of the "hyperlink" F1 distribution, for the run 1

Now, looking in detail at the error on the first book, we found several main causes of errors and misses:

- Ground truth errors: we saw 7 wrong additional entries, which seems to have been automatically generated from the ToC of an advert page at the end of the book (see page 640).
- Title errors: our run does not include the 'Chapter XX" at the beginning of the title, discarding about 90% of the found entries.
- One error is caused by the ToC not conforming to the ordering property, our method rely upon.
- Combined OCR noise and segmentation issues, .e.g. the ToC contains 'Chapter XX – some title' while the same information is split over two paragraph in the document body. Combined with OCR errors, typically a 'B' instead of a 'E', the similarity threshold is not met.

Unfortunately, those problems combined with the importance of the title in the Inex measure lead to important measure variation, as exemplified below on the first book:

Book #0 – run 1	Precision	Recall	F-Measure
Inex (title + linkls)	5%	4%	4
Hyperlinks, INEX ground truth	92%	69%	79
Hyperlinks, **fixed** ground truth	92%	80%	86

Table 3: Variable results depending on the measure, for book #0 in run 1

With respect to the ToC level, we have observed that the distribution of entries for level 1, 2 and 3 was about 33%, 36% and 26% respectively. This stresses the importance of reconstructing the ToC hierarchy.

6.2. Run 2: Paragraph level, DTW similarity

The DTW similarity should better deal with OCR errors, but to our surprise the results are not better. It turned out that the SW extracted 1% more links (36):

All books – run 2	Precision	Recall	F-Measure
Titles	24,38%	20,31%	21,45
Levels	13,15%	11,43%	11,90
Links	20,87%	18,02%	18,84
Complete entries	10,12%	9,26%	9,47
Entries disregarding depth	20,87%	18,02%	18,84

Table 4: Inex results for the run 2

Our 'hyperlinks' measure shows a minor improvement:

All books	Precision	Recall	F-Measure
Hyperlinks (i.e. ignoring titles)	71%	41%	52

Table 5: Hyperlink measure, which ignore title errors, for the run 2

6.3. Run 3: Line level, Jaccard similarity

Working at line level does not make much sense with the Inex evaluation method.

All books – run 3	Precision	Recall	F-Measure
Titles	16,53%	15,41%	15,46
Levels	7,44%	7,24%	7,13
Links	14,46%	13,47%	13,53
Complete entries	5,54%	5,42%	5,33
Entries disregarding depth	14,46%	13,47%	13,53

Table 6: Inex results for the run 3

Our Hyperlink measure shows a loss in precision.

All books	Precision	Recall	F-Measure
Hyperlinks (i.e. ignoring titles)	61%	42%	50

Table 7: Hyperlink measure, which ignore title errors, for the run 3

6.5. Ground Truth Issues

The ground truth used for this evaluation suffers from several coherence issues:

- Should the links point to the title page of a chapter or to the actual start of its text? Compare for instance the documents #13 0050CA95E49A5E97 and #20 00A4141D9CC87E65.
- Should the label of the entry (chapter, section,…) or its number be part of the extracted title? This choice has an enormous impact on the whole evaluation because of the importance of the title in the measure design. The choice made for the groundtruth is not consistent across all books. In fact, given a book, the choice can be difficult since the body and ToC pages can differ on this matter.
- ToC entry segmentation for old-style ToC, shown below, the subentries should be extracted or not, independently of the presence of a page locator since the document body shows clearly that there are subsections.

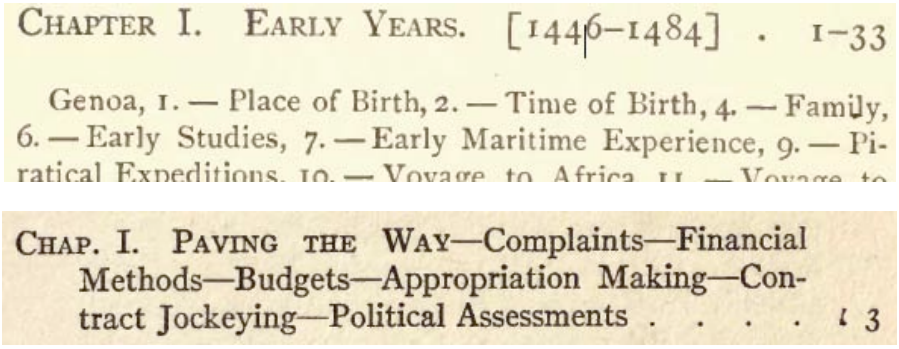


Figure 1: excerpt from book #3 (0008D0D781E665AD) and #52 (0E5E2F4BC9008492) showing a chapter title as well as the title of the subsections. In the first case the ground truth indicates the subsections but not in the second one (probably because of the absence of page number?).

6.5. Evaluation Method Issues and Suggestions

Given the previous observations, we suggest some improvement for measuring the quality of the results:

- Results at the book level should be made available
- Case normalization: the measure should be computed in a case-independent way. Indeed certain documents can have a title in uppercase in the ToC but in lowercase or capitalized form in the document body.
- In some applications, such as providing hyperlinks for navigation purpose, the quality of the links is more important than the exactitude of the title, provided it reads well and has some appropriate meaning. So we suggest measuring the link quality independently of the title quality. In fact, the latter should be measured as a complementary indication, e.g. computing an edit distance with the ground truth title.
- When the title is used as primary quality measure, a less strict title matching function should be used unless a sound and methodic way to determine uniformly the title has been designed.

7. Conclusion

It is difficult to draw conclusions because of the issues found with the ground truth, and to less extent with the evaluation method, which we could work around.

We have found very interesting the corpus proposed for the INEX, composed of historical documents with a large variety of table of contents. Many of them were challenging because of the need for segmenting entries at a lower level than the paragraph level, as shown in figure 1.

Some ToC did not respect at all the properties we enforce, since some ToC entries were short sentence *summarizing* the section contents rather than reproducing some title present in the document body, e.g. book #3 0008D0D781E665AD.

It would have been very useful to have a human evaluation of the results, in order to give a perspective different than the one underlying the ground truth preparation or quality measure design.

We are grateful to the organizers and thank them for their work, despite the numerical results are disputable in our opinion.

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)

Adhoc and Book XML Retrieval with Cheshire

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 also submitted two runs for the Adhoc Focused Element search task and one for the Best in Context task. For all of these runs we used the TREC2 logistic regression probabilistic model. For the Adhoc Element runs and Best in Context runs we used the "pivot" score merging method to combine paragraph-level searches with scores for document-level searches.

1 Introduction

In this paper we will first discuss the algorithms and fusion operators used in our official INEX 2008 Book Track and Heterogenous (Het) track runs. Then we will look at how these algorithms and operators were used in the various submissions for these tracks, and finally we will discuss problems in implementation, and 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 refering 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 [10, 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 [10]. As originally formulated, the LR model of probabilistic IR attempts to estimate the probability of relevance for each document based on a set of statistics about a document collection and a set of queries in combination with a set of weighting coefficients for those statistics. The statistics to be used and the values of the coefficients are obtained from regression analysis of a sample of a collection (or similar test collection) for some set of queries where relevance and non-relevance has been determined. More formally, given a particular query and a particular document in a collection $P(R \mid Q, D)$ is calculated and the documents or components are presented to the user ranked in order of decreasing values of that probability. To avoid invalid probability values, the usual calculation of $P(R \mid Q, D)$ uses the "log odds" of relevance given a set of S statistics derived from the query and database, such that:

$$\begin{aligned}
\log O(R|C,Q) = log\frac{p(R|C,Q)}{1-p(R|C,Q)} &= log\frac{p(R|C,Q)}{p(\overline{R}|C,Q)} \\
&= c_0 + c_1 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\frac{qtf_i}{ql+35} \\
&+ c_2 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\log\frac{tf_i}{cl+80} \\
&- c_3 * \frac{1}{\sqrt{|Q_c|}+1}\sum_{i=1}^{|Q_c|}\log\frac{ctf_i}{N_t} \\
&+ c_4 * |Q_c|
\end{aligned}$$

where C denotes a document component and Q a query, R is a relevance variable, and

$p(R|C,Q)$ is the probability that document component C is relevant to query Q,
$p(\overline{R}|C,Q)$ the probability that document component C is not relevant to query Q, (which is 1.0 - $p(R|C,Q)$)
$|Q_c|$ is the number of matching terms between a document component and a query,
qtf_i is the within-query frequency of the ith matching term,
tf_i is the within-document frequency of the ith matching term,
ctf_i is the occurrence frequency in a collection of the ith matching term,
ql is query length (i.e., number of terms in a query like $|Q|$ for non-feedback situations),
cl is component length (i.e., number of terms in a component), and
N_t is collection length (i.e., number of terms in a test collection).
c_k are the k coefficients obtained though the regression analysis.

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)[12] and TREC-8 (Voorhees and Harman 1999)[13].)

Blind relevance feedback is typically performed in two stages. First, an initial search using the original queries is performed, after which a number of terms are selected from the top-ranked documents (which are presumed to be relevant). The selected terms are weighted and then merged with the initial query to formulate a new query. Finally the reweighted and expanded query is run against the same collection to produce a final ranked list of documents. It was a simple extension to adapt these document-level algorithms to document components for INEX.

The TREC2 algorithm has been been combined with a blind feedback method developed by Aitao Chen for cross-language retrieval in CLEF. Chen[2] 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" (qtf_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 qtf_i. For terms in the top 10 and in the original query the new qtf_i is set to 1.5 times the original qtf_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 Result Combination Operators

As we have also reported previously, the Cheshire II system used in this evaluation provides a number of operators to combine the intermediate results of a search from different components or indexes. With these operators we have available an entire spectrum of combination methods ranging from strict Boolean operations to fuzzy Boolean and normalized score combinations for probabilistic and Boolean results. These operators are the means available for performing fusion operations between the results for different retrieval algorithms and the search results from different different components of a document. For Hetergeneous search we used a variant of the combination operators, where MINMAX normalization across the probability of relevance for each entry in results from each sub-collection was calculated and the final result ranking was based on these normalized scores.

In addition, for the Adhoc Thorough runs we used a merge/reweighting operator based on the "Pivot" method described by Mass and Mandelbrod[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:

$$P(R \mid Q, C_{new}) = (X * P(R \mid Q, C_{comp})) + ((1 - X) * P(R \mid Q, C_{art})) \quad (2)$$

Where X is a pivot value between 0 and 1, and $P(R \mid Q, C_{new})$, $P(R \mid Q, C_{comp})$ and $P(R \mid Q, C_{art})$ are the new weight, the original component weight, and article weight for a given query. Although we found that a pivot value of 0.54 was most effective for INEX04 data and measures, we adopted the "neutral" pivot value of 0.4 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

We used the latest version of the Wikipedia database for this year's Adhoc runs, and created a number of indexes similar to those described in previous INEX papers[9].

Table 2. Wikipedia Article-Level Indexes for INEX 2008

Name	Description	Contents	Vector?
docno	doc ID number	//name@id	No
names	Article Title	//name	Yes
topic	Entire Article	//article	Yes

topicshort	Selected Content	//fm/tig/atl
		//abs
		//kwd
		//st
			Yes

xtnames	Template names	//template@name	No
figure	Figures	//figure	No
table	Tables	//table	No
caption	Image Captions	//caption	Yes
alltitles	All Titles	//title	Yes

links	Link Anchors	//collectionlink
		//weblink
		//wikipedialink
			No

Table 2 lists the document-level (/article) indexes created for the INEX database and the document elements from which the contents of those indexes were extracted.

As noted above the Cheshire system permits parts of the document subtree to be treated as separate documents with their own separate indexes. Tables 3 & 4 describe the XML components created for INEX and the component-level indexes that were created for them.

Table 3. Wikipedia Components for INEX 2006

Name	Description	Contents
COMPONENT_SECTION	Sections	//section
COMPONENT_PARAS	Paragraphs	//p | //blockquote | //indentation1| //indentation2|//indentation3
COMPONENT_FIG	Figures	//figure

Table 3 shows the components and the path used to define them. The first, COMPONENT_SECTION, component consists of each identified section in all of the documents, permitting each individual section of a article to be retrieved separately. Similarly, each of the COMPONENT_PARAS and COMPONENT_FIG components, respectively, treat each paragraph (with all of the alternative paragraph elements shown in Table 3), and figure (<figure> ... </figure>) as individual documents that can be retrieved separately from the entire document.

Table 4. Wikipedia Component Indexes for INEX 2006†Includes all subelements of section or paragraph elements.

Component or Index Name	Description	Contents	Vector?
COMPONENT_SECTION			
sec_title	Section Title	//section/title	Yes
sec_words	Section Words	*†	Yes
COMPONENT_PARAS			
para_words	Paragraph Words	*†	Yes
COMPONENT_FIG			
fig_caption	Figure Caption	//figure/caption	No

Table 4 describes the XML component indexes created for the components described in Table 3. These indexes make individual sections (COMPONENT_SECTION) of the INEX documents retrievable by their titles, or by any terms occurring in the section. These are also proximity indexes, so phrase searching is supported within the indexes. Individual paragraphs (COMPONENT_PARAS) are searchable by any of the terms in the paragraph, also with proximity searching. Individual figures (COMPONENT_FIG) are indexed by their captions.

Few of these indexes and components were used during Berkeley's simple runs of the 2006 INEX Adhoc topics. The two official submitted Adhoc runs and scripts used in INEX are described in the next section.

We decided to try the same methods on the Book Track data this year, but we did not use multiple elements or components, since the goal of the main Books Adhoc task was to retrieval entire books and not elements. We did, however create the same indexes for the Books and MARC data that we created last year as 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 last year's Book track runs.

Table 5. MARC Indexes for INEX Book Track 2008

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).

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.

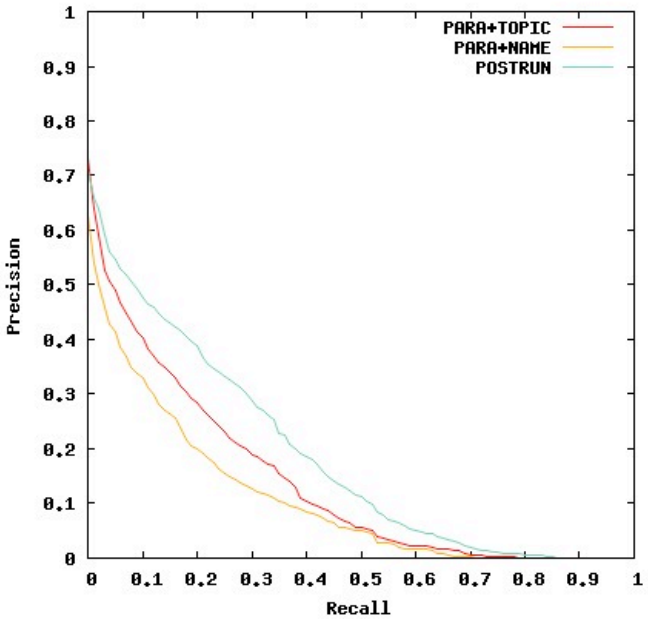


Fig. 1. Berkeley Adhoc Element Retrieval Results

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 indentifiers 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.

We encountered a number of system problems dealing with the Books database this year, since the numbers unique terms exceeded 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

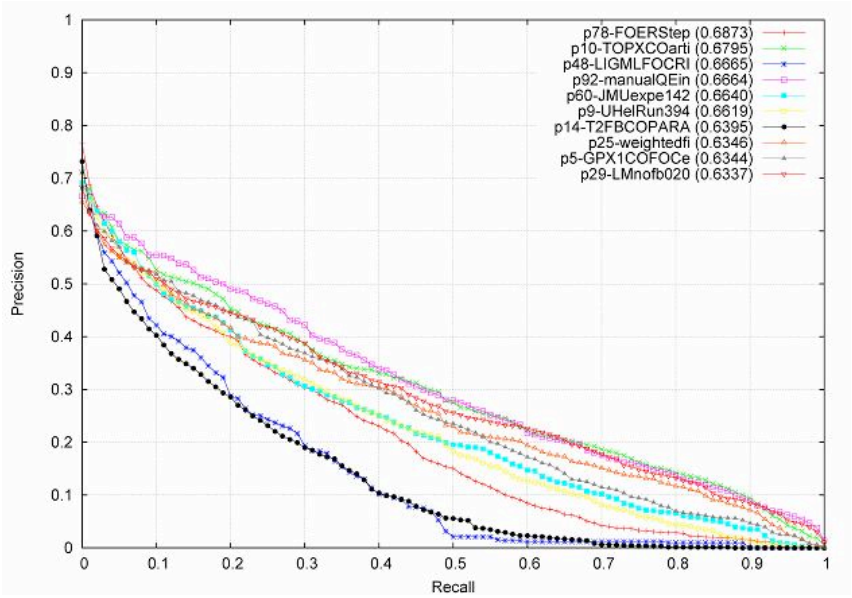


Fig. 2. Top 10 (by group) Adhoc Retrieval Runs

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 Adhoc Track Runs

We submitted three runs this year to the Adhoc "Focused" track, two for the CO Element search tasks, and one for the "Best in context" task. Figure 1 shows the precision/recall curves for the two Element search runs. The better performing run used a fusion of paragraph with full document (topic) search and had an iP at 0.01 of 0.6395. This was ranked fourteenth out of the sixty-one runs submitted, and since many of the top-ranked runs came from the same groups, the run appeared in the "top ten" graph on the official site (reproduced as Figure 2). As Figure 2 shows, the run was fairly strong for precision at low recall levels, but overall showed a lack of recall placing it much lower than the majority of the top ten submissions at higher recall levels. We had intended this run to use the blind feedback mechanism described above, but fail to specify it correctly. As a result the run used only the TREC2 logistic regression algorithm and the weighted merging described above, but with no blind feedback.

Our second "Focused" run used a fusion of paragraphs with searches on the "name" element of the Wikipedia documents. This run, also shown in Figure 1

had an iP at 0.01 of 0.5410, and ranked forty-fourth out of sixty-one runs - a much poorer showing. This run also was intended to use the blind feedback, but did not.

As an unofficial experiment after we discovered the lack of blind feedback in these runs, we ran the same script as for our first official run (using paragraphs and the topic index) but correctly specified the use of blind feedback in addition to the TREC2 Logistic Regression algorithm. This unofficial run obtained a iP at 0.01 of 0.6586, which would have ranked tenth if it had been submitted. This run is shown in Figure 1 as "POSTRUN" and indicates how the lack of blind feedback in our official submissions adversely impacted the results.

Our Best in context run basically took the raw results data from the Adhoc focused run using paragraphs and topics, and selected the top-ranked paragraphs for each document as the "best in context", all other elements were eliminated from the run. The results clearly show that this is not a very effective strategy, since our results of a MAgP of 0.0533 was ranked thirty-second out of thirty-five runs submitted overall.

5 INEX 2008 Book Track Runs

We submitted three runs for the Book Search task of the Books track, one using MARC data only, one using full Book contents only, and a third performing a merge of the MARC and Book data. Evaluation was delayed for the Book track and we do not yet have any results.

We are also participating in the "Active Reading" task of the Books track which is still getting underway.

6 Conclusions and Future Directions

For all of the Adhoc (focused and best in context) runs that we submitted this year, only paragraphs were used as the retrieved elements, and we did not (as in previous years) attempt to merge the results of searches on multiple elements. This helps to partially explain both the low recall for the Focused task and the low score of the Best in context task. The failure to correctly specify blind feedback in the runs also had a negative impact. However, since this is the first year when we have managed to submit any runs for the Focused task without them being disqualified for overlap, we can consider it a significant improvement. We note for the future that double-checking scripts before running them, and submitting the results is very good idea.

References

1. A. Chen. Multilingual information retrieval using english and chinese queries. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum, CLEF-2001, Darmstadt, Germany, September 2001*, pages 44–58. Springer Computer Scinece Series LNCS 2406, 2002.

2. A. Chen. *Cross-Language Retrieval Experiments at CLEF 2002*, pages 28–48. Springer (LNCS #2785), 2003.
3. A. Chen and F. C. Gey. Multilingual information retrieval using machine translation, relevance feedback and decompounding. *Information Retrieval*, 7:149–182, 2004.
4. W. S. Cooper, A. Chen, and F. C. Gey. Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression. In *Text REtrieval Conference (TREC-2)*, pages 57–66, 1994.
5. W. S. Cooper, F. C. Gey, and 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 approaches for thorough and heterogeneous xml retrieval. In *Advances in XML Information Retrieval: INEX2006*, pages 318–330. Springer (LNCS #4518), 2005.
10. 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.
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. E. Voorhees and D. Harman, editors. *The Seventh Text Retrieval Conference (TREC-7)*. NIST, 1998.
13. E. Voorhees and D. Harman, editors. *The Eighth Text Retrieval Conference (TREC-8)*. NIST, 1999.

RMIT University at the INEX Book Search Track

Mingfang Wu, Falk Scholer, and James A. Thom

RMIT University, Melbourne, Australia
{mingfang.wu,falk.scholer,james.thom}@rmit.edu.au

Abstract. This paper describes the RMIT group's participation in the INEX book search track in 2008.

1 Introduction

This paper describes the participation of the RMIT group in the Initiative for the Evaluation of XML retrieval (INEX) book search track in 2008, specifically the book retrieval task. This is the first year that RMIT has participated in the book search track at INEX, and we explore effectiveness of book retrieval task by experimenting with various approaches for query construction, and indexing and retrieving books at different granulaties.

2 Our Approach

We explore two broad categories of approaches in our runs:

1. How to construct the query from the INEX topic, and
2. Whole book retrieval compared with two different techniques for book retrieval based on page retrieval.

We submitted 8 runs in total, 4 of each category.

2.1 Query Construction

INEX book track topics contain different components: title, description, task, and information need. An example is shown in Figure 1. We begin by describing four different approaches to constructing the query from the topic:

RmitBookTitle: Use the words in the topic title.
RmitBookTitleInfneed: Use the words in the topic title and the topic information need elements.
RmitBookTitleBoolean: Boolean "AND" is applied to all query terms in **RmitBookTitle**.
RmitBookTitleInfneedManual.xml: Use all terms as in **RmitBookTitle**, and add some manually selected terms from **RmitBookTitleInfneed**.

```
<inex_topic track="bs07" task="book-ad-hoc" topic_id="20"  ct_no="277">
<title>Last Supper painting da Vinci</title>
<description>
I am looking for description, critique, theories and explanations of
Leonardo d Vinci's painting of the Last Supper.
</description>
<narrative>
<task>
I want to gather background information on the painting of da Vinci
titled Last Supper for an essay I need to write.
</task>
<infneed>
I would like to find out as much as possible about the painting.
What is known about its commissioning and delivery.
When it was painted, where? How is its topic and composition of its
characters explained. The technique used in painting it.
I am also interested in its history, how it survived the times,
where is it exhibited. I am not interested in copies of the painting.
Images are also not relevant.
</infneed>
</narrative>
</inex_topic>
```

Fig. 1. A sample topic

Consider the sample topic shown in Figure 1. Applying our four different approaches results in the following query terms (after stopping and stemming):

RmitBookTitle: Last Supper paint da Vinci
RmitBookTitleInfneed: supper paint da vinci find possibl paint known commiss deliveri paint topic composit charact explain techniqu paint interest histori surviv time exhibit interest copi paint imag relev
RmitBookTitleBoolean: last AND supper AND paint AND da AND vinci
RmitBookTitleInfneedManual: supper paint da vinci paint commiss deliveri paint topic composit charact techniqu paint histori surviv time exhibit

We submitted four runs, each using one of the querying strategies described above. The Zettair search engine[1] was used to index the collection. For retrieval, the BM25 similarity function [1] was used for document weighting and ranking (with $k1 = 1.2, b = 0.75$).

In the above four runs, a book is an indexable unit. Stemming and stopping are applied during indexing time, and to queries.

2.2 Ranking Based on Page-level Evidence

Our other four runs are variations on retrieval, using individual pages as the indexable unit, and combining results to rank books.

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

Sometimes, a topic is just mentioned in passing in a book, and may not be the main theme of the book. To rank those books that are primarily dedicated to the topic more highly, we require a topic be mentioned on most of the book's pages. We therefore experimented with the following two strategies:

1. The top 3000 retrieved pages per query are merged according to the books that they are sourced from. The books are then weighted and ranked according to percentage of pages that are retrieved per book.
2. For each query, the top 3000 pages are retrieved and then merged according to their originating books, based on the percentage of the maximum number of continuous pages that are retrieved per book.

Combing those two page-level evidence ranking methods and two querying strategies, we submitted four further runs to the track:

RmitPageMergeTitle: Query terms are the same as in **RmitBookTitle**, books are ranked according to method 1;
RmitConPageMergeTitle: Query terms are the same as in **RmitBookTitle**, books are ranked according to method 2;
RmitPageMergeTitleManual: Query terms are the same as in **RmitBookTitleInfneedManual**, books are ranked according to method 1;
RmitConPageMergeTitleManual: Query terms are the same as in **RmitBookTitleInfneedManual**, books are ranked according to method 2.

3 Concluding Remarks

Since evaluation for the book retrieval task was only just starting at the time of submission to the preproceedings, we are unable to compare the effectiveness of the different approaches at this stage.

References

1. S. Robertson, S. Walker, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of TREC-3*, pages 109–126. Available online at trec.nist.gov/pubs/, 1994.

Overview of the INEX 2008 Efficiency Track

Martin Theobald[1] and Ralf Schenkel[1,2]

[1] Max-Planck-Institut Informatik, Saarbrücken, Germany
[2] Saarland University, Saarbrücken, Germany

Abstract. This paper presents an overview of the Efficiency Track that was newly introduced to INEX in 2008. The new INEX Efficiency 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*. As opposed to the purely synthetic XMark or XBench benchmark settings that are still prevalent in efficiency-oriented XML retrieval tasks, the Efficiency Track continues the INEX tradition using a rich pool of manually assessed relevance judgments for measuring retrieval effectiveness. Thus, one of the main goals is to attract more groups from the DB community to INEX, being able to study effectiveness/efficiency trade-offs in XML ranked retrieval for a broad audience from both the DB and IR communities. 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 General Setting

1.1 Test Collection

Just like most INEX tracks, the Efficiency Track uses the 2007 version of the INEX-Wikipedia collection [2] (without images), an XML version of English Wikipedia articles initially introduced for INEX 2006 and slightly revised in 2007. The collection is available for download from the INEX website http://www.inex.otago.ac.nz/ for registered participants, or directly from http://www-connex.lip6.fr/ denoyer/wikipediaXML/. Although this 4.38 GB XML-ified Wikipedia collection is not particularly large from a DB point-of-view, it has a rather irregular structure with many deeply nested paths, which will be particularly challenging for traditional DB-style approaches, e.g., using path summaries. There is no DTD available for INEX-Wikipedia.

1.2 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 topic `description` and `narrative` fields providing hints for the human assessors or allowing for more semi-automatic query expansion settings, see Figure 1.3). The Efficiency Track focuses on the following types of of queries (also coined "topics" in good IR tradition), each representing different retrieval challenges:

- **Type (A) Topics:** 540 topics (no. 289–828) are taken over from previous Ad-hoc Track settings used in 2006–2008, which constitute the major bulk of topics used also for the Efficiency Track. These topics represent classic, Ad-Hoc-style, focused passage or element retrieval (similar to the INEX Ad-Hoc Focused subtask 2006–2008, see for example [3]), with a combination of NEXI CO and CAS queries. Topic ids are taken over from the Ad-Hoc track as well, thus allowing us to reuse assessments from the Ad-Hoc Track for free.

- **Type (B) Topics:** 21 topics (no. 829–849) are derived from interactive, feedback-based query expansion runs, kindly provided by the Royal School Of Library And Information Science, Denmark, investigated in the context of the INEX Interactive Track 2006 [5, 6]. These CO topics are intended to simulate high-dimensional query expansion settings with up to 112 keywords (topic no. 844), which cannot be evaluated in a conjunctive manner and are expected to pose a major challenge to any kind of search engine. Respective expansion runs have been submitted by RSLIS also to the 2006 Ad-Hoc track, such that relevant results are expected to have made it into the relevance pools of INEX 2006 Ad-Hoc track assessments as well. An additional `adhocid` attribute marks the original Ad-Hoc id of the topic that it has been derived from, such that—at least incomplete—assessments are available for these type (B) topics.

- **Type (C) Topics:** 7 new topics (no. 850–856) have been developed and submitted by Efficiency Track participants in 2008. These topics 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. Assessments were originally intended to get accomplished by Efficiency Track participants, as well, but were then skipped due to their low amount and the low respective impact on result effectiveness as compared to the more than 500 Ad-Hoc topics that already come readily assessed. The evaluation of runtimes however remains very interesting over this structure-enhanced set of type (C) topics.

1.3 Topic Format and Assessments

Just like the original NEXI queries, type (A) queries have some full-text predicates such as phrases (marked by quotes ""), mandatory keywords (+), and keyword negations (-). Although participants were encouraged to use these full-text hints, they are not mandatory just like in other INEX tracks. Because of their high-dimensional nature, most type (B) and (C) queries require IR-style, non-conjunctive (aka. "andish") query evaluations

that can either preselect the most significant query conditions or dynamically relax both the structure- and content-related conditions at query processing time. The reuse of type (A) and (B) lead to 308 topics for which assessments from the INEX 2006–2008 Ad-hoc Tracks are available. An additional conversion to the new 2008 version of the INEX-Eval tool and the (passage-based) assessments format was needed to incorporate the 2008 assessment files (QRels) and has meanwhile been made available online for download from the track homepage http://www.inex.otago.ac.nz/tracks/efficiency/efficiency.asp.

```
<topic id="856" type="C">
<co_title>
  State Parks Geology Geography +Canyon
</co_title>
<cas_title>\
  //article//body[about(.//section//p, State Park) and
                  about(.//section//title, Geology) and
                  about(.//section//title, Geography)]
                //figure[about(.//caption, +Canyon)]
</cas_title>
<xpath_title>
  //article//body[.//section//p ftcontains "State Park" and
                  .//section//title ftcontains "Geology" and
                  .//section//title ftcontains "Geography"]
                //figure[.//caption ftcontains "Canyon"]
</xpath_title>
  <description>
    I'm looking for state parks with sections describing
    their geology and/or geography, preferably with a figure of
    a canyon as target element.
  </description>
  <narrative>
    State park pages often follow the common pattern of having
    sections entitled with "Geology" or "Geography". I'm
    particularly interested in those pages with a figure of a
    canyon, e.g., the Grand Canyon.
  </narrative>
</topic>
```

Fig. 1. Example type (C) topic (no. 856)

All topic titles are provided in the NEXI syntax (in both their CO and CAS formulations) and (new for the Efficiency Track) in their corresponding XPath 2.0 Full-Text specification. XPath 2.0 queries were automatically generated from the respective NEXI CAS titles, while the CAS title itself was taken over from the CO title and wrapped into a pseudo target element of the form `//*[about(...)]` whenever there was no actual CAS title available.

A "Call for Structure-Enhanced Queries" for the new type (C) queries was issued to all registered INEX participants in early May 2008. The final set of 568 Efficiency Track topics was released in early July 2008, with the intention to keep a relatively tight time window between the release of the topics and the run submission deadline to prevent people from overtuning to particular topics.

1.4 Sub-Tasks

The Efficiency Track particularly encourages the use of top-k style query engines. The result submission format includes options for marking runs as top-15, top-150, and top-1500 (the latter corresponding to the traditional Ad-hoc submission format), using either a `Focused` (i.e., non-overlapping), `Thorough` (incl. overlap), or `Article` retrieval mode (see below). Automatic runs may use either the title field, including the NEXI CO, CAS, or XPATH titles, additional keywords from the narrative or description fields, as well as automatic query expansions if desired. At least one automatic and sequential run with topics being processed one-by-one is mandatory for each participating group. Participants are invited to submit as many runs in different retrieval modes as possible.

- **Article:** This is the mode that corresponds to a classical search engine setting to the largest extent. All documents may be considered either in their plain text or XML version. Moreover, queries can be used in both their CO and CAS (incl. XPath 2.0 Full-Text) formulation. In the Article-Only/CO combination this setting resembles a classical IR setting with entire documents as retrieval units and plain keywords as queries. Article-only runs are always free of overlapping results.

- **Thorough:** The Thorough mode represents the original element-level retrieval mode used in INEX 2003-2005. Here, any element correctly identified as relevant to the query will contribute to the recall of the run. This setting intentionally allows overlapping elements to be returned, since removing overlap may mean a substantial burden for different systems. We thus re-conceal the Thorough setting used in previous INEX years with respect to efficiency aspects, such that actual query processing runtimes can be clearly distinguished from the runtime needed to remove overlapping results (which typically is a costly post-processing step).

- **Focused:** Focused (i.e., overlap-free) element- and/or passage-level 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 runtimes should include the time needed to remove overlap, which may give rise to interesting comparisons between systems following both Thorough and Focused retrieval strategies.

2 Run Submissions

The submission format for all Efficiency Track retrieval modes is defined by the following DTD, depicted in Figure 2. The following paragraph provides a brief explanation of the DTD fields:

- Each *run* submission *must* contain the following information:
  - `participant-id` - the INEX participant id
  - `run-id` - your run id
  - `task` - either `focused`, `thorough`, or `article`

```
<!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           (article|thorough|focused) #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 >
<!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 >
<!ELEMENT result (file, path, rank?, rsv?) >
<!ELEMENT file   (#PCDATA)>
<!ELEMENT path   (#PCDATA)>
<!ELEMENT rank   (#PCDATA)>
<!ELEMENT rsv    (#PCDATA)>
```

Fig. 2. DTD for Efficiency Track run submissions

- `query` - either automatic or manual mode (at least one automatic mode using exactly one of the title fields is required; in manual mode any form of manual query expansion is allowed)
- `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 (optional)
- `hardware_year` - date of purchase of the hardware (optional)

    - `topk` - top-k run or not (if it is a top-k run, there may be at most k elements per topic returned)

- 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

Providing CPU and I/O times is optional for each topic. Also, it is sufficient to provide a list of matching elements along with their `path` locators (as canonical XPath expressions—see, again, the Ad-hoc Track settings [3]). Providing `rank` and relevance score values `rsv` was also optional. In `article` retrieval mode, all results' elements paths had to be `/article[1]`. Moreover, the many different types of (optional) description fields are supposed to encourage participants to provide detailed descriptions along with their run submissions.

Particularly interesting for the Efficiency Track submissions is the `runtime` field, of course. This can optionally be split into `cpu_time` and `io_time`, the latter two of which had not been used by any of the participants, though. So we focus on actual wallclock running times as efficiency measure for our 2008 setting. Top-k runs with less than 1,500 ranks have only been submitted by the Max-Planck-Institut Informatik. Distribution has only been used by the University of Frankfurt, using a cluster with 8 nodes.

3 Metrics

To assess the quality of the retrieved results, the Efficiency Track applies the same metrics as used in the Ad-Hoc track. Runs in `Focused` or `Article` mode were evaluated with the interpolated precision metric [4], using the evaluation toolkit from INEX 2008; the assessments for the topics from 2006 and 2007 have been converted to the

new Qrel-based format. Runs in `Thorough` mode were evaluated with the precision-recall metric as implemented in inex_eval [1] after converting the Qrels from 2008 to the old XML-based assessment format. In the future, the track will probably use metrics implemented in EvalJ[3].

4 Participants

An overall amount of 20 runs was submitted by 5 participating groups. The following paragraphs provide short system descriptions submitted by the participants.

Max-Planck-Institut Informatik [10] For the INEX Efficiency Track 2008, we were just on time to finish and (for the first time) evaluate our brand-new TopX 2.0 prototype. Complementing our long-running effort on efficient top-k query processing on top of a relational back-end, we now switched to a compressed object-oriented storage for text-centric XML data with direct access to customized inverted files, along with a complete reimplementation of the engine in C++. Core of the new engine is a multiple-nested block-index structure that seamlessly integrates top-k-style sorted access to large blocks stored as inverted files on disk with in-memory merge-joins for efficient score aggregations.

University of Frankfurt [16] University of Frankfurt has developed Spirix, 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. The runs submitted to the INEX efficiency track compare different structure similarity functions which are then used to improve efficiency of routing and ranking.

University of Toronto [42] Summary not yet available.

University of Twente & CWI [53] Summary not yet available.

JustSystems Corporation [56] Summary not yet available.

[3] http://evalj.sourceforge.net/

5 Results

Table 5 summarizes all run parameters as they were delivered in the runs' headers. Table 5 summarizes all effectiveness (iP, MAiP) and efficiency results (avg.&sum of wall-clock runtimes in milliseconds) for the respective number of topics processed (#topics). Tables 5–5 summarize the results by topic type for all Focused runs (with effectiveness results only being available for type (A) and (B) topics). Figure 3 depicts detailed interpolated precision plots for all Focused and (the only) Article-only run(s); while Figure 4 depicts classic precision-recall plots for the Thorough runs. Figures 5–6 finally depict the respective interpolated precision plots split by type (A) and (B) topics (type (C) plots are skipped due to the lack of assessments).

Part.ID	Run ID	Task	#CPU	RAM	#Nodes	Hardw.Cost	Year	Top-k	Cache	Seq.	Aut.	Title Fields
10	TOPX2-Eff08-CAS-15-Focused-W	Foc.	4	16	1	8,000 Eur	2005	15	OS+TopX	Yes	Yes	CAS
10	TOPX2-Eff08-CAS-15-Thorough-W	Tho.	4	16	1	8,000 Eur	2005	15	OS+TopX	Yes	Yes	CAS
10	TOPX2-Eff08-CAS-150-Focused-	Foc.	4	16	1	8,000 Eur	2005	150	OS+TopX	Yes	Yes	CAS
10	TOPX2-Eff08-CAS-1500-Focused-W	Foc.	4	16	1	8,000 Eur	2005	1500	OS+TopX	Yes	Yes	CAS
10	TOPX2-Eff08-CO-15-Focused-W	Foc.	4	16	1	8,000 Eur	2005	15	OS+TopX	Yes	Yes	CO
10	TOPX2-Eff08-CO-15-Thorough-W	Tho.	4	16	1	8,000 Eur	2005	15	OS+TopX	Yes	Yes	CO
10	TOPX2-Eff08-CO-150-Focused-W	Foc.	4	16	1	8,000 Eur	2005	150	OS+TopX	Yes	Yes	CO
10	TOPX2-Eff08-CO-1500-Focused-W	Foc.	4	16	1	8,000 Eur	2005	1500	OS+TopX	Yes	Yes	CO
16	001-Uni Frankfurt,Strict	Foc.	8	16	8	n/a	n/a	1500	n/a	Yes	Yes	CAS
16	002-Uni Frankfurt,Baseline	Foc.	8	16	8	n/a	n/a	1500	n/a	Yes	Yes	CAS
16	003-Uni Frankfurt,Architect-S	Foc.	8	16	8	n/a	n/a	1500	n/a	Yes	yes	CAS
16	004-Uni Frankfurt,Fine-Sim	Foc.	8	16	8	n/a	n/a	1500	n/a	Yes	Yes	CAS
16	005-Uni Frankfurt,Path-Sim	Foc.	8	16	8	n/a	n/a	1500	n/a	Yes	Yes	CAS
42	B2U0_full-depth-heur	Foc.	1	2	1	n/a	n/a	1500	Lucene	Yes	Yes	CAS
42	B2U0_full-depth-sr	Tho.	1	2	1	n/a	n/a	1500	Lucene	Yes	Yes	CAS
53	pftijah_article_strict	Art.	1	8	1	1,000 Eur	2008	1500	DBMS	Yes	Yes	CO
53	pftijah_asp_strict	Tho.	1	8	1	1,000 Eur	2008	1500	DBMS	Yes	Yes	CAS
53	pftijah_asp_vague	Tho.	1	8	1	1,000 Eur	2008	1500	DBMS	Yes	Yes	CAS
53	pftijah_star_strict	Tho.	1	8	1	1,000 Eur	2008	1500	DBMS	Yes	Yes	CAS
56	VSM_RIP	Foc.	1	2	1	1,500 USD	2004	1500	None	Yes	Yes	CO+CAS

Table 1. Run parameters as taken from the submission headers

6 Conclusions

This paper gave an overview of the INEX 2008 Efficiency Track. We intend to continue and expand this track in upcoming INEX years, thus hoping to increase the general visibility of this project and to attract more people from the DB&IR fields to efficient XML-IR settings. We also aim to establish the Efficiency Track, along with its large body of IR-style topics and readily available assessments, as a reference benchmark for more realistic XML-IR experiments outside the INEX community. One step towards this direction was to introduce queries in the more common XPath 2.0 Full-Text syntax.

References

1. Overview of the INitiative for the Evaluation of XML retrieval (INEX) 2002. In N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, editors, *INitiative for the Evaluation of XML Re-*

Part.ID	Run ID	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP	AVG MS.	SUM MS.	#Topics
Focused									
10	TOPX2-Eff08-CAS-15-Focused-W	0.4587	0.3878	0.2592	0.1918	0.0662	90.99	51,499	566
10	TOPX2-Eff08-CAS-150-Focused-W	0.4747	0.4282	0.3494	0.2915	0.1094	112.32	63,574.00	566
10	TOPX2-Eff08-CAS-1500-Focused-W	0.4824	0.4360	0.3572	0.3103	0.1241	253.42	143,436	566
10	TOPX2-Eff08-CO-15-Focused-W	0.4751	0.4123	0.2793	0.1971	0.0726	49.79	28,180	566
10	TOPX2-Eff08-CO-150-Focused-W	0.4955	0.4520	0.3674	0.3114	0.1225	85.96	48,653	566
10	TOPX2-Eff08-CO-1500-Focused-W	0.4994	0.4560	0.3749	0.3298	0.1409	239.73	135,688	566
16	001-Uni Frankfurt,Strict	0.0035	0.0035	0.0034	0.0034	0.0007	188,862.50	15,109,000	80
16	002-Uni Frankfurt,Baseline	0.1969	0.1926	0.1834	0.1724	0.0867	186,565.00	14,925,200	80
16	003-Uni Frankfurt,Architect-Sim	0.2070	0.1960	0.1812	0.1669	0.0768	326,098.75	26,087,900	80
16	004-Uni Frankfurt,Fine-Sim	0.1971	0.1927	0.1758	0.1626	0.0747	197,731.25	15,818,500	80
16	005-Uni Frankfurt,Path-Sim	0.2077	0.1983	0.1826	0.1661	0.0718	321,493.75	25,719,500	80
42	B2U0_full-depth-heur	0.4388	0.3964	0.3344	0.3013	0.1357	2,994.00	1,679,634	561
56	VSM_RIP	0.4836	0.4058	0.3077	0.2553	0.0895	4,807.55	2,730,687	568
Article									
53	pftijah_article_strict	0.4599	0.4272	0.3689	0.3346	0.1839	701.98	398,722	568
Thorough			**P@0.01**	**P@0.05**	**P@0.10**	**MAP**			
10	TOPX2-Eff08-CAS-15-Thorough-W		0.1811	0.0288	0.0069	0.0053	89.31	50,549	566
10	TOPX2-Eff08-CO-15-Thorough-W		0.1890	0.0357	0.0084	0.0065	70.91	40,133	566
42	B2U0_full-depth-sr		0.2196	0.0541	0.0077	0.0080	9,172.05	5,145,519	561
53	pftijah_asp_strict		0.2674	0.1008	0.0294	0.0136	2,306.08	1,309,854	568
53	pftijah_asp_vague		0.2653	0.1120	0.0357	0.0141	8,213.05	4,665,010	568
53	pftijah_star_strict		0.2415	0.1029	0.0471	0.0169	17,186.03	9,761,663	568

Table 2. Effectiveness/efficiency summary of all runs

Part.ID	Run ID	MAiP	AVG MS.	SUM MS.	#Topics
10	TOPX2-Eff08-CO-15-Focused-W	0.0712	18.88	10,157	538
10	TOPX2-Eff08-CO-150-Focused-W	0.1234	49.12	26,427	538
10	TOPX2-Eff08-CO-1500-Focused-W	0.1430	191.27	102,903	538
10	TOPX2-Eff08-CAS-15-Focused-W	0.0643	48.84	26,276	538
10	TOPX2-Eff08-CAS-150-Focused-W	0.1094	61.25	32,953	538
10	TOPX2-Eff08-CAS-1500-Focused-W	0.1249	165.53	89,055	538
16	001-Uni Frankfurt,Strict	0.0007	188,862.50	15,109,000	80
16	002-Uni Frankfurt,Baseline	0.0867	186,565.00	14,925,200	80
16	003-Uni Frankfurt,Architect-Sim	0.0768	326,098.75	26,087,900	80
16	004-Uni Frankfurt,Fine-Sim	0.0747	197,731.25	15,818,500	80
16	005-Uni Frankfurt,Path-Sim	0.0169	17,186.03	9,761,663	568
42	B2U0_full-depth-heur	0.1373	2,716.45	1,450,584	534
53	pftijah_article_strict	0.1884	604.51	326,438	540
56	VSM_RIP	0.0936	4,253.85	2,297,077	540

Table 3. Summary over all 540 type (A) topics (Focused runs only)

Part.ID	Run ID	MAiP	AVG MS.	SUM MS.	#Topics
10	TOPX2-Eff08-CO-15-Focused-W	0.0915	844.67	17,738	21
10	TOPX2-Eff08-CO-150-Focused-W	0.1094	1038.90	21,817	21
10	TOPX2-Eff08-CO-1500-Focused-W	0.1125	1468.67	30,842	21
10	TOPX2-Eff08-CAS-15-Focused-W	0.0915	1044.71	21,939	21
10	TOPX2-Eff08-CAS-150-Focused-W	0.1096	1074.66	22,568	21
10	TOPX2-Eff08-CAS-1500-Focused-W	0.1124	1479.33	31,066	21
42	B2U0_full-depth-heur	0.1143	8,052.14	169,095	21
53	pftijah_article_strict	0.1224	3,212.52	67,463	21
56	VSM_RIP	0.0329	14,583.33	306,250	21

Table 4. Summary over all 21 type (B) topics (Focused runs only)

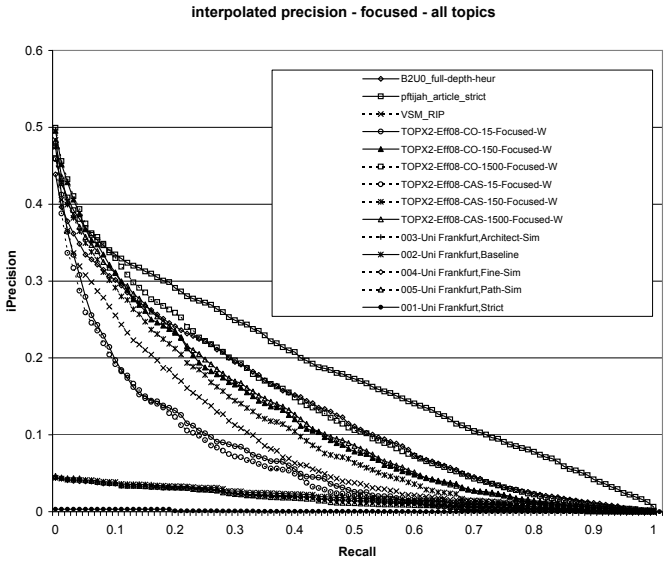


Fig. 3. Interpolated precision plots for all Focused and Article runs

Part.ID	Run ID	MAiP	AVG MS.	SUM MS.	#Topics
10	TOPX2-Eff08-CO-15-Focused-W	n/a	41.00	287	7
10	TOPX2-Eff08-CO-150-Focused-W	n/a	58.86	412	7
10	TOPX2-Eff08-CO-1500-Focused-W	n/a	277.57	1,943	7
10	TOPX2-Eff08-CAS-15-Focused-W	n/a	469.42	3,286	7
10	TOPX2-Eff08-CAS-150-Focused-W	n/a	1150.14	8,051	7
10	TOPX2-Eff08-CAS-1500-Focused-W	n/a	3330.71	23,315	7
42	B2U0_full-depth-heur	n/a	14,629.86	102,409	7
53	pftijah_article_strict	n/a	688.71	4,821	7
56	VSM_RIP	n/a	18,194.29	127,360	7

Table 5. Summary over all 7 type (C) topics (Focused runs only)

trieval (INEX). Proceedings of the First INEX Workshop. Dagstuhl, Germany, December 8–11, 2002, ERCIM Workshop Proceedings, Sophia Antipolis, France, March 2003. ERCIM. http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf.

2. L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 2006.
3. N. Fuhr, J. Kamps, M. Lalmas, S. Malik, and A. Trotman. Overview of the INEX 2007 Ad Hoc Track. In *INEX*, pages 1–23, 2007.
4. 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.
5. S. Malik, B. Larsen, and A. Tombros. Report on the INEX 2005 Interactive Track. *SIGIR Forum*, 41(1):67–74, 2007.
6. S. Malik, A. Tombros, and B. Larsen. The Interactive Track at INEX 2006. In *INEX*, pages 387–399, 2006.

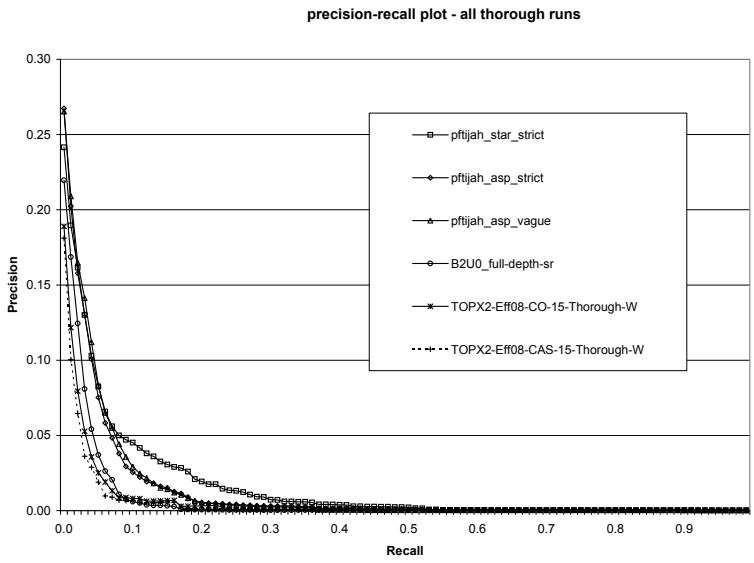


Fig. 4. Precision-recall plots for all Thorough runs

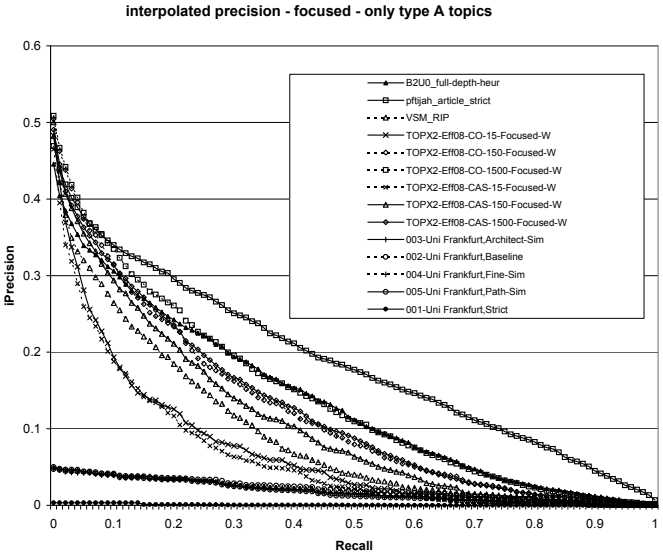


Fig. 5. Interpolated precision plots for type (A) Focused runs

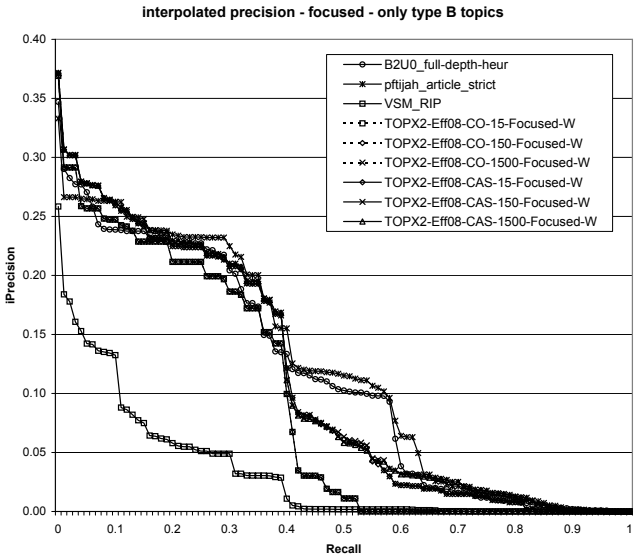


Fig. 6. Interpolated precision plots for type (B) Focused runs

Efficient XML and Entity Retrieval with PF/Tijah: CWI and University of Twente at INEX'08

Henning Rode[1], Djoerd Hiemstra[2], Arjen de Vries[1], Pavel Serdyukov[2]

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

Abstract.

1 Introduction

PF/Tijah is a research prototype created by the University of Twente and CWI Amsterdam with the goal to create a flexible environment for setting up search systems. By integrating the PathFinder (PF) XQuery system [1] with the Tijah XML information retrieval system [2] it combines database and information retrieval technology. The PF/Tijah system is part of the open source release of MonetDB/XQuery developed in cooperation with CWI Amsterdam and the University of Tübingen.

PF/Tijah is first of all a system for *structured retrieval* on XML data. Compared to other open source retrieval systems it comes with a number or unique features [3]:

- It can execute any NEXI query without limits to a predefined set of tags. Using the same index, it can easily produce a "focused", "thorough", or "article" ranking, depending only on the specified query and retrieval options.
- The applied retrieval model, score propagation and combination operators are set at query time, which makes PF/Tijah an ideal experimental platform.
- PF/Tijah embeds NEXI queries as functions in the XQuery language. This way the system supports ad hoc result presentation by means of its query language. The efficiency task submission described in the following section demonstrates this feature. The declared function `INEXPath` for instance computes a string that matches the desired INEX submission format.
- PF/Tijah supports text search combined with traditional database querying, including for instance joins on values. The entity ranking experiments described in this article intensively exploit this feature.

With this year's INEX experiments, we try to demonstrate the mentioned features of the system. All experiments were carried out with the least possible pre- and post-processing outside PF/Tijah. Section 2 shows with the application of the system to the INEX efficiency track, how a wide range of different NEXI queries can be executed efficiently. Section 3 demonstrates how combined database and retrieval queries provide a direct solution to specialized tasks like entity ranking.

2 Efficiency

The INEX efficiency task combines retrieval quality and performance. In order to test the performance on a wide range of different queries, the task uses a query set of 568 structured queries combined from other tasks and collected over the last years. The queries vary with respect to the contained number of query terms and structural requirements. A subset for instance represents typical relevance feedback queries containing a considerable higher number of query terms.

The retrieval efficiency of PF/Tijah was improved in the last year with respect to several aspects, which we wanted to test by our submissions. The index structure, containment joins, and score computation had been changed [4] to improve the execution of simple query patterns such as

```
//tag[about(., term query)]
```

PF/Tijah creates a full-text index on top of Pathfinder's pre/post encoding of XML files [5]. Instead of assigning a pre-order value to complete text-nodes as done by the Pathfinder, the Tijah full-text index enumerates each single term. Both the Pathfinder encoding and the separate full-text index are held in database tables. An "inverted" table is created by clustering the table (pre-order values) on tag- and term ID.

PF/Tijah does not use top-k query processing strategies. Neither tag-term pairs nor scores are precalculated or indexed in order avoid redundancy on the one hand, and to allow at query time the application of arbitrary ranking functions on the other hand. The applied ranking function is specified in PF/Tijah for each single query. Furthermore, PF/Tijah's containment join operator relies on input sorted in document order. Node sequences sorted on score order as they are typically accessed in the top-k query processing framework do not match this requirement. PF/Tijah does not implement any caching strategy itself. However, the underlying database system tries to make use of the operating system's caching functionalities.

2.1 Submissions

We submitted in total 4 runs, 1 "article" ranking and 3 "thorough" element rankings. Since PF/Tijah does not support top-k query processing, all submitted runs return the complete set of the 1500 highest ranked elements for each query. The applied ranking function for all submissions follows the language modeling framework for retrieval. The so-called NLLR, normalized logarithmic likelihood ratio, compares for each query term its distribution within element and query model. The ranking aggregates single term scores on the level of scored elements. Query terms marked by a leading '-' to indicate that they should *not* occur in relevant elements were removed from the queries, since PF/Tijah currently does not support this feature. For the same reason, phrases were treated as normal query terms only.

For repeatability we report here the complete XQuery that was used to produce the ranking in PF/Tijah. The XQuery below was generated for Topic 856.

The individual queries only substitute the inside NEXI string accordingly. The costly function call producing the required INEX path string was omitted when running time measurements, since it does not reflect the retrieval performance itself:

```
declare function INEXPath($n as node()) as xs:string
{
   let $paths :=
       for $a in $n/ancestor-or-self::*
       where local-name($a) ne "docs"
       return if (local-name($a) eq "article")
          then concat(local-name($a),"[1]")
          else concat(local-name($a),"[",
              string(1 + count($a/preceding-sibling::*
              [local-name() eq local-name($a)])),"]")
   return string-join($paths, "/")
};

let $opt := <TijahOptions returnNumber="1500" ir-model="NLLR"
             prior="NO_PRIOR" txtmodel_returnall="FALSE"/>
let $nexi := "//article//body[about(.//section//p, State Park) and
             about(.//section//title, Geology) and
             about(.//section//title, Geography)]
             //figure[about(.//caption, Canyon)]"
return <topic id="856"> {
   for $res at $rank in tijah:queryall($nexi, $opt)
   return <result><file> {
      concat("",$res/ancestor-or-self::article/name/@id)}</file>
      <path>{INEXPath($res)}</path>
      <rank>{$rank}</rank></result> }
</topic>
```

For the *article* ranking we automatically created NEXI queries by substitution of the placeholder `?CO-TITLE?` below with the content-only (CO) field of the query topic:

```
//article[about(., ?CO-TITLE?)]
```

The run should show how our XML retrieval system performs when used as a standard document retrieval system.

In contrast, the "thorough" element rankings use the content-and-structure (CAS) field of each query topic. The first "thorough" run, called *star-strict*, executes the unmodified CAS query as provided in the query topic. The final two runs perform a slight modification. Since the new PF/Tijah system is tuned towards queries starting with a tag-name selection rather than searching in all element nodes, we translated queries starting with the pattern

```
//*[about(., terms)]...
```

to

```
//(article|section|p)[about., terms)]...
```

The runs based on this modification are called *asp-strict* and *asp-vague*. The distinction between both is explained in the following.

Thinking in terms of XPath, the base of the NEXI language, the scoring predicates `[about(., terms)]` are first of all evaluated to a boolean value, causing those elements to pass that satisfy the query. If the predicates are translated to a scoring operator in the algebra tree, that only assigns scores to all elements, the predicate becomes obsolete as a filter and the side effect of the predicate evaluation, the score assignment, has become the primary aim. This is clearly not the only possible query interpretation. We can require that an element has to reach a certain score threshold in order to satisfy the predicate condition. The least strict setting of such a threshold would be to filter out all zero scored element. In other words, the `about` function would assign a `true` value to all elements that contain at least one of the query terms. For a query of the form

```
//article[about(., xml)]//p[about(.,ir)]
```

strict semantics will pass only those articles that match the keywords of the first about, whereas *vague* semantics also considers results of paragraphs about "ir" that are not occurring within articles about "xml". The two submitted runs, *asp-strict* and *asp-vague*, compare the different query interpretation with respect to retrieval quality and performance.

2.2 Results

Test System setup The test system used for all time measurements in this article was an INTEL Core2 Quad machine running on 2.4 Ghz with 8 GB main memory. The necessary index structures could hence be held in memory, but not in the considerably smaller CPU caches. Queries were executed sequentially. For time measurements, we omitted the generation of the INEXPath as mentioned above and stored only node identifiers instead. We measured the full execution of the query, including the query compilation phase.

run	avg time	sum time	min time	max time
article	0.702	399	0.327	11.814
star-strict	17.186	9762	0.324	330.495
asp-strict	2.306	1310	0.324	52.388
asp-vague	8.213	4665	0.444	1235.572

Table 1. Execution time overview in sec

Table 1 shows an overview on the execution times of the different runs. The article ranking is obviously faster on average than the three other runs evaluating

the CAS query. Since some CAS queries in the query set issue a simple fielded search, it is not surprising that the minimal execution time stays almost the same for all runs. Looking at the average and maximal execution time for a single query, we observe, however, huge differences. Most of the time differences can be attributed to queries that contain the pattern `//*` in the NEXI path. If a posting list of a certain tagname is fetched from the inverted index, the system guarantees the pre-order sortedness of the list, which is required for the subsequent containment evaluation. Fetching the entire inverted index, however, will not return a pre-order sorted element list, and therefore requires a resorting of the entire node set. The difference becomes apparent when comparing the execution times of the two runs *star-strict* and *asp-strict*. Even the expensive substitute pattern selecting all `article`, `section`, and `p` nodes shows still a better performance.

Evidently, the application of *strict* query semantics yield a better query performance. The average total execution is around 4 times faster than in the case of a *vague* interpretation. The early filtering on intermediary result sets especially helps on highly structured queries. Consequently, we observe similar minimal execution times but clear differences when looking at the highest times measured for evaluating a single query. The differences of the two query interpretations needs to be studied as well in terms of retrieval quality.

run	MAiP	iP[0.10]	iP[0.5]	iP[0.01]
article	0.1839	0.3346	0.3689	0.4272
	MAP	P@0.10	P@0.5	P@0.01
star-strict	0.0169	0.0471	0.1029	0.2415
asp-strict	0.0136	0.0294	0.1008	0.2674
asp-vague	0.0141	0.0357	0.1120	0.2653

Table 2. Retrieval quality presented in official measures

Table 2 reports the official measurements used in the efficiency track, which differ for "article" and "thorough" run submissions. Therefore, we can only compare our three "thorough" runs. The substitution of `//*`-queries sacrifices recall but not early precision. The two *asp* runs even yield a slightly higher precision on top of the ranked list. Comparing the *strict* and *vague* semantics we observe as expected a better retrieval quality when applying the vague "andish" interpretation. The differences, however, stay again small when looking at the top of the retrieved list.

3 Entity Ranking

The INEX entity ranking task searches for entities rather than articles or elements with respect to a given topic. With entities we mean here unique instances

of a given type, such as "Hamburg" and "München" being an instance of type "German cities". For a given query topic such as "hanseatic league" and target entity type "German cities" a good entity retrieval system should return "Hamburg", but not "München" since it is off topic, or "Novgorod" since it is not a German city.

The target type is given as a Wikipedia category in the INEX task. Furthermore, each retrieved entity needs to have its own article in the Wikipedia collection. Obviously, this decision is only suitable for entity ranking within an encyclopedia, where we can assume that most mentioned entities in fact have their own entry. In consequence, a baseline ranking is achieved by a straightforward article ranking on the Wikipedia corpus combined with an appropriate category filtering mechanism.

The INEX task further provides a few relevant example entities for each query topic. The given entities can be used as relevance feedback to improve the initial text query or to redefine the set of target categories. Another application for the example entities comes with the list completion task. This task asks to derive appropriate target categories automatically from the given relevant entities.

Our main aim for this year's track participation was to express entity ranking queries completely in the XQuery language. Hence, we wanted to show that PF/Tijah is "out of the box" able to express and evaluate complex entity ranking queries with a high retrieval quality. One preprocessing step, however, turned out to be unavoidable. The INEX wikipedia corpus comes without category tagging in the provided XML format. Instead, the categorization of all articles is provided by separate plain text files. In order to unify all given information, we integrated the category tagging in the XML corpus itself as shown in the following example:

```
<article><name id="13467">Hamburg</name>
    <body>....</body>
    <category id="5654">cities in germany</category>
    <category id="52414">port cities</category>
</article>
```

Next to the title keywords, target categories, and relevant entities provided with each search topic, we generated for each search topic an additional list of relevant derived categories. Namely those categories assigned to the specified relevant entities. The derived relevant categories are used as mentioned above for refinement of the target categories as well as for the list completion task:

```
for $topic in doc("topics.xml")//inex_topic
let $relevant_entities := $topic//entity/@id
return collection("wikipedia")//
       article[name/@id =
$relevant_entities]//category/@id
```

3.1 Submissions

We submitted in total 6 runs, 4 runs for the entity ranking task, and 2 list completion submissions. The submissions can also be divided into 3 runs based

solely on a direct article ranking, and 3 other runs using also the scores of adjacent articles in the link graph.

We start by describing the direct article rankings. The ranking and category filtering is performed by a single XQuery, which is shown below. The fields fields ?QID?, ?QTERMS?, ?CATS?, ?DERIVEDCATS? were substituted according to the given query topic:

```
    (: part1 - retrieval :)
    let $query_num := "?QID?"
    let $q_terms := tijah:tokenize("?QTERMS?")
    let $opt := <TijahOptions ir-model="LMS" returnNumber="1000"
                collection-lambda="0.5"/>
    let $nexi := concat("//article[about(.,", $q_terms, ")]")
    let $tijah_id := tijah:queryall-id($nexi, $opt)
    let $nodes := tijah:nodes($tijah_id)

    (: part2 - determine target categories :)
    let $targetcats := distinct-values(((?CATS?), (?DERIVEDCATS?)))

(: part3 - filtering and output generation :)
for $a at $rank in $nodes
let $score := if ($a//category/@id = $targetcats)
              then tijah:score($tijah_id, $a)
              else tijah:score($tijah_id, $a) * 0.0000001
order by $score descending
return string-join((string($query_num), "Q0", concat("WP",$a/name/@id),
       string($rank), string($score), "ER_TEC"), " ")
```

The presented XQuery ranks in the first part all articles of the Wikipedia collection according to the topic of the query. We applied here a standard language modeling retrieval model with the smoothing factor set to $\lambda = 0.5$. Moreover, the result set was limited to the top 1000 retrieved articles.

The second part determines the target categories. Whereas our first run *ER_TC* uses only the categories provided with the query topic, the second run *ER_TEC* refines the target category set by uniting the given and derived categories as shown in the query. The list completion *LC_TE*, on the other hand, uses only the derived but not the given categories.

The final part performs the actual filtering and required TREC-style output generation. Notice that the applied filtering in fact only performs a reordering and does not remove articles from the ranked list. Last year's experiments had clearly shown that the reordering comes with a higher recall compared to the filtering technique.

The other 3 runs *ER_TC_idg*, *ER_TEC_idg*, *LC_TE_idg* exploit the retrieval scores of adjacent nodes and follow otherwise a symmetrical experiment schema with respect to the used target categories. The underlying idea behind the exploitation of link structure is adopted from other entity ranking tasks such as expert finding, where we typically find a number of topical relevant documents that mention relevant entities, but entities do not have a textual description themselves. A sample cutout of such a graph is visualized in Figure 1. The edges here

symbolize containment of entities within documents. Entities are then ranked by a propagation of scores from adjacent documents.

Although entity ranking on the Wikipedia corpus is different since entities are represented by their own articles and have a text description themselves, it still often occurs the articles outside the target category carry valuable information for the entity ranking. Recall the above given example query searching for German cities in the hanseatic league. We will find Wikipedia entries about the history of the hanseatic league listing and linking to all major participating cities. While such article remains outside the target category, the links to relevant city pages are of high value for the ranking. Especially, when a city's description itself does not reach far enough into history. We developed last year a ranking method matching this condition [6]. The personalized weighted indegree measure tries to combine the article ranking itself $w(e|q)$ with the ranking of other Wikipedia entries $w(e'|q)$ linking entity e:

$$PwIDG(e) = \mu w(e|q) + (1-\mu) \sum_{e' \in \Gamma(e)} w(e'|q) \tag{1}$$

A corresponding indegree score computation can be expressed as well in XQuery. The below shown query part substitutes the score computation in the previous entity ranking example and sets the parameter μ to 0.85:

```
for $a at $rank in $nodes
let $in_score := sum(
     for $l in $nodes//collectionlink[@*:href =
          concat($a/name/@id, ".xml")]
     let $source_article := exactly-one($l/ancestor::article)
     return tijah:score($tijah_id, $source_article)
)
let $score := if ($a//category/@id = $targetcats)
    then 0.85 * tijah:score($tijah_id, $a) + 0.15 * $in_score
    else (0.85 * tijah:score($tijah_id, $a) + 0.15 * $in_score)
         * 0.0000001
order by $score descending
return string-join((string($query_num), "Q0", concat("WP",$a/name/@id),
     string($rank), string($score), "1_cirquid_ER_TEC_idg"), " ")
```

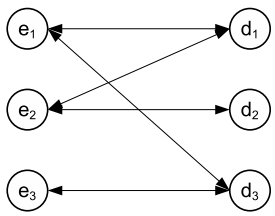


Fig. 1. Part of a link graph containing entities e_i and other documents d_j

Notice that each link between two entities is counted separately here. We tested before a version of the query that establishes only one link between two entities e_1 and e_2 even if e_1 links e_2 multiple times. Initial tests on last years data indicated, however, a higher retrieval quality for the above presented query.

3.2 Training

We trained the parameter μ on the data of last year's entity ranking task. For the chosen relevance propagation method a setting of $\mu = 0.85$ showed the best performance with respect to precision on top of the retrieved list as well as for mean average precision:

μ	0.8	0.85	0.9	0.95
MAP	0.3373	**0.3413**	0.3405	0.3349
P5	**0.4435**	**0.4435**	0.4304	0.4348
P10	0.3739	**0.3783**	0.3717	0.3630

3.3 Results

The results will be shown in the final version of this paper, but have not been evaluated at the time writing the notebook paper.

4 Conclusions

We demonstrated with this article the flexibility and effectiveness of the chosen approach to integrate the retrieval language NEXI with the database query language XQuery. The PF/Tijah system allows to express a wide range of INEX experiments without changes to the system itself. Often time consuming pre- and post-processing of data is not necessary or reduced to simple string substitutions of query terms for each given query.

Although PF/Tijah does not apply top-k query processing techniques, it shows a good performance on a wide range of NEXI queries. Future developments should address the currently bad supported retrieval on the entire node set, issued by `//*`-queries.

The INEX entity ranking task demonstrates how standard retrieval functions can be applied to non-standard retrieval tasks with the help of score propagation expressed on the XQuery level. A combined DB/IR system as PF/Tijah can demonstrate here its full advantage.

References

1. Boncz, P., Grust, T., van Keulen, M., Manegold, S., Rittinger, J., Teubner, J.: MonetDB/XQuery: a fast XQuery processor powered by a relational engine. In: SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM (2006) 479–490

2. List, J., Mihajlovic, V., Ramírez, G., de Vries, A., Hiemstra, D., Blok, H.E.: Tijah: Embracing Information Retrieval methods in XML databases. Information Retrieval Journal **8**(4) (2005) 547–570
3. Hiemstra, D., Rode, H., van Os, R., Flokstra, J.: PF/Tijah: text search in an XML database system. In: Proceedings of the 2nd International Workshop on Open Source Information Retrieval (OSIR), Seattle, WA, USA, Ecole Nationale Supérieure des Mines de Saint-Etienne (2006) 12–17
4. Rode, H.: From Document to Entity Retrieval. PhD thesis, University of Twente, CTIT (2008)
5. Grust, T., van Keulen, M., Teubner, J.: Accelerating XPath evaluation in any RDBMS. ACM Trans. Database Syst. **29** (2004) 91–131
6. Rode, H., Serdyukov, P., Hiemstra, D.: Combining Document- and Paragraph-Based Entity Ranking. In: Proceedings of the 31th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008). (2008) 851–852

TopX 2.0 at the INEX 2008 Efficiency Track

- A (Very) Fast Object-Store for Top-k-style XML Full-Text Search -

Martin Theobald[1], Mohammed AbuJarour[3], and Ralf Schenkel[1,2]

[1] Max-Planck-Institut Informatik, Saarbrücken, Germany
[2] Saarland University, Saarbrücken, Germany
[3] Hasso-Plattner-Institut, Potsdam, Germany

Abstract. For the INEX Efficiency Track 2008, we were just on time to finish and (for the first time) evaluate our brand-new TopX 2.0 prototype. Complementing our long-running effort on efficient top-k query processing on top of a relational back-end, we now switched to a compressed object-oriented storage for text-centric XML data with direct access to customized inverted files, along with a complete reimplementation of the engine in C++. Core of the new engine is a multiple-nested block-index structure that seamlessly integrates top-k-style sorted access to large blocks stored as inverted files on disk with in-memory merge-joins for efficient score aggregations. The main challenge in designing this new index structure was to marry no less than three different paradigms in search engine design: 1) sorting blocks in descending order of the maximum element score they contain for threshold-based candidate pruning and top-k-style early termination; 2) sorting elements within each block by their id to support efficient in-memory merge-joins; and 3) encoding both structural and content-related information into a single, unified index structure. Our INEX 2008 experiments demonstrate efficiency gains of up to a factor of 30 compared to the previous Java/JDBC-based TopX 1.0 implementation over a relational back-end. TopX 2.0 achieves overall runtimes of less than 51 seconds for the entire batch of 568 Efficiency Track topics in their content-and-structure (CAS) version and less than 29 seconds for the content-only (CO) version, respectively, using a top-15, focused (i.e., non-overlapping) retrieval mode—an average of merely 89 ms per CAS query and 49 ms per CO query.

1 Introduction

The database group (Department 5) at the Max-Planck-Institut Informatik traditionally is very interested in applying efficient database technology to challenging information retrieval tasks. Our most recent XML-IR search engine, TopX [10], has meanwhile become a mature and well established, default search engine for the topic development phase in the INEX Ad-hoc Track and also serves as Webservice endpoint for the Interactive Track. Thus a natural step forward was to focus in more detail on efficiency aspects in XML-IR for the new INEX 2008 Efficiency Track.

TopX is a native IR-engine for semistructured data with IR-style, non-conjunctive (aka. "andish") query evaluations, which is particular challenging for efficient XPath-like query evaluations because of the huge intermediate set of candidate result elements,

i.e., when any XML element matching any of the query conditions may be a valid result. In this "andish" retrieval mode, the result ranking is solely driven by score aggregations, while the query processor needs to combine both content-related and structural aspects of so-called content-and-structure (CAS) queries into a single score per result element. Here, high scores for some query dimensions may compensate weak (even missing) query matches at other dimensions. Thus, the query processor may dynamically relax query conditions if too few matches would be found in a conjunctive manner, whereas the ranking allows for the best (i.e., top-k) matches be cut-off if too many results would be found otherwise. These query evaluations are more difficult to evaluate than typical DB-style conjunctive queries, as we can no longer use conjunctive merge-joins (corresponding to intersections of index list objects), but we need to find efficient ways of merging index lists in a non-conjunctive manner (corresponding to unions of index list objects, or so-called "outer-joins" in DB terminology). Top-k-style evaluation strategies are crucial in these XML-IR settings, not only for pruning index list accesses (i.e., physical I/O's) but also for pruning intermediate candidate objects that need to be managed dynamically in memory at query processing as early as possible. Pruning also the latter in-memory data structures is particularly beneficial for good runtimes if CPU-time becomes a dominating factor, e.g., when XPath evaluations are costly, or when many index lists already come from cache.

TopX 2.0 combines query processing techniques from our original TopX 1.0 prototype [11, 10] and carries over ideas for block-organized inverted index structures from our IO-Top-k algorithm [1] to the XML case. The result is a novel, object-oriented storage (albeit our index objects have a very regular structure) for text-oriented XML data, with sequential, stream-like access to all index objects. Just like the original engine, TopX 2.0 also supports more sophisticated cost-models for sequential and random access scheduling along the lines of [6, 7, 10]. In the following we focus on a description of our new index structure and its relation to the element-specific scoring model we propose, while the non-conjunctive XPath query processor remains largely unchanged compared to TopX 1.0 (merely using merge-joins over entire index blocks instead of per-document hash-joins). Moreover, the TopX core engine has been refurbished in the form of a completely reimplemented prototype in C++, with carefully designed data-structures and our own cache management for inverted lists, also trying to keep the caching capabilities of modern CPU's in mind for finding appropriate data structures.

2 Scoring Model

We refer the reader to [10] for a thorough discussion of the scoring model, while we merely aim to briefly review the most important concepts here. Our XML-specific version of Okapi BM25 has proven to be mature and effective (and therefore remained unchanged, compare to, e.g., [3]) during four years of INEX Ad-hoc Track participations. It allows for the exact precomputation of fine-grained scores for the major building blocks of a CAS query—tag-term pairs in our case—with specific scores for each element-type/term combination. Computing individual weights for tag-term pairs introduces a certain factor of redundancy compared to a plain per-article/term scoring model, as each term occurrence is recursively propagated "upwards" the document tree, and its

frequency is then aggregated with more occurrences of the same term (thus treating each XML element in the classic IR notion of a document).

This volitional redundancy factor for materializing the scoring model (formally defined below) corresponds to about the average depth of a text node in the collection, while we aim to compensate the storage and I/O overhead through index compression, a novel feature for TopX 2.0. With the customized compression scheme described in Section 3, we clearly focus on good decoding performance, rather than on a maximum-possible compression rate that would possibly put an overly high load on the CPU at query processing time, in order to keep the overhead for the CPU low and the sequential throughput high when reading large, compressed blocks from disk. In the following, we distinguish between *content scores*, i.e., scores for the tag-term pairs of a query, and *structural scores* , i.e., scores assigned to additional navigational tags as they occur in longer path conditions or branching path queries.

2.1 Content Scores

For content scores, we make use of element-specific statistics that view the *full-content* of each XML element (i.e., the concatenation of all its descending text nodes in the entire subtree) as a bag of words:

1) the *full-content term frequency*, $ftf(t,n)$, of term t in node n, which is the number of occurrences of t in the full-content of n;
2) the *tag frequency*, N_A, of tag A, which is the number of nodes with tag A in the entire corpus;
3) the *element frequency*, $ef_A(t)$, of term t with regard to tag A, which is the number of nodes with tag A that contain t in their full-contents in the entire corpus.

The score of a tag-term pair of an element e with tag A with respect to a content condition of the form `//T[about(., t)]` (where `T` either matches A or is the tag wildcard operator $*$) is then computed by the following BM25-inspired formula:

$$score(e, \texttt{T[about(., t)]}) = \frac{(k_1+1)\ ftf(t,e)}{K + ftf(t,n)} \cdot \log\left(\frac{N_A - ef_A(t) + 0.5}{ef_A(t) + 0.5}\right)$$

$$\text{with} \quad K = k_1 \left((1-b) + b \frac{\sum_{t'} ftf(t',e)}{avg\{\sum_{t'} ftf(t',e') \mid e' \text{ with tag } A\}}\right)$$

We used the default values of $k_1 = 1.25$ and $b = 0.75$ as Okapi-specific tuning parameters (see also [4] for tuning Okapi BM25 on INEX). Note that our notion of tag-term pairs enforces a strict evaluation of the query conditions, i.e., only those elements whose tag matches the query tag get a non-zero score.

For a query content condition with multiple terms, the score of an element satisfying the tag constraint is computed as the sum of the element's content scores for the corresponding content conditions, i.e.:

$$score(e, \texttt{//T[about(., } t_1 \ldots t_m \texttt{)]}) = \sum_{i=1}^{m} score(e, \texttt{//T[about(., } t_i \texttt{)]})$$

Note that content-only (CO) queries have not been in the primary focus when defining this scoring function, but keyword conditions as sub-conditions of structural queries. CO queries are therefore evaluated with the tag wildcard operator "*" which matches any tag. The score for a "*"-term pair then is the same as the score for the original tag-term pair. Hence an XML document yields as many "*"-tag matches as there a distinct tags that also match the term condition, possibly with a different score each.

2.2 Structural Scores

Given a query with structural and content conditions, we transitively expand all structural query dependencies. For example, in the query `//A//B//C[about(., t)]` an element with tag `C` has to be a descendant of both `A` and `B` elements (where branching path expressions can be expressed analogously). This process yields a *directed acyclic query graph* with tag-term conditions as leaves, tag conditions as inner nodes, and all transitively expanded descendant relations as edges.

Our structural scoring model essentially counts the number of navigational (i.e., tag-only) conditions that are completely satisfied by the structure a candidate element and assigns a small and constant score mass c for every such tag condition that is matched. This structural score mass is then aggregated with the content scores, again using summation. In our INEX 2008 setup, we have set $c = 0.01$, whereas content scores are normalized to $[0, 1]$. That is, we emphasized the relative weights of content conditions much more than in previous INEX years, where we used a structural score mass of $c = 1.0$. This reflects both, effectiveness aspects for the Wikipedia collection and the new passage-based metrics, where structural constraints in CAS queries are even more of a mere hint than in the previous collections and years, as well as efficiency aspects, where unmatched structural conditions can more easily be compensated by high content-related score which leads to earlier pruning of result element with weak content-related matches.

3 Index Structures

Just like in the original TopX 1.0 prototype, our key for efficient query evaluation is a combined inverted index for XML full-text search that combines content-related and structural information of tag-term pairs, as well as a smaller structure-only index that lets us evaluate additional tag conditions of a CAS query. These two index structures directly capture the scoring model described in the previous Section 2. For both index structures, XML elements (matching a given tag-term pair or an individual tag, respectively) are grouped into so-called *element blocks* of elements that share the same tag, term, and document id. Novel for TopX 2.0 is a second level of nesting, where

element blocks are in turn grouped into larger, so-called *document blocks*, with element blocks being sorted in ascending order of their document id within each document block, which allows for efficient m-way merge-joins between document blocks when evaluating multi-dimensional queries. Top-k-style index pruning is further supported by sorting the document blocks in descending order of their maximum element-block score. By default, TopX uses a pre-/post-order/level-based labeling scheme [8, 2] to capture the structural information, which proved to be superior to, for-example, Dataguide-based techniques over heterogeneous collections such as INEX-Wikipedia and/or typical NEXI-style CAS queries that heavily make use of the descendant axis. Note that we may skip the $level$ information when using the NEXI-style descendant axis only, and we thus focus on $(pre, post, score)$ triplets in the following. In summary, TopX maintains two separate inverted files:

- the *content index* that stores, for each tag-term pair, all elements with that tag that contain the term, including their BM25-based relevance score and their pre- and post labels.
- the *structure index* that stores, for each tag, all elements with that tag, including their pre- and post label.

Both inverted files come with disk-based dictionaries to find the offset into the file where the entries for a tag-term pair or a tag start. Similarly to the scoring model described earlier, the focus when designing this index structure were CAS queries (in fact, more general XPath full-text queries with support for all 13 different XPath axes). For INEX, CO queries can be managed as a special case of the above indexing strategy, using a virtual "*" tag for each XML element that matches the term condition. Managing individual tag-term pairs as well as the "*"-term pairs *separately* within the same inverted file as materialized block-index however requires a replication and regrouping of all element blocks into their "*" element structure (resulting in a redundancy factor of roughly 2). Note that the structural index may naturally be skipped if we were to process CO queries only.

Besides these two inverted files, TopX provides additional disk-based dictionaries (with random access by document/element id) that are needed to post-process query results for creating the output format of INEX runs:

- the *document index* that stores, for each document id, document metadata such as its file id and its complete URL (used to identify the document in an INEX run).
- the *path index* that stores, for each element, a canonical XPath expression of that element within the document (used to identify the element in an INEX run).

All dictionaries are implemented as persistent hash tables that are read on-demand and then largely cached in-memory [9]. In the following we focus on the inverted index structures mentioned above.

3.1 Previous Relational Encoding

TopX uses tuples of the format *(tag, term, docid, pre, post, score, maxscore)* as basic schema for the content index, and $(docid, tag, pre, post)$ for the structure index, respectively. Note that $maxscore$ is the maximum *score* per document id $docid$, element id

pre, tag, and $term$, which is needed to enforce a respective grouping of tuples into the element block structure in a relational schema with sorted access in descending order of $maxscore$ [11]. To illustrate the redundancy that arises from a relational encoding of this multi-attribute schema, we quickly review the previous DB schema used in TopX 1.0 that is highly redundant in two orthogonal ways: first, because of the materialization of the scoring model discussed above, with a different score for each tag and term; and second, because of the frequent repetition of attributes (used as keys) required for a relational encoding. Here, a denormalized schema is necessary to implement efficient sorted (i.e., sequential) access to element blocks stored in descending order of their *maxscore* in a database table. In typical DBMS's, B^+-trees are the method of choice for such an index structure with precomputed blocks and sorted access to leaf nodes via linked lists (pointers to other leaf blocks). Note that some DBMS's support the use of so-called Index-Only-Tables (IOT's) to store the entire table directly as a B^+-tree. Assuming a simple 32-bit (i.e., 4-byte) based encoding of integers for the $docid$, tag,

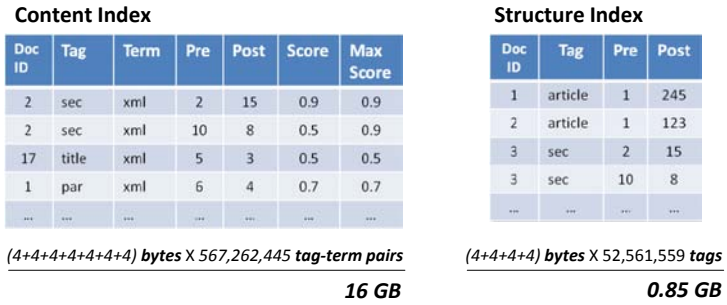


Fig. 1. Relational encoding and rough space consumption for TopX 1.0 on INEX-Wikipedia

$term$, pre, $post$ attributes and 32-bit floating-point numbers for $score$ and $maxscore$, we get a rough space consumption of 16 GB for the more than 560 million tag-term pairs and 0.85 GB for the more than 50 million tags we extract from INEX-Wikipedia, respectively (see also Section 5). This corresponds to an index blowup of a factor of about 4 compared to the 4.38 GB of XML sources for this collection.

Thus, a better approach is to keep the volitional redundancy that arises from our fine-grained scoring model but to encode this into a customized block structure that avoids the redundancy that is due to the relational encoding. These blocks are stored directly on disk in the spirit of a more compact object-oriented storage—however with very regular object structure. Our goal is to compress the index to a sensible extent, thus to save CPU time by keeping the fully precomputed scoring model over tag-term-pairs (with element-specific scores of terms), but to use moderate compression techniques to keep the index size at least similar to the overall size of the original XML data and thus to also spare I/O costs. We aim at highly efficient decompression techniques to spare valuable CPU time needed for top-k-style candidate queuing and non-conjunctive XPath evaluations during the actual TopX query processing. Also, the $maxscore$ attribute can be spared altogether in an object-oriented storage.

3.2 Content Index

TopX 2.0 maintains a single, binary inverted-file for content constraints, coined *content index*, whose structure is depicted in Figure 2. Here, the content index consists of individual index lists for two example tag-term pairs `//sec[about(.,''XML'')]` and `//title[about(.,''XML'')]`, along with their respective file offsets. These are further subdivided into histogram, document, and element blocks.

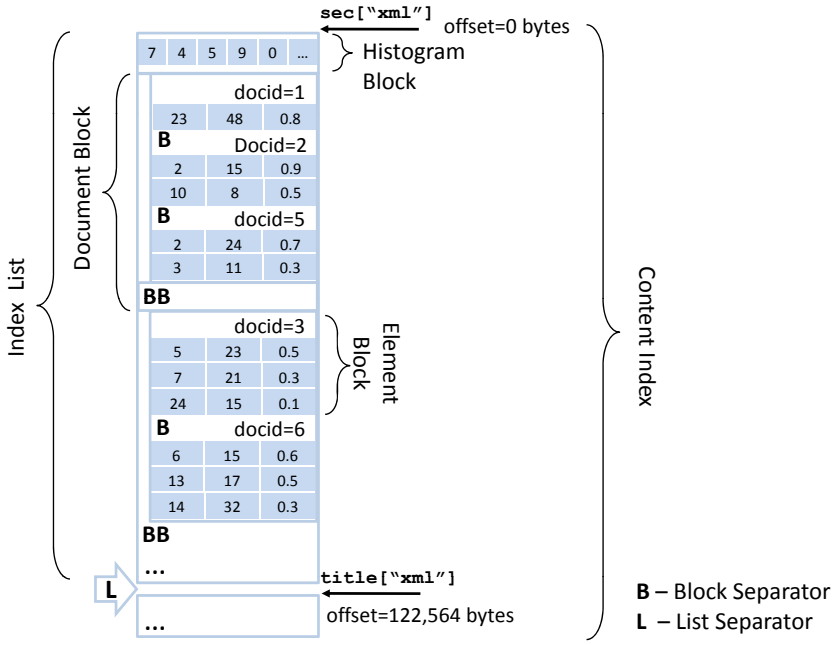


Fig. 2. Inverted and nested block structure of the content index for TopX 2.0

Index Lists. Index lists are the largest units in our index structure. For the structural index, an index list contains all tag-term pairs of elements contained in the collection (whereas for the content index, an index list contains all the elements' tags that occur in the collection). Sequential and random access to these inverted files is implemented by two auxiliary dictionaries each (see below). The physical end of an index list is marked by a special list separator byte `L` to prevent the engine from jumping into the next list when an index list has been entirely scanned.

Element Blocks. Each element block consists of a header, which contains the document's id ($docid$), and one or more entries of $(pre, post, score)$ triplets, where each triplet corresponds to one element within that document that contains the tag-term pair. For efficient evaluation of queries with structural constraints, each entry in an element block consists not only of the element's id (which corresponds to its pre value) and $score$, but also encodes the $post$ attribute, i.e., the entire information to locate and score

an element within the XML document. Element block sizes are data-dependent, i.e., an element block contains as many $(pre, post, score)$ triplets as there are XML elements in the document that match the term condition. The physical end of an element block is marked by a reserved separator byte `B` that indicates the beginning of the next document block.

Document Blocks. As opposed to element blocks, the size of a document block is a configurable parameter of the system and merely needs to be chosen larger than the size of the largest element block. The sequence of document blocks for a tag-term pair is constructed by first computing, for each element block, the maximal score of any element in that block. Within a document block, element blocks are then resorted by document id, to support efficient merge joins of other element blocks with the same document id (as opposed to the hash-joins needed in TopX 1.0). This sequence of element blocks is grouped into document blocks of up this fixed size, and the document blocks are then sorted by descending maximum element-block score, i.e., the maximum $maxscore$ among all element blocks they contain. Block sizes can be chosen generously large, typically in the order of 256–512KB, and block accesses are counted as a single I/O operation on disk. A document block ends as soon as adding another element block would exceed such a 256KB block boundary. The next document block then again contains element blocks with similar $maxscore$ sorted by descending document id $docid$. The physical end of document block is marked by an additional (second) separator byte `B`.

Histograms. To support probabilistic pruning techniques [12, 11, 10] and more sophisticated cost models for random access scheduling [1], the inverted file can include histograms that allow us to estimate, for a given tag-term pair, how many documents have a maximal score below or above a certain value. We use fixed-width histograms with a configurable number h of buckets. For a given tag-term pair, bucket i stores the number of documents whose maximal score is in the interval $[1 - \frac{i-1}{h}; 1 - \frac{i}{h}[$. bounds can be derived from h and i) for tag-term pairs with at least two document blocks, as histograms are only used to decide how many blocks should be read beyond the first block. For these tag-term pairs, the dictionary entry points to the beginning of the histogram instead of the first document block. (As most lists fit in a single document block, most lists do not have a histogram at their front. Lists with a histogram (for which we need to decode the histogram first) start with the separator byte `B` to distinguish a histogram block from a regular document block.

Auxiliary Access Structures. TopX maintains an additional, hash-based and persistent dictionary [9] to store the offset, for each tag-term pair, in the above binary inverted file, pointing to where the first document block in the inverted list for that tag-term pair starts for sorted access (SA). The dictionary itself is addressed via random access only, similarly to a hash-index in a DBMS, using a 64-bit hash of the tag-term condition as key and fetching the offset from the structural index as 64-bit value. This dictionary is usually small enough to fit mostly in main memory, allowing for a very efficient access to the file offsets.

A second such file-based dictionary can optionally be maintained to find the right offset for random accesses (RA) to document blocks, using 64-bit hashes of tag-term pair plus document id as key in this case. This dictionary returns the offset into the binary inverted file pointing to where the corresponding document block within the inverted list of that tag-term pair starts. Note that documents that occur in the first document block for a tag-term pair do not have to be kept in this index as the first block is always read with sorted access anyway and hence never is a candidate for random accesses. This keeps this second dictionary reasonably small, since the major amount of index lists does not exceed the first document block when the document block sizes are chosen reasonably large (e.g., 256 KB or more). To support both sequential and random access to both the structure and content indexes, we thus need four such dictionaries. Without random access scheduling, TopX can deal with as little as two compact dictionaries for accessing the inverted files.

3.3 Structure Index

TopX maintains a similar inverted file to store, for a given tag, all elements with that tag name, along with their pre and $post$ labels. The overall structure of this file, coined *structure index*, is similar to that of the inverted file for content constraints—a sequence of document blocks for each tag, which in turn consist of a sequence of element blocks corresponding elements to one document each. Each element block consists of one pre-/post-order entry for each element in the document with this tag. Note that there is no score of these entries, as scores for structural constraints are constant and can therefore be dropped from the index. No histogram headers are needed in this index structure for the same reason. In contrast to the content index, element blocks are simply stored in ascending order of document id. For processing CAS queries, structural index lookups are merely used as "soft filters" to aggregate additional score mass if a tag condition is matched.

For the sorted access (SA) entry points, a dictionary stores, for each tag, the offset to the first document block in the inverted file for this tag. Analogously to the content index, another optional dictionary can be used to also support random access (RA) to this structural index. This stores, for each tag and document id, the offset to the document block that contains the document's element block, if such a block exists.

3.4 CPU-friendly Compression

Switching from a relational encoding to a more object-oriented storage already reduces the index size down to about 8 GB—less than half the size needed for a relation schema but still almost twice as much as for the XML source files. Fast decompression speed and low CPU overhead is more important than a maximum-possible compression ratio for our setting. Moreover, with the large number of different attributes we need to encode into our block index, we do not assume a specific distribution of numbers, e.g., with most numbers being small, which rules out Huffman or Unary codes. We also intentionally avoid more costly compression schemes like PFor-Delta (compare to [13]) that need more than one scan over the compressed incoming byte stream (not to mention dictionary-based techniques like GZip and the a-like). We thus employ different

(simple) compression techniques for different parts of the index based on variations of delta and variable-length encodings requiring only linear scans over the incoming byte stream, thus touching each byte fetched from disk only exactly once.

Within each document block, there is a sequence of element blocks which are ordered by document id. Here, we exploit the order and store, instead of the document id of an element block, the delta to the document id of the previous block. This value is compressed with a variable-byte compression scheme. Within each element block, we first sort the entries by their pre value. We can now use delta encoding for the pre values to save storage space for smaller numbers. However, many element blocks contain just a single entry, so delta encoding alone is not sufficient. For most short documents, values of pre and $post$ attributes are small (i.e., they can be stored with one or two bytes), but there may still be some values which require three bytes. We therefore encode these values with a variable number of bytes and use an explicit length indicator byte to signal the number of bytes used for each entry. Scores on the other hand are 32-bit floating point numbers between 0 and 1, but storing them in this format would be far too precise for our needs. Instead, we designate a fixed number of bytes to store such a score and store, instead of a floating point value s, the corresponding integer value $\lfloor s \cdot 2^8 \rfloor$ (if we use a single byte), or $\lfloor s \cdot 2^{16} \rfloor$ if we use two bytes. These integer values are then stored with a variable number of bytes, where the actual number of bytes needed is again stored in the encoding byte of the entry.

To further reduce space consumption, we write each $(pre, post, score)$ triplet within an element block into a single code sequence, allowing us to use even less than one byte per attribute if the numbers are small. Using a fixed precision of only 6 bits for the $score$ attribute and up to 13 bits for the pre and $post$ attributes, we only need a single length-indicator byte per triplet. If pre and $post$ exceed $2^{13} = 8,196$, we can switch to two bytes per length indicator. Histogram entries are also stored as delta-encoded, variable-length integer numbers. They can be further compressed by cutting off the bucket sequence when only empty buckets with zero frequency follow.

3.5 Index Construction

The construction of the inverted files leverages well-known techniques from the construction of inverted files in text retrieval. The construction process operates in three different phases:

- *Parsing* the document collection, using a standard XML DOM parser, and generating a list of $(docid, tag, term, pre, post, ftf)$ tuples in a temporary file, which will later be used to create the content index. All ftf values are computed on-the-fly when a document is parsed. At the same time, a second intermediate text file is created with $(docid, tag, pre, post)$ tuples; this will later be transformed into the structure index. This step additionally fills the auxiliary document and path indexes.
- *Materializing* the scoring model after parsing is finished and element frequencies ef_A and average element lengths are known, the first intermediate file generated in the previous step, computes the score for each tuple and writes tuples of the form $(tag, term, docid, pre, post, depth, score)$ to another intermediate file. Here, the $score$ attribute already contains the final BM25-based content score.

– *Inverting* then consists of first *sorting* the intermediate file generated in the previous step in $(tag, term, docid)$ order (to keep elements from the same document together) and then, for each coherent tag-term list, generating the document and element blocks and entries in the corresponding dictionaries. Additionally, the structure index is created by sorting the second intermediate file from the first step by $(tag, docid)$, materializing appropriate document and element blocks, and adding the corresponding entries to the sorted and random access dictionaries.

4 Caching

In server mode, TopX 2.0 also comes with its own cache management, as opposed to the previous TopX 1.0 that could only rely on the underlying system's and DBMS's general caching strategies. Index lists (either top-k-style pruned index list prefixes, or even entire lists) are decoded, and the in-memory data structures are now explicitly kept as cache for the current and subsequent queries. Once read, they may be reused for arbitrary amounts of queries without the need to issue repeated I/O's to the same lists stored on disk. If TopX 2.0 is compiled and run in a 64-bit server environment, we may configure it to use generous amounts of memory for the cache. For our Efficiency Track experiments, we set the maximum cache size to up to 2,048 content and structure-related index lists, which led to maximum memory consumption of up to about 4 GB of main memory. For the caching itself, we currently employ a simple least-frequently-used (LFU) pruning strategy, which keeps the frequently used index lists in memory, such that for example the entire set of structural index lists used in the benchmark topics such as `article` or `section` quickly come completely from cache after a few queries are issued.

5 Experiments

5.1 Index Construction and Size

Indexing the INEX Wikipedia corpus [5] consisting of 659,388 XML documents with TopX yields about 567 million tag-term pairs and about 52 million tags (number of indexed elements) as depicted in Table 3. The average depth of an element is 6.72 which is a good indicator for the redundancy factor our scoring model involves. A comparison of storage sizes for the indexes is also depicted in Figure 3, while the uncompressed size of the corpus is 4.38 GB. We used no stemming for faster and more precise top-k runs. Stopwords were removed for the index construction.

All experiments in this paper were conducted on a quad-core AMD Opteron 2.6 GHz with 16 GB main memory, running 64-bit version of Windows Server. Caching was set to up to 2,048 index lists which resulted in a maximum main memory consumption of about 4 GB during the benchmark execution. All runtimes were measured over a hot system cache (i.e., after an initial execution of all queries), and then with varying usages of the TopX 2.0 internal cache (with a C suffix of runs ids indicating that the internal cache was cleared after each query, and with W indicating that the

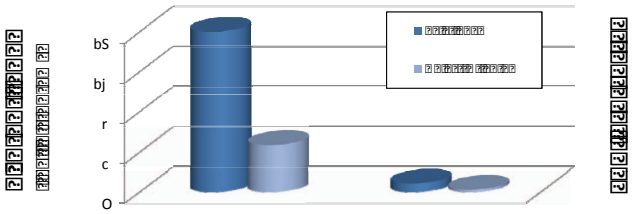


Fig. 3. Comparison of storage space needed for a relational encoding (TopX 1.0) and a more object-oriented, compressed encoding (TopX 2.0), in GB

	Content Index	Structure Index
# index objects	567,262,445 (overall tag-term pairs)	52,561,559 (overall tags)
# index lists	20,810,942 (distinct tag-term pairs)	1,107 (distinct tags)
# document blocks	20,815,884	2,323
# element blocks	456,466,649	8,999,193
index size (relational, uncompressed)	16 GB	0.85 GB
index size (object-oriented, uncompressed)	8.6 GB	0.43 GB
index size (object-oriented, compressed)	3.47 GB	0.23 GB
size of dictionary (SA)	0.55 GB	176 KB
size of dictionary (RA, optional)	2.24 GB	0.27 GB

Table 1. Statistics for the content and structure indexes

cache was kept and managed in a LFU manner). Each benchmark run however started with an empty internal cache.

The size of the RA dictionary is listed here only for completeness, as random-access scheduling has no longer been used for the TopX 2.0 experiments. Note that TopX 1.0 heavily relied on random access scheduling to accelerate candidate pruning because of the relatively bad sequential throughput when using a relational back-end, as compared to the relatively good random-access caching capabilities of most DBMS's, which makes random access scheduling more attractive over a relational back-end. With plain disk-based storage, we decided to spare random accesses altogether and save the storage-overhead needed for the random-access structures (typically leading to another redundancy factor of 2).

Index construction (see Section 3.5) still is a rather costly process, taking about 20 hours over INEX-Wikipedia to fully materialize the above index structure from the XML source collection—due to our heavy pre-computations but to the benefit of extremely fast query response times. With our simple yet sufficiently effective compression scheme, we achieve a compression factor of more than 2 over an uncompressed storage, which helps us keep the index size in the same order as the original data, in spite of materializing our highly redundant scoring model. Altogether, we obtain a factor of 4 less storage space compared to an uncompressed relational encoding.

5.2 Summary of Runs

Table 5.2 summarizes all Efficiency Track runs, using various combinations of Focused vs. Thorough, CO vs. CAS, as well as top-k points of $k = 15,\ 150,\ 1,500$, the latter being the original result size demanded by the Ad-Hoc track. Run names encode the

following parameter space:

`[<Engine>-<Track>-<CO/CAS>-<Top-k>-<Overlap mode>-<(W)arm/(C)old internal cache>]`

Run ID	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP	AVG MS.	SUM MS.	#Topics
Focused								
TOPX2-Eff08-CO-15-Focused-W	0.4751	0.4123	0.2793	0.1971	0.0726	49.79	28,180	566
TOPX2-Eff08-CO-150-Focused-W	0.4955	0.4520	0.3674	0.3114	0.1225	85.96	48,653	566
TOPX2-Eff08-CO-1500-Focused-W	0.4994	0.4560	0.3749	0.3298	0.1409	239.73	135,688	566
TOPX2-Eff08-CAS-15-Focused-W	0.4587	0.3878	0.2592	0.1918	0.0662	90.99	51,499	566
TOPX2-Eff08-CAS-150-Focused-W	0.4747	0.4282	0.3494	0.2915	0.1094	112.32	63,574	566
TOPX2-Eff08-CAS-1500-Focused-W	0.4824	0.4360	0.3572	0.3103	0.1241	253.42	143,436	566
Thorough								
TOPX2-Eff07-CO-15-Thorough-W	n/a	n/a	n/a	n/a	n/a	70.91	40,133	566
TOPX2-Eff07-CAS-15-Thorough-W	n/a	n/a	n/a	n/a	n/a	89.31	50,549	566
Focused (cold internal cache)								
TOPX2-Eff07-CO-15-Focused-C	0.4729	0.4155	0.2795	0.1979	0.0723	51.65	29,234	566
TOPX2-Eff07-CAS-15-Focused-C	0.4554	0.3853	0.2583	0.1905	0.0655	96.22	54,461	566

Table 2. Effectiveness vs. efficiency summary of all TopX 2.0 runs

The iP and MAiP effectiveness measures reflect the 308 Ad-Hoc topics from INEX 2006–2008 for which assessments were readily available. Only 566 out of 568 topics were processed due to a rewriting problem that led to empty results for two of the topics. We generally observe a very good early precision at the lower recall points, which is an excellent behavior for a top-k engine. Compared to overall results from the Ad-Hoc Track, we however achieve lower recall at the top-1,500 compared to the best participants (also due to not using stemming and strictly evaluating the target element of CAS queries). TopX 2.0 however shows an excellent runtime behavior of merely 49.70 ms. average runtime per CO and 90.99 ms. average runtime per CAS query. Also, starting each query with a cold internal cache (but warm system cache) instead of a warm internal cache consistently shows about 10 percent decrease in runtime performance for both the CO and CAS modes.

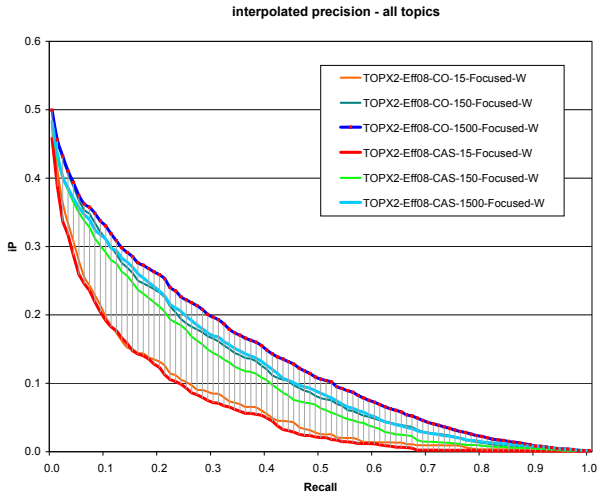


Fig. 4. Interpolated precision plots of all TopX 2.0 Efficiency Track runs

5.3 Efficiency Runs By Topic Type

Tables 5.3–5.3 depict the TopX 2.0 performance summaries grouped by topic type. We see a very good runtime result of only 18.88 ms. average processing time for CO topics and 48.84 ms. on average per CAS query for the type (A) (i.e., classic Ad-Hoc topics) and top-15 runs. Also, CO retrieval does not only seem to be more efficient but also more effective than a respective CAS mode, with a maximum MAiP value of 0.14 for CO compared to 0.12 for CAS (returning the top-1,500 elements in Focused mode). As expected, there is a serious runtime increase for type (B) topics, with up to 112 query dimensions, but only comparably little additional overhead for larger values of k, as runtimes are dominated by merging the huge amount of index lists here (in fact most top-k approaches seem to degenerate for high-dimensional queries). Type (C) topics were very fast for the small value of $k = 15$ but then showed a high overhead for the larger values of k, i.e., when returning the top-150 and top-1,500. As no assessments were available for the 7 type (C) (structure-enhanced) topics, the respective effectiveness fields are left blank.

Run ID	MAiP	AVG MS.	SUM MS.	#Topics
TOPX2-Eff08-CO-15-Focused-W	0.0712	18.88	10,157	538
TOPX2-Eff08-CO-150-Focused-W	0.1234	49.12	26,427	538
TOPX2-Eff08-CO-1500-Focused-W	0.1430	191.27	102,903	538
TOPX2-Eff08-CAS-15-Focused-W	0.0643	48.84	26,276	538
TOPX2-Eff08-CAS-150-Focused-W	0.1094	61.25	32,953	538
TOPX2-Eff08-CAS-1500-Focused-W	0.1249	165.53	89,055	538

Table 3. Summary of all TopX 2.0 Focused runs over 538 type (A) topics

Run ID	MAiP	AVG MS.	SUM MS.	#Topics
TOPX2-Eff08-CO-15-Focused-W	0.0915	844.67	17,738	21
TOPX2-Eff08-CO-150-Focused-W	0.1094	1038.90	21,817	21
TOPX2-Eff08-CO-1500-Focused-W	0.1125	1468.67	30,842	21
TOPX2-Eff08-CAS-15-Focused-W	0.0915	1044.71	21,939	21
TOPX2-Eff08-CAS-150-Focused-W	0.1096	1074.66	22,568	21
TOPX2-Eff08-CAS-1500-Focused-W	0.1124	1479.33	31,066	21

Table 4. Summary of all TopX 2.0 Focused runs over 21 type (B) topics

Run ID	MAiP	AVG MS.	SUM MS.	#Topics
TOPX2-Eff08-CO-15-Focused-W	n/a	41.00	287	7
TOPX2-Eff08-CO-150-Focused-W	n/a	58.86	412	7
TOPX2-Eff08-CO-1500-Focused-W	n/a	277.57	1,943	7
TOPX2-Eff08-CAS-15-Focused-W	n/a	469.42	3,286	7
TOPX2-Eff08-CAS-150-Focused-W	n/a	1150.14	8,051	7
TOPX2-Eff08-CAS-1500-Focused-W	n/a	3330.71	23,315	7

Table 5. Summary of all TopX 2.0 Focused runs over 7 type (C) topics

6 Conclusions and Future Work

This paper introduces our new index structure for the TopX 2.0 prototype and its initial evaluation on the INEX 2008 Efficiency Track. TopX 2.0 demonstrates a very good allround performance, with an excellent runtime for keyword-oriented CO and typical CAS queries and still good runtimes for very high-dimensional content (type B) and

strucural (type C) query expansions. Overall we believe that our experiments demonstrate the best runtimes reported in INEX so far, while we are able to show that this performance does not have to be at the cost of retrieval effectiveness. The scoring model and non-conjunctive query evaluation algorithms remained unchanged compared to TopX 1.0, which also managed to achieve ranks 3 and 4 in the Focused Task of the 2008 Ad-hoc Track.

Our future work may investigate improved caching strategies, consider incremental index updates (thus leave space in blocks, similar to DB pages), and expand into even more features, such as incremental query expansion techniques not yet taken over from TopX 1.0, and more XQuery constructs beyond just XPath 2.0. Just like the previous Java-based prototype, we intend to provide an open-source release of the new C++ based TopX 2.0 soon.

References

1. H. Bast, D. Majumdar, M. Theobald, R. Schenkel, and G. Weikum. IO-Top-k: Index-optimized top-k query processing. In *VLDB*, pages 475–486, 2006.
2. P. A. Boncz, T. Grust, M. van Keulen, S. Manegold, J. Rittinger, and J. Teubner. MonetDB/XQuery: a fast XQuery processor powered by a relational engine. In S. Chaudhuri, V. Hristidis, and N. Polyzotis, editors, *SIGMOD Conference*, pages 479–490. ACM, 2006.
3. A. Broschart, R. Schenkel, M. Theobald, and G. Weikum. TopX@INEX2007. In N. Fuhr, J. Kamps, M. Lalmas, and A. Trotman, editors, *INEX*, volume 4862 of *Lecture Notes in Computer Science*, pages 49–56. Springer, 2007.
4. 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.
5. L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 2006.
6. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*. ACM, 2001.
7. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.
8. T. Grust. Accelerating XPath location steps. In M. J. Franklin, B. Moon, and A. Ailamaki, editors, *SIGMOD Conference*, pages 109–120. ACM, 2002.
9. S. Helmer, T. Neumann, and G. Moerkotte. A robust scheme for multilevel extendible hashing. In *Computer and Information Sciences - 18th International Symposium (ISCIS)*, pages 220–227, 2003.
10. 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.
11. 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.
12. M. Theobald, G. Weikum, and R. Schenkel. Top-k query evaluation with probabilistic guarantees. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *VLDB*, pages 648–659. Morgan Kaufmann, 2004.
13. J. Zhang, X. Long, and T. Suel. Performance of compressed inverted list caching in search engines. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 387–396, New York, NY, USA, 2008. ACM.

Recognition of Structural Similarity to Increase Performance

Judith Winter[1] and Nikolay Jeliazkov[1]

[1] J.W.Goethe University, Department of Computer Science, Frankfurt, Germany
winter@tm.informatik.uni-frankfurt.de, nikolay.jeliazkov@gmail.com

Abstract. As researchers focusing on distributed Information Retrieval (IR), we regard good performance not only as achieving high precision but also as considering efficiency issues such as the network traffic produced to answer a given query. Therefore, the new Efficiency Track of INEX is of high interest for us. We have taken the opportunity to run some of its highly structured queries on our top-k search engine to investigate whether we can improve performance by recognizing and exploiting structural similarity. We have analyzed different structural similarity functions and applied them to both ranking and routing.

Keywords: XML Retrieval, Structural Similarity, Distributed Search, INEX, Efficiency, Peer-to-Peer

1 Introduction and Motivation

Currently, most systems participating at INEX aim at effectiveness in terms of precision and recall. Efficiency is rarely considered and not rewarded by the current INEX measures. In our group, research is focused on distributed IR solutions where good performance includes both effectiveness and efficiency. The new Efficiency Track is therefore of high interest for us, especially after the cancelation of the heterogeneous track. There are several practical and theoretical reasons that motivated us to participate in the Efficiency Track. Firstly, our system is based on a Peer-to-Peer (P2P) network such that we have to consider the network traffic between peers. We cannot send long posting lists but have to prune them which corresponds to the particular encouragement of top-k style search engines in the Efficiency Track. Secondly, our system exploits structural user hints to improve effectiveness in the ranking process and to improve efficiency in the routing process. Contrary to INEX's ad-hoc topics, which consist of many CO and poorly structured CAS queries, the high-dimensional structured type-C queries of the efficiency track offer an excellent opportunity to test more sophisticated structural similarity functions. Thirdly, many ad-hoc participants tune their systems for the Wikipedia collection whereas most P2P systems are faced with heterogeneous collections, not based on the same schema and with great variety regarding content and structure. Thus, a more database-oriented view of INEX would be in our interest, e.g. to discuss the application of schema-mapping methods in XML-retrieval. Finally, efficiency issues such as reducing the

amount of messages between peers are of major concern in P2P-IR. The opportunity to discuss these challenges with other participants interested in efficiency issues is highly appreciated, e.g. to analyze how XML-structure can help.

This paper presents a comparison of different structural similarity functions in order to take advantage of richly structured CAS queries for improving effectiveness and efficiency. The top-k search engine Spirix [6], based on a P2P network, has been used for the runs.

2 Spectrum of Structural Similarity

There is ongoing research in the field of recognizing similarity between structures. Existing solutions propose different strategies and functions to calculate the similarity between the given structural conditions in a query and the structures found in a collection. According to [2], these strategies can be divided into four groups depending on the thoroughness they achieve in analyzing the similarity: *perfect match*, *partial match*, *fuzzy match*, and *baseline (flat)*. In the first case, only exact matching structures are considered, thus only retrieving relevant documents with the search terms found in the exact XML context as specified by the user. In the case of the *partial match* strategies, one of the compared structures has to be a sub-sequence of the other and the overlapping ratio is measured. The *fuzzy match* type of functions takes into account gaps or wrong sequences of the different tags. With the *baseline* strategy the specified XML structures are ignored, thus resulting in a conventional IR technique and losing the possible benefits of a XML-structured documents.

As we observe later in this paper, the *perfect match* strategies usually result in a reduced precision, because a large amount of relevant text is ignored due to slight differences in the structures. A comparison between the other extreme (*baseline*) and the two types of similarity functions that take structure into account (*partial* and *fuzzy match*) is an important research topic – can XML structure help to improve performance?

We analyzed several functions as representatives of the mentioned types of strategies. A formula which determines whether the query or the found structure is a sub-sequence of the other one and then measures the overlapping ratio between them (*partial match*) is proposed in [2]. The group of the *fuzzy match* type of similarity functions performs deeper analysis of the considered structures and noticeably expands the flexibility and the possibility of a precise ranking of all found structures for a search term according to their similarity to the query structure. In order to achieve this, [1] defines an algebra for calculating and assessing the different operations and their costs for transforming an XML tree specified by the query into the XML tree of the found structure. [5] ignores the tree representation and handles the structures as sequences of tags. Nevertheless, it uses the same strategy of counting the costs for transforming one structure into another. It is based on the *Levenstein Edit Distance* [4], a method used 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.

Another approach for the similarity analysis of XML structures falling in the scope of the *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 [3], divided in two major groups - *semantic* and *structural*. The first group consists of the factors *semantic completeness (SmCm)*, measured by the ratio between the found query tags, and the total amount of tags in the query structure, and *semantic correctness (SmCr)*, measured as a function of all semantic similarities between the tags in the query and the target structures. Within the *structural* group, three factors are distinguished in [3]. The *structural completeness (StCm)* represents the overall coverage of the query XML tree by the target tree - how many of the wanted hierarchical parent-child relationships between the tags are satisfied by an analogous pair in the found structure. The *structural correctness (StCr)* is computed as the complement of the amount of found but reversed hierarchical pairs in respect to all found pairs. The last factor, the *structural cohesion (StCh)*, represents the deviation of the found XML structure from the query and is calculated by the complement of the ratio between the non-relevant tags and the total amount of tags in the target.

3 Recognition of Structural Similarity

Can detecting structural similarity help to improve IR performance? We analyzed several different types of functions, applied appropriate enhancements and evaluated them with the INEX measures by using the Wikipedia document collection and a selection of topics with structural conditions.

We developed the following formula based on [2] as a representative of the *partial match* type of strategies:

$$Sim_1(s_q, s_{ru}) = \begin{cases} \left(\frac{1+|s_q|}{1+|s_{ru}|}\right)^{\alpha}, & \text{if } s_q \text{ is sub-sequence of } s_{ru} \\ \beta \cdot Sim_1(s_{ru}, s_q), & \text{if } s_{ru} \text{ is sub-sequence of } s_q \\ 0, & \text{else} \end{cases} \quad \textbf{(Sim1)}$$

s_q represents the structural condition in the query and s_{ru} 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, thus achieving a greater flexibility for the user. For example, <*author*> and <*writer*> are rather similar, and a precise similarity value can be designated to these tags. We are considering the possibility of giving the user the opportunity to define such similarities himself.

As a representative of the class of functions based on cost calculation for transforming the query structure into the target one, we used the method proposed in [5] **(Sim2)**. We implemented and evaluated this method with a suitable normalization and, as above, an enhancement with a tag dictionary was also applied.

The five similarity factors from [3] promised 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 with methods based on [4]. 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:

$$Sim_3 = \alpha \cdot \left(\frac{\omega_1 SmCm + \omega_2 SmCr}{2} \right) + \beta \cdot \left(\frac{\omega_3 StCm + \omega_4 StCr + \omega_5 StCh}{3} \right)$$

(Sim3)

All similarity factors were normalized, and parameter boundaries were set as follows: $\alpha + \beta \leq 1$, $\omega_1 + \omega_2 \leq 2$, and $\omega_3 + \omega_4 + \omega_5 \leq 3$. This assures 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 same applies for the five parameters ω_i, which are 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.

We designed an appropriate index in order to support these strategies in the context of P2P networks. 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 structures of a specific term. In order to reduce the variety of structures, several methods were tried, such as stemming of tags and removal of stopword tags (e.g. conversionwarnings).

3 Evaluation

All similarity functions were implemented as a part of our P2P-based search engine Spirix [6]. The hardware used was a Linux system with 8x2GHz Intel Xeon CPU and 16GB of RAM. The retrieval time has been approximately 253 seconds per topic but other processes used the machine at the same time. The ranking is based on an adaption of BM25E.

Of all Efficiency Track topics, we chose 80 richly structured topics from the INEX 2007 ad-hoc track, which are part of the type-A topics in 2008, and all type-C topics as those are high-dimensional structured queries. However, the type-C topics were not assessed and have therefore not been officially evaluated.

Our runs aimed at comparing the three proposed structural similarity functions with a baseline run, where no structure was used, as well as with a run where only perfect matches were considered. We first compared the functions when using them for ranking, i.e. aimed at improved effectiveness by using structural hints. Table 1 shows preliminary evaluation with the INEX 2007 tools (focused task). Secondly, we applied the best performing similarity function to improve efficiency during routing and retrieval.

Table 1. Precision at different recall-levels for the compared similarity functions.

	baseline	Sim1	Sim2	Sim3	perfect match
iP[0.00]	0,3241	0,3507	0,3534	0,3300	0,0151
iP[0.01]	0,3134	0,3305	0,3364	0,3195	0,0151
iP[0.05]	0,2909	0,2999	0,3001	0,2886	0,0151
iP[0.10]	0,2771	0,2838	0,2870	0,2746	0,0099
iP[0.30]	0,1911	0,1892	0,1760	0,1819	0,0063
MAiP	0,1400	0,1321	0,1252	0,1278	0,0038

The *perfect match* case resulted in very low retrieval quality, as the most relevant documents were disqualified in the ranking process due to slight structural differences. Thus, this strict handling of the structural hints specified by the user can only be advantageous in database-oriented approaches and is of no use in the context of IR. At recall-level iP[0.01], an increase in precision can be observed for every similarity function compared to the baseline. For higher recall-levels (above iP[0.30]), this advantage disappears and the baseline strategy shows a better precision than any other case, resulting in its higher mean average interpolated precision (*MAiP*). Nevertheless, most users are interested in early precision only – for which the use of similarity functions can lead to an improvement.

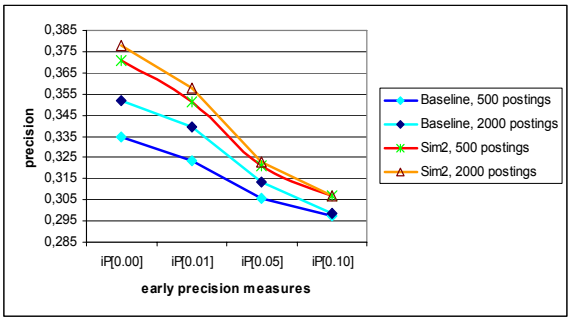


Fig. 1. Using structure to improve routing and reduce the size of posting lists.

After improving effectiveness, we took the best performing structure function ***Sim2*** and used it to investigate its effect on the routing process (this run was not submitted to the Efficiency Track but is based on the results in table 1). The postings (500 respective 2000 postings) were selected using BM25E for the baseline run. In the better performing run ***Sim2***, the postings were selected by an impact factor proportional to the product of a BM25E-based weight and the calculated structural similarity. Figure 1 shows how calculating structural similarity can improve efficiency by helping to select the adequate postings. For example, at iP[0.01] we achieve a better precision (0,3580) by using ***Sim2***, even if we select only 500 postings instead of 2000 for the *baseline*. Thus, we can save more than 1500 postings.

Based on these observations, we are currently evaluating the effect of using structure for runs with #postings $\in$ {10, 50, 100…unlimited} to determine the point where additional postings increase network traffic but do not lead to further improvement of precision.

4 Discussion

In this paper, we discussed our participation at the new Efficiency Track of INEX 2008. We have welcomed the possibility of using higher dimensioned structural hints, like the type-C topics, in order to retrieve more relevant and precise results, especially in the context of a convergence between IR-based and database-oriented approaches, where advantage can be taken of additional information such as XML-structure to achieve good performance. Thus, the newly started Efficiency Track offers new possibilities to continue the research in this field.

We have presented different structural similarity functions, analyzed them, and showed that – for the selected structured topics – an improvement of performance could be achieved during ranking and routing. This evaluation is an indication that structural hints can help indeed. In order to confirm the generality of this observation, we will extend our research further. For this purpose, we will need more highly-structured topics which should be more carefully constructed than the current ones. Additionally, we think that the Wikipedia collection lacks the variety of needed rich semantic structures.

References

[1] Al-Khalifa, S.; Yu, C.; Jagadish, H.V.: ***Querying Structured Text in an XML Database***. SIGMOD 2003, June 9-12, San Diego, CA, USA (2003)

[2] Carmel, D.; Maarek, Y.; Mandelbrod; Mass, Y.; Soffer: ***Searching XML Documents via XML Fragments***. In: Proc. of the 26th Int. ACM SIGIR, Toronto, Canada (2003)

[3] Ciaccia, P.; Penzo, W.: ***Adding Flexibility to Strucuture Similarity Queries on XML Data***. In: T. Andreasen et al. (Eds.): FQAS 2002, LNAI 2522, Seiten 124-139, Springer-Verlag, Berlin Heidelberg (2002)

[4] Levenshtein, V. I.: ***Binary codes capable of correcting deletions, insertions, and reversals***. Soviet Physics Doklady 10:707-710 (1966)

[5] Vinson, A.; Heuser, C.; Da Silva, A.; De Moura, E.: ***An Approach to XML Path Matching***. WIDM'07, November 9, Lisboa, Portugal (2007)

[6] Winter, J.; Drobnik, O.: ***University of Frankfurt at INEX2008 – An Approach For Distributed XML-Retrieval*** . In: Preproceedings of the 7th International Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2008), Dagstuhl, Germany (2008)

DRAFT - Overview of the INEX 2008 Entity Ranking Track

Gianluca Demartini[2], Arjen de Vries[1] Tereza Iofciu[2], and Jianhan Zhu[3]

[1] CWI & Delft University of Technology
The Netherlands
arjen@acm.org
[2] L3S Research Center
Leibniz Universität Hannover
Appelstrasse 9a D-30167 Hannover, Germany
{demartini,iofciu}@L3S.de
[3] University College London
Adastral Park Campus
Ipswich, IP5 3RE, UK
j.zhu@adastral.ucl.ac.uk

Abstract. This *DRAFT* overview of the INEX 2008 Entity Ranking track mainly explains the construction of the pools through sampling, and the underlying motivation. The final version will present more details about the various runs.

1 Introduction

Many user tasks would be simplified if search engines would support typed search, and return entities instead of just web pages. In 2007, INEX has started the 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 and entity list completion, 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 will continue to run both the entity ranking and list completion tasks this year. In addition, we propose a new entity relation search (ERS) task investigating how well systems can not only find entities relevant to a topic but also establish correct relations between entities.

2 INEX-XER Setup

2.1 Setup

The track uses the Wikipedia XML data, where we exploit the category metadata about the pages to loosely define 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 models for entity ranking should handle the situation that the category assignments to Wikipedia pages are not always consistent, and also far from complete. The human assessor is of course not constrained by the category assignments made in the corpus when making his or her relevance assessments!

We expect that the data set provides a su! ciently useful collection as a starting point for the purpose of the track. The challenge for participants is to exploit the rich information from text, structure and links to perform the search tasks.

2.2 Tasks

XML Entity Ranking (XER) concerns triples of type `<query, category, entity>`. 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 [4]).

The motivation of the new entity relation search (ERS) task is that searchers are not satisfied with a list of relevant entities to a query anymore, because they would like to know more details about these entities, such as their relations with other entities, and their attributes. One example is find museums in the Netherlands exhibiting Van Goghs artworks, and the cities where these museums are located. A system needs to first find a number of relevant museums, and then establish correct correspondence between each museum and a city. We expect the new ERS task will help explore new research areas in information retrieval with potential connections with other research areas such as information extraction, social network analysis, natural language processing, semantic web, and question answering.

We divide ERS into an entity ranking phase followed by a relation search phase. ERS concerns tuples of type `<query, category, entity, relation-query, target-category, target-entity>`. The query, category, and entity are already defined in the entity ranking task. The relation-query in form of free text describes the relation between an entity and a target entity. The target-category specifies the type of the target entity. Target-entity specifies example instances of the target entity type.

[4] `http://inex.is.informatik.uni-duisburg.de/2007/linkwiki.html`

2.3 Topics

Participants from eleven institutions have created a small number of (partial) entity lists with corresponding topic text. Candidate entities correspond to the names of articles that loosely belong to categories (for example may be subcategory) in the Wikipedia XML corpus. As a general guideline, the topic title should be type explanatory, a human assessor should be able to understand from the title what the category type needed is. Some of the topics have been extended for the ERS pilot task.

2.4 The 2008 test collection

We received 33 runs submitted by six participants. The pools have been based on all submitted runs, using a sampling strategy detailed in the next Section.

3 Investigation on Sampling strategies

In INEX-XER 2008 we decided to use sampling strategies for generating pools of documents for relevance assessments. The main two reasons why we decided to introduce sampling are to reduce the judging effort and to include into the pools also documents from higher ranks.

The first aspect we have to analyze is how the comparative performances of systems is affected while we perform less relevance judgements. We used the 2007 INEX-XER collection simulating the situation of performing less relevance judgements. We compared three different sampling strategies, that is, a uniform random sampling from the top 50 documents retrieved by the IRSs, a sampling based on the relevance distribution among the different ranks, and a stratified sampling with strata manually defined by looking at the distribution of relevance from the previous year.

For the experimental comparison of the three different sampling approaches we used the 24 'genuine' INEX-XER topics from the 2007 collection. As only data from 2007 has been used, we used the leave-one-out approach for simulating the approach of learning from the previous year data. That is, we considered all the topics but one as previous year data. In these topics the relevance assessments and, therefore, the relevance distribution over the ranks is known. In the topics not considered for learning we assume not to know the relevance assessment (that is, a topic from the current year) and we compare the original IRSs ranking on this topic with the one produced by the sampling. We repeat this for all the 24 topics in the collection.

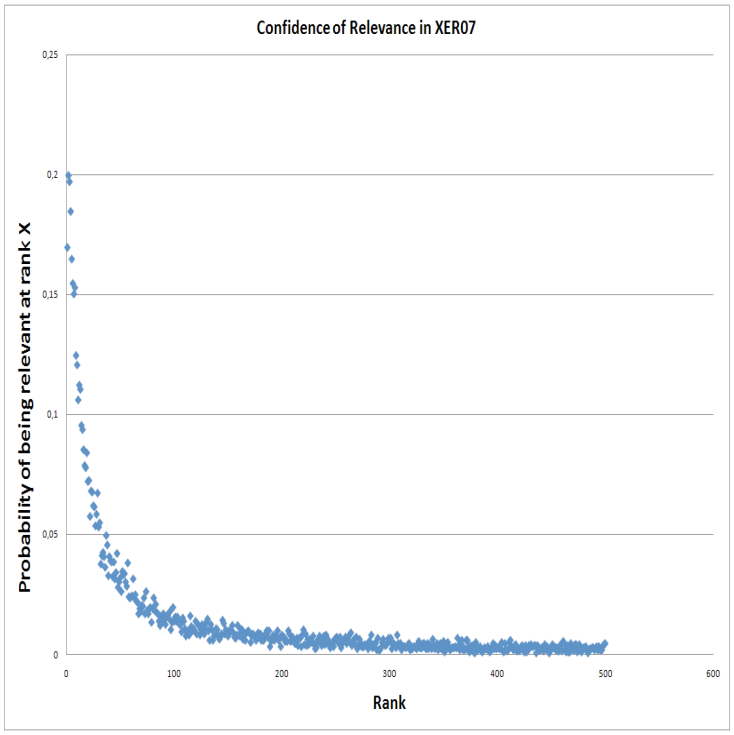
3.1 Uniform Random Sampling

The first approach we decided to investigate is a Uniform Random Sampling of retrieved documents. In order to do so, we first randomly selected some ranks at which to take documents from the runs. Then, we considered only the assessments on those documents for ranking the systems assuming that the other

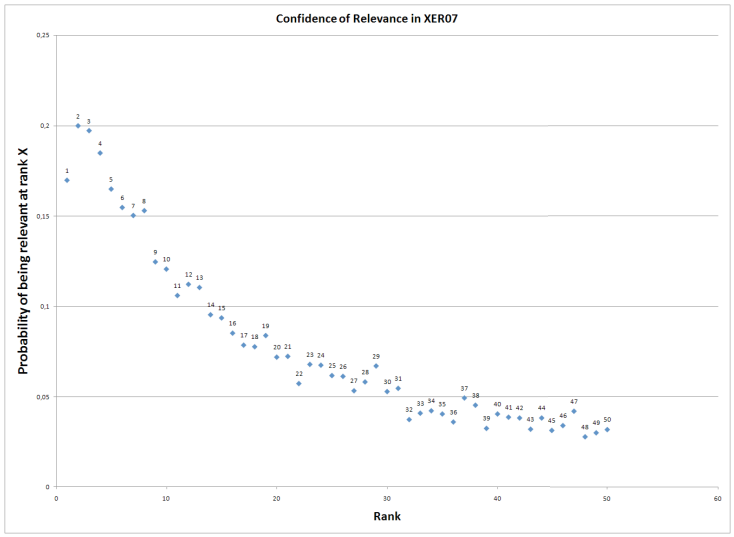
entities were not judged and, therefore, not relevant. Finally, we measured the correlation with the original system ranking using the 24 XER topics from 2007. The correlation values are presented in Figure 3.2.

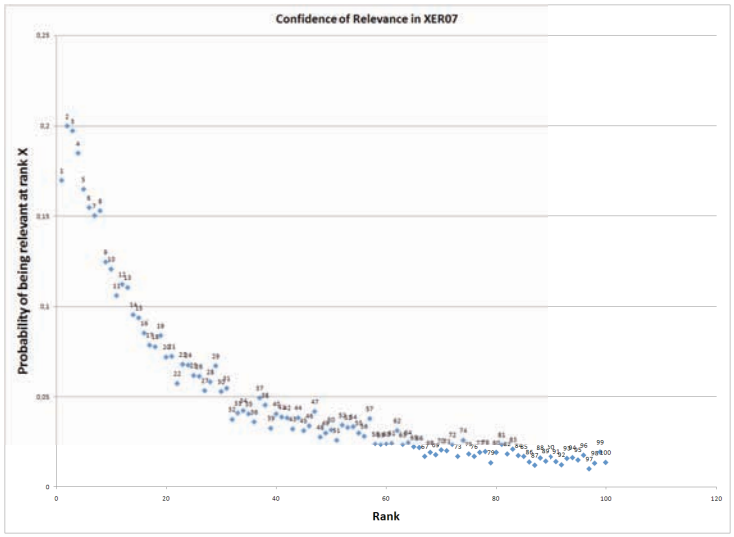
3.2 Relevance based Random Sampling

In order to perform a sampling with a higher chance of selecting relevant documents into the pools, we used the distribution of relevance over ranks and learned from the 2007 data the probability of finding a relevant entity at each rank (up to 50 as the depth of 2007 pool) in the runs. We then sampled the 2008 runs using such probability distribution. The relevance distribution in 2007 for ranks up to 50 is displayed in Figure 3.2. In Figure 3.2 it is possible to see the curve for all the ranks.

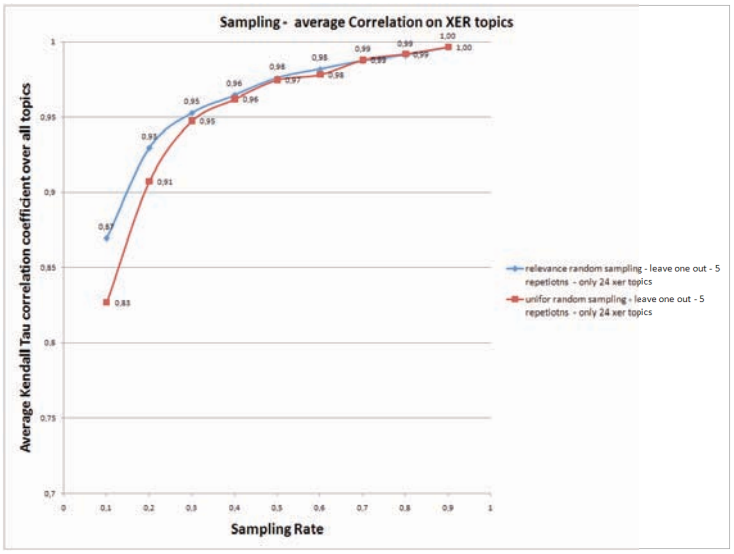


For evaluating this technique and comparing it with the random sampling we proceed as follows. As 2007 was the first edition of INEX-XER, we can not learn the relevance distribution from a previous year and test it on the following and, of course, we can not learn the distribution and test the effectiveness on the same data. Therefore, in our experiments the system learns the probabilities


Confidence of Relevance in XER07
Probability of being relevant at rank X
Rank
0,25
0,2
0,15
0,1
0,05
0
0
10
20
30
40
50
60


Confidence of Relevance in XER07
Probability of being relevant at rank X
Rank
0,25
0,2
0,15
0,1
0,05
0
0
20
40
60
80
100
120

using all the topics but one. This distribution is used for generating a random sample (based on such probabilities) of documents from the runs. The systems' ranking on the remaining topic not used for learning is then computed, and it is compared with the original ranking for that topic. This process is iterated over all topics and the average correlation value is taken. Figure 3.2 shows the correlation between the original system ranking and the ranking derived from either the relevance based or the uniform random sampling.



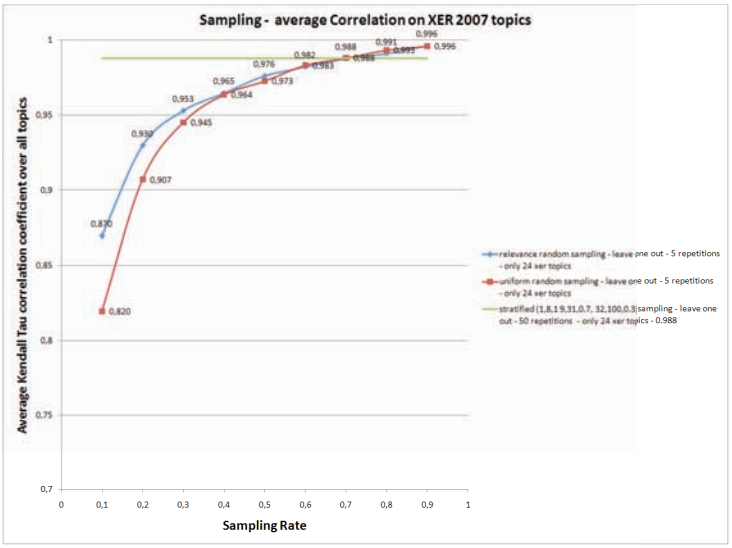
3.3 Stratified Sampling

A third option to performing a sampling in order to construct pools for relevance assessment is the stratified approach which aims at including in the pools a big number of relevant results. The idea is to perform sampling within each stratum independently of the other. In this way it is possible to sample more documents in higher strata and less from strata which are down in the ranking. There is then the need to optimally selecting the strata and the sampling percentage for each strata, which is an open problem. Using stratified sampling also allows to compute as evaluation metrics *xinfAP* [1] which is a better estimate of AP in the case of incomplete assessments.

Considering the results shown in Figure 3.2, we decided to use the following strata for the pool construction of INEX-XER 2008:

- 1,8 100%
- 9,31 70%
- 32,100 30%

This means that we include in the pool 45 documents from each run. In order to compare this approach with the ones presented above, we computed the correlation using the strata-based sampling strategy. The result is presented in Figure 3.3.



It is possible to see that the stratified sampling with the selected parameters performs as well as a uniform random sampling at 70% and a relevance based sampling of 70% in terms of IRSs ranking correlation. The two 70% sampling approaches make each run contribute 35 documents to the pool while the stratified approach, by going down to rank 100 in the runs, make each run contribute 45 documents. Given that we used the 2007 collection for the experimental comparison we must notice that relevance assessments have been performed up to rank 50. Therefore, several documents ranked from 51 to 100 may not have been assessed, so they are considered not relevant in the experiments even if they could be. If we want to fairly compare the judgement effort of the three sampling approaches we have to count the number of documents the stratified sampling approach make the runs contribute up to rank 50, that is 30. In this way we can see that the stratified approach enable a lower judging effort than the uniform random sampling and the relevance based sampling.

4 Results

No results are available yet.

References

1. 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.

L3S at INEX 2008: Entity Ranking using Structured Information

Nick Craswell[1], Gianluca Demartini[2], Julien Gaugaz[2], and Tereza Iofciu[2]

[1] Microsoft Research Cambridge
7 JJ Thomson Ave
Cambridge, UK
nickcr@microsoft.com
[2] L3S Research Center
Leibniz Universität Hannover
Appelstrasse 9a D-30167 Hannover, Germany
{demartini,gaugaz,iofciu}@L3S.de

Abstract. Entity Ranking is a recently emerging search task in Information Retrieval, where the goal is not finding documents matching the query words, but instead finding entities which match those described in the query.
In this paper we focus on the Wikipedia corpus interpreting it as a set of entities and propose algorithms for finding entities based on structured representation of entities for three different search tasks: entity ranking, list completion, and entity relation search. The main contribution is a methodology for indexing entities using a structured representation. Our approach focuses on creating an index of facts about entities for the different search tasks. More, we use the link structure information for improving the effectiveness of search.

1 Introduction

Entity ranking (ER) is an important step over the classical document search as it has been done so far. The goal is to find entities relevant to a query rather than just finding documents (or passages from documents) which contain relevant information. Ranking entities according to their relevance with respect to a given query is important in scenarios where the information load is bigger than what the user can handle. That is, with a correct ranking scheme the system can present the user with only entities of interest, and avoid the user having to analyze the entire set of retrieved entities.

As a step in this direction, we present, in this paper, our approach to ranking entities in Wikipedia which is based on the usage of a structured representation of entities in order to enable effective search. Conceptually, we represent an entity as a set of attribute name / value pairs. For example an entity representing Albert Einstein can be represented as follows:

first name Albert

last name Einstein
born 1879-03-14
died 1955-04-18
fields physics

More, we evaluate it on the Wikipedia XML corpus provided within the INEX 2008 initiative. The main contribution of this paper is a set of methodologies for representing entities in a structured fashion for enabling entity ranking.This is done using different representations for the different ER search tasks, namely entity ranking (XER), list completion (LC), and entity relation search (ERS).

The rest of the paper is organized as follows. In section 2 we describe the general architecture of the developed retrieval systems. In section 3 we describe the algorithms developed for the three different search tasks.

2 Architecture Overview

In this section we describe the architecture of the Entity Ranking System we used for creating the runs submitted to INEX 2008. The architecture design for the system used for XER runs is based on the model presented in [1]. The first step is the creation of the inverted index out of the XML Wikipedia document collection. Starting from the raw structured XML documents, we create a Lucene[3] index[4] with one Lucene document (i.e., a vector in the Vector Space) for each Wikipedia document. We first parse the document collection using standard Java libraries[5]. After this, we create an index with different fields (acting as separate inverted indexes, which can be combined for retrieval) for the title, text, and category of Wikipedia entities. After the creation of the inverted index, the system is able to process the XER topics.

After the generation of the query, the fields of the index can be queried and a ranked list of entities is retrieved merging the ranked lists coming from the different fields. The ranking of the retrieved entities is done according to cosine similarity with the query using the standard TFxIDF scoring function. The methods used for retrieving results for the XER task are described in Section 3.1

3 Algorithms

3.1 Entity Ranking Task

Entity Ranking using NLP techniques on Topic Title and Description. The first XER run that we submitted to INEX 2008 is a title-only run (i.e., it uses only the title part of the topic). It uses a combination of NLP and IE techniques

[3] `http://lucene.apache.org/`
[4] The IR model used by Lucene is the Vector Space Model with standard cosine similarity.
[5] We used the Java 6 `javax.xml.stream.*` classes.

on the user query in order to improve and adapt it for the ER search task. The system extends the original keyword query adding also related words and synonyms, and it adds more weight on named entities and key-concepts (e.g., the type of result which is needed by the user) which are present in the original query by duplicating the terms in the query. Additionally, the key-concepts which are present in the original query are used to search in the anchor-text of the outlinks of the pages. Finally, the match of the category information in the query and the category information of the Wikipedia pages also influences the ranking of the results. A detailed description of the algorithm is presented in [1]

The second ER runs that we submitted to INEX-XER is also using the description part of the topic. In this case, the system also uses the description information in order to provide better results to the user. The entire title and description are used as a long query. A TF.IDF vector is built out of it and Wikipedia articles are ranked according to their cosine similarity to the query. Additionally, the title and the category information in the topic is used for searching the category information in the Wikipedia articles. This helps in ranking first entities of the desired type. Finally, also the outgoing links of the Wikipedia documents are used for finding relevant entities. The topic title and the named entities present in the description are used together for searching in the outlinks of a page. The combinations of these three searches produce the final rank of entities.

Unsupervised Template-Less Information Extraction. We describe hereafter a first approach to generate structured queries from an unstructured document which has not been used for the officially submitted runs. By structured query we understand a set of attribute name / value pairs. It comprises two steps: *entity reference resolution* first identify the entity or entities referenced in the document. Those reference are then used for *extracting attribute* name and values related to them from the text. Those two steps are described in the following.

Entity Reference Resolution For building the structured query, we first have to find references of the considered entity occurring in the Wikipedia page at hand. It is presently done in two different fashions depending on whether the considered page is the page representing the entity, or a page linking to the page representing the entity.

- **On a page representing the considered entity**, occurrences are all set of terms similar to the page title. The title might have two parts. The first part is called the *main title*, and is present in all titles. The second part of a title is called *sub-title* and is the terms in the title occurring after a separating character like a '—' or in parentheses '()', '{}' or '[]'. For example, in the title "Napoleon (1995 film)", the main title is "Napoleon" and the sub-title is 1995". In this case we consider a set of token on the page representing the considered entity similar to the title if they consist of either only the main title or the whole title (main plus sub).

– **On a page linking to the considered entity page**, occurrences are simply anchored text of the link.

Attribute Extraction With help of a grammatical structure parser and manually created rules on the grammatical structure, we extract attribute names and values representing the entity represented by a Wikipedia page. Those rules include:

– The entity reference is a subject, the attribute name is an active verb plus prepositions, and the attribute value is the object of the verb.
– The entity reference is an object, the attribute name is a passive verb plus prepositions, and the attribute value is the subject of the passive verb.

Other ad-hoc rules were also created based on a training set of Wikipedia pages, and the entity references in them. Those extracted attributes have the following particularities compared to attributes usually found in knowledge bases:

– Attribute names are less expressive and consists mostly of verbs plus modifiers or prepositions.
– Attribute values are mostly longer, and dirtier in the sense that not all the terms of the attribute value are relevant. This is because the value consists mostly of a whole verbal phrase.

The set of extracted information creates a repository of entity descriptions. In the future we will investigate how to leverage this repository to improve entity search. One possible approach would be to apply similar technique to the INEX topic to obtain a set of attribute representing the entities sought. We then retrieve relevant entities by performing a keyword (full-text) query on the structured repository consisting of the attribute values of the entity thus obtained. Note that we can also leverage the structure in the query, by taking into account that two keywords belonging to a same attribute in the query should also appear in the same attribute in the structured repository.

3.2 List Completion Task

In this task the information need is specified by the topic title and the type of the desired entities is given by example entities. From the INEX topics we can not use the category information anymore, this has to be learned from the examples entities with the help of the Wikipedia structure.

Instinctively, one would say that if two entities are related by satisfying a query need, they should have at least one common category. But, as Wikipedia is a collaborative effort, entities usually belong to more than one category, depending on what the authors considered appropriate. The category information for entities is not always complete and sometimes is not entirely correct also. Thus, from the topic example entities (usually two or three) we need to discover which are the categories that will best satisfy the query need.

In our approach, for the example entities we extract two sets of categories from Wikipedia. The first set, which we call *main categories*, consists of the direct

categories the example entities belong to. From these categories we filter out the ones that are too general, i.e. have a high indegree (i.e. the number of entities that belong to that category) and we keep the ones with indegrees smaller than 1000. Each category from this set has a score computed based on the number of entities that have the respective category. The score for a category is calculated as ten to the power of the number of occurring entities. Furthermore, the score is then divided by the category's popularity, based of the indegree of the category. The assumption is that the higher the indegree of a category is, the less specific that category is and the entities that belong to that category do not have a strong relation.

The second set of categories, which we called *soft categories*, consists of the top 30 categories assigned to the entities that are linked to by the example entities from the topics. The score of these categories is given by the count of their occurrences in the linked entities. These categories are also filtered based on the size of the indegree. These scores are then divided by the categories' popularity score.

Finally, for our first LC run, we extract from Wikipedia the entities that belong to the categories from these two sets, we order the entities based on their popularity and on the score of the categories they belong to.

For our second LC run, we also used the topic title. In order to use together title and example entities, we search the collection (indexed using the vector space model with tf.idf weights and cosine similarity) using the title and we re-rank the results based on the category sets found from the example entities.

3.3 Entity Relation Search Task

For the Entity Relation Search Task we have indexed the relation between entities from Wikipedia in a structured fashion. We consider two entities to be related if they co-occur in a sentence sized window. For each two co-occurring entities we have indexed the sentences in which they appear together to define the relation between the entities. In the 1 table we show a snippet from entity relation index, with a few examples for the entities *Flint River* and *Albert Einstein*.

The Entity Relation Search that we submitted to INEX 2008 are based on our two XER runs. The first run is based on the title-only Entity Ranking Run. It is using the entities retrieved for the ER run as left entities LE. Then, for each left entity $l_i \in LE$ a search for relevant right entities is performed. Possible candidates are retrieved from the index: all triples containing as subject the left entities are considered. All the right entities candidates are ranked according to a score which is computed as the overlap between the predicate in the index and the Entity Relation Search Title in the topic.

The second ERS run is based based on our second ER run and it is computed as the first one.

Entity	Predicate	Entity
flint river	the flint river with an area of 568 square miles is a tributary to the	tennessee river
flint river	much of the 342 sq	watershed
albert einstein	he proposed the and also made major contributions to the development of and the theory provided the foundation for the study of and gave scientists the tools for understanding many features of the universe that were discovered well after einstein death	cosmological models
albert einstein	he proposed the and also made major contributions to the development of and	theory of relativity

Table 1. Entity relation index

Acknowledgments. This work was partially supported by the Okkam project funded by the European Commission under the 7th Framework Programme (IST Contract No. 215032).

References

1. Gianluca Demartini, Claudiu S. Firan, Tereza Iofciu, Ralf Krestel, and Wolfgang Nejdl. A model for ranking entities and its application to wikipedia. In *LA-WEB*, 2008.

CSIR at INEX 2008 Entity-Ranking Task: Entity Retrieval and Entity Relation Search in Wikipedia

Jiepu Jiang[1], Wei Lu[1], Xianqian Rong[1], Yangyan Gao[1]

[1] Center for Studies of Information Resources (CSIR),
School of Information Management, Wuhan University, P. R. China
{jiepu.jiang, reedwhu, rongxianqian, gaoyangyan2008}@gmail.com

Abstract. In this paper, we mainly describe our methods taken in INEX 2008 entity-ranking task. Firstly, we discuss how to extend current expert retrieval models to deal with the category field in INEX entity ranking queries. Then, on such basis, we propose similar solutions to the former list completion task and the entity relation search task.

Keywords: Entity retrieval, entity ranking, language model.

1 Introduction

In recent years, expert retrieval has been focused and widely discussed in various aspects. A few ranking models are created for this issue and have been proved to be effective. In contrast, the issue of entity retrieval as a general case is less discovered until the INEX entity-ranking task in 2007. This task has introduced a perspective of entity retrieval beyond expert retrieval, which involves not only more types of entities in practice, but also various tasks.

One of the main foci in INEX entity-ranking task is to return entities that satisfy a topic with multiple fields (the INEX XER task). Typical topic fields consist of a text query field, which describes the entity search query using natural language, and also a category field specifying possible categories of relevant entities in Wikipedia and an example field which contains a small and incomplete list of relevant entities. Two mandatory runs are required for the INEX XER task: one should use the text field and the category field, while the other should make use of the text field and the example entity list (the former list completion task).

The main difference between the INEX XER task and an expert retrieval task (such as the TREC expert search task) resides in the extra demands for match in entity categories for the former. In the expert search task, we can focus in topical demands of queries since the entity type is pre-defined. As a result, expert retrieval models are aiming at finding topical relevant entities. However, in the INEX XER task, we are facing candidate entities of various categories and the users' demands for category are adhoc instead of pre-defined. Generally, current expert retrieval models are incapable of dealing with such extra demands for category. As a result, we extend current expert search models to incorporate such extra demands for category.

On such basis, the entity relation search task (ERS) task is discussed. The ERS task is a new task in INEX. Different from the XER task, the ERS task aims at finding right entity pairs with some specified relationship. We adopt a 3-step method in this task: firstly, topical relevant entities are retrieved using our proposed extensive model; then, we find for each relevant entity a list of entities have the specified relationship; in the end, those found entity pairs are re-ranked all together.

The remainder of this paper is organized as follows: section 2 reviews models on expert search and INEX entity ranking; in section 3, we describe models of the XER task and the ERS task; section 4 draws a conclusion. In the pre-proceeding session of INEX, evaluation of models is not available and thus is not included in this paper.

2 Related Works

The problem of expert retrieval (finding relevant experts of a specific topic) can be considered as the issue of retrieving entities of some pre-defined category, i.e. people. Early approaches of expert retrieval usually involve empirical methods and structured or semi-structured resources. Since the TREC 2005 expert search task, expert retrieval is focused by IR researchers and a lot of models are created. Specifically, language modeling methods for expert search are proved to be highly effective and thus are widely adopted.

Generally, language models for expert retrieval rank experts by the probability that the query is generated by the experts. In TREC 2005, Cao et al. [1] and Azzopardi et al. [2] introduce two kinds of language models for expert retrieval. These methods are later explained by Balog et al. [3] as candidate model (model 1) and document model (model 2). Fang et al. [4] also described a similar framework, but they had explicitly modeled on relevance and used the probability ranking principle to rank.

Further, some detailed problems are studied under the framework. Serdyukov et al. [5] introduced a method to enhance performance by query modeling. Balog et al. [6] elaborated the estimation method of candidate-document association. Serdyukov et al. [7] explored the relevance propagation in expert search. Petkova et al. [8] explored the dependence between candidate and terms using proximity-based methods. Most recently, Balog et al. [9] fully considered non-local information available in the collection and significantly improved the performance.

In contrast, only limited researchers studied the entity retrieval as a general case. Most of them noticed the problem of categories and proposed feasible solutions. For example, Vercoustre et al. [10] used a Jaccard coefficient between query category set and entity category set for ranking. Demartini et al. [11] expanded the category query field using YAGO, while Jamsen et al. [12] achieved category expansion using Wikipedia hierarchies. Zhu et al. [13] treats entities' categories as a metadata field and searches entities between multi-fields.

In this paper, we will extend current expert retrieval models to deal with the entity retrieval problem under the environment of the INEX entity-ranking task.

3 Models

In this section, we will describe our language modeling process for entity retrieval. A lot of language models for expert retrieval have already been explored and testified to be effective, which constitute an important part of our models. For the XER task, we mainly discuss the problem under the settings of two mandatory runs. For the ERS task, we propose models using both the mandatory fields and some extra features.

3.1 Entity Ranking Task

The query of entity ranking can be multi-fields. In this paper, we mainly discuss the queries of two mandatory runs mentioned in section 1. For the first run, only the text query and the category field can be used. So, we can represent the query as $q(q_{text}, q_{cat})$, where q_{text} refers to the text query, and q_{cat} represent the category query. The second run is studied in the following subsection as the former list completion task.

As a result, from a language model perspective, the problem of entity retrieval can be translated as: given a query q, to find the possible relevant entity e, i.e. to estimate $p(e|q)$ for each entity and accordingly rank entities. Then, $q(e|q)$ can be transformed to (1). Assuming each entity shares the equal probability, $q(e|q)$ is proportional to $p(q|e)$. Thus, we can use $p(q|e)$ to rank entities.

$$p(e \mid q) = \frac{p(q \mid e) * p(e)}{p(q)} \tag{1}$$

Considering q consists of two parts (q_{text} and q_{cat}), we can simply assume them to be independent and transform $p(q|e)$ as (2). Then, we can treat $p(q_{text}|e)$ and $p(q_{cat}|e)$ separately.

$$p(q \mid e) = p(q_{text}, q_{cat} \mid e) = p(q_{text} \mid e) * p(q_{cat} \mid e) \tag{2}$$

For the $p(q_{text}|e)$, we adopt the expert search model 2 [3] to estimate it. Assuming e is independent for the generation of q_{text} in d, we can also use (4) as a simplification. Then, we estimate $p(q_{text}|d)$ using text retrieval models as in (5). In (5), each query terms in q_{text} is assumed to be independent, $p_{ml}(t|d)$ is the max likelihood estimation of t in d, $p_c(t)$ is the probability of t in the whole corpus, λ is a parameter which is set to 0.5 in our runs.

$$p(q_{text} \mid e) = \sum_{d \in D} p(q_{text} \mid d, e) * p(d \mid e) \tag{3}$$

$$p(q_{text} \mid e) = \sum_{d \in D} p(q_{text} \mid d) * p(d \mid e) \tag{4}$$

$$p(q_{text} \mid d) = \prod_{t \in q_{text}} p(t \mid d) = \prod_{t \in q_{text}} \{(1-\lambda) * p_{ml}(t \mid d) + \lambda * p_c(t)\} \tag{5}$$

For the $p(d|e)$, we also adopt a general method, i.e. (6). In (6), *D'* refers to a subset of documents that are associated with e, and $a(d_i,e)$ is the association between d_i and e. In our model, $a(d_i,e)$ is simply measured as the frequency of e in d_i. For elaborated methods, Balog et al. [6] had a thorough exploration.

$$p(d \mid e) = \frac{a(d,e)}{\sum_{d_i \in D'} a(d_i,e)} \tag{6}$$

As for $p(q_{cat}|e)$, q_{cat} can be treated as a category set, where every single element category is generated independently from the entity's labeled category set CAT_e. Then, we can estimate $p(q_{cat}|e)$ as (7):

$$p(q_{cat} \mid e) = \prod_{cat_i \in q_{cat}} p(cat_i \mid CAT_e) \tag{7}$$

$$p(q_{cat} \mid e) = \left(\prod_{cat_i \in q_{cat}} p(cat_i \mid CAT_e) \right) * \left(\prod_{cat_j \notin q_{cat}} \{1 - p(cat_j \mid CAT_e)\} \right) \tag{8}$$

It should be noted that in (7) we adopt the q_{cat} as a sequence of categories, though it seems more reasonable to treat it as a set and estimate it in (8). But we choose (7) here for the following reasons: on the one hand, the category queries are not ensured to be accurate, thus it is also arguable to model other categories as not generated by CAT_e; on the other hand, a thorough estimation of a large amount of unseen categories in (8) consumes much computational resources, which is not efficient.

For (7), we estimate each $p(cat_i|CAT_e)$ using a maximum likelihood estimation with a smooth. Then $p(cat_i|CAT_e)$ can be further divided into each categories in CAT_e, which is presented in (9).

$$p(cat_i \mid CAT_e) = (1-\lambda) * \sum_{cat_j \in CAT_e} \frac{p(cat_i \mid cat_j)}{|CAT_e|} + \lambda * p(cat_i) \tag{9}$$

In (9), $p(cat_i)$ is the corpus probability of cat_i, which is used for smoothing. Then we discuss several possible circumstances for $p(cat_i|cat_j)$ in a rule-based case: when cat_i is exactly cat_j, or cat_i is cat_j's parent category, we set $p(cat_i|cat_j)$ to 1; when cat_i is cat_j's child category, we set $p(cat_i|cat_j)$ to $1/|cat_j|$, where $|cat_j|$ is the size of the cat_j's child category set; for other circumstances, $p(cat_i|cat_j)$ is set to 0.

3.2 List Completion Task

Another mandatory run of the XER task is formerly known as the list completion task, which provides a list of example entities instead of the category field in retrieval. So, we can represent the query as $q(q_{text}, q_{lc})$, in which q_{text} refers to the text query and q_{lc} represents the list of example entities. We can rank each entity e according to $p(q|e)$, which can also be transformed to (10) if we assume the independence between q_{text} and q_{lc}.

$$p(q \mid e) = p(q_{text}, q_{lc} \mid e) = p(q_{text} \mid e) * p(q_{lc} \mid e) \quad (10)$$

In (10), we can also estimate $p(q_{text}|e)$ in (3). Then, the problem resides in $p(q_{lc}|e)$, which can be transformed to (11). In (11), cat_i refers to any category labeled with entities in the list *lc*. Then, $p(cat_i|CAT_e)$ can be estimated in (9).

$$p(q_{lc} \mid e) = \prod_{cat_i \in lc} p(cat_i \mid CAT_e) \quad (11)$$

In INEX 2008, we submit two mandatory runs for the XER task:

(1) Run *1_CSIR_ER_TC_mandatoryRun*. This run uses methods described in 3.1, which makes use of the title field and the category field.

(2) Run *1_CSIR_LC_TE_mandatoryRun*. This run uses methods described in 3.2, which makes uses of the title field and the example list.

3.3 Entity Relation Search

The entity relation search (ERS) task is a new task for entity ranking, which aims at finding entity pairs with appropriate relation. For this task, the problem can be solved in three steps: firstly, searching for a list of entities relevant to the topic, as stated in 3.1 and 3.2; then, for each relevant entity *e* retrieved, finding a group of target entities that have the specified relation with *e*; in the end, re-rank entity pairs all together.

As a result, the main problem is: given entity *e*, to find a list of target entities that have relation q_r with e_0 and match category query q_{cat}. So, queries can be represented as a multi-field query $q(e_0, q_r, q_{cat})$, in which e_0 is the entity that retrieved entities are associated with, q_r represents the relation text query, q_{cat} represents the category of the target entities. Then, we can rank each entity according to $p(q|e)$, which can also be transformed to (12) if we assume independence between any two parts of q.

$$p(q \mid e) = p(e_0, q_r, q_{cat} \mid e) = p(e_0 \mid e) * p(q_r \mid e) * p(q_{cat} \mid e) \quad (12)$$

Then, we can estimate $p(q_r|e)$ as (3) and $p(q_{cat}|e)$ as (7). As for the $p(e_0|e)$, we can also adopt models similar to expert search models in estimation. By treating e_0 as a term, we can also use (3) to estimate it.

In INEX 2008, we submit two runs for the ERS task:

(1) Run *1_CSIR_ERS_TC_R_TC_ERSWITHCATE*. This run uses methods in (12), which makes use of the title field and the category field to find each e_0, and use relation title and target entity categories to find each target entities.

(2) Run *1_CSIR_ERS_TC_R_T_mandatoryRun*. Compared with (12), this run only makes use of the relation title when finding target entities for each e_0.

4 Conclusion

In this paper, we describe our methods of entity ranking and entity relation search in the INEX 2008 entity-ranking task. Some simple methods are discussed to understand the difference between expert search and entity retrieval, i.e. to deal with the demand

for category in entity retrieval task. Refinements of models are left as future works, which includes the independence assumptions and estimation methods. Besides, we plan to future discover methods of dealing with categorical demands under some more relaxed settings, for example, to use only the text query field. The estimation of models takes advantages of Wikipedia features, which is also one of the limitations and needs to be relaxed.

References

1. Y. Cao, J. Liu, S. Bao, H. Li. Research on Expert Search at Enterprise Track of TREC 2005. In Proceedings of the 14th Text REtrieval Conference (TREC 2005), 2005.
2. L. Azzopardi, K. Balog, M. de Rijke. Language Modeling Approaches for Enterprise Tasks. In Proceedings of the 14th Text REtrieval Conference (TREC 2005), 2005.
3. K. Balog, L. Azzopardi, M. de Rijke. Formal Models for Expert Finding in Enterprise Corpora. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, Seattle, Washington, USA, 2006: 43-50.
4. H. Fang, C. Zhai. Probabilistic Models for Expert Finding. In Proceedings of the 29th annual European Conference on Information Retrieval Research, Rome, Italy, 2007: 418-430.
5. P. Serdyukov, Sergey Chernov, Wolfgang Nejdl. Enhancing Expert Search through Query Modeling. In Proceedings of the 29th annual European Conference on Information Retrieval Research, Rome, Italy, 2007: 737-740.
6. K. Balog, M. de Rijke. Associating People and Documents. In Proceedings of the 30th annual European Conference on Information Retrieval Research, Glasgow, Scotland, 2008: 296-308.
7. P. Serdyukov, Henning Rode, Djoerd Hiemstra. University of Twente at the TREC 2007 Enterprise Track: Modeling relevance propagation for the expert search task. In Proceedings of the 16th Text REtrieval Conference (TREC 2007), 2007.
8. D. Petkova, W. B. Croft. Proximity-Based Document Representation for Named Entity Retrieval. In Proceedings of the 16th ACM conference on Conference on information and knowledge management, Lisbon, Portugal, 2007: 731-740.
9. K. Balog, M. de Rijke. Non-Local Evidence for Expert Finding. In Proceedings of the 17th ACM conference on Conference on information and knowledge management, Napa Valley, California, USA, 2008: 731-740.
10. A. Vercoustre, J. A. Thom, J. Pehcevski. Entity Ranking in Wikipedia. In Proceedings of the 2008 ACM symposium on Applied computing, Fortaleza, Ceara, Brazil, 2008.
11. G. Demartini, C. Firan, T. Iofciu. L3S Research Center at the INEX Entity Ranking Track. In Proceedings of the INEX 2007, 2007.
12. J. Jamsen, T. Nappila, P. Arvola. Experiments on Category Expansion at INEX 2007. In Proceedings of the INEX 2007, 2007.
13. J. Zhu, D. Song, S. Ruger. Integrating Document Features for Entity Ranking. In Proceedings of the INEX 2007, 2007.

The Impact of Topic Difficulty Prediction on Entity Ranking

Jovan Pehcevski[1], Vladimir Naumovski[1], and Anne-Marie Vercoustre[2]

[1] Faculty of Management and Information Technologies, Skopje, Macedonia
{jovan.pehcevski,vladimir.naumovski}@mit.edu.mk
[2] INRIA, Rocquencourt, France
anne-marie.vercoustre@inria.fr

Extended Abstract

This extended abstract describes the joint participation of the Faculty of Management and Information Technologies and INRIA groups in the Initiative for the Evaluation of XML retrieval (INEX) XML Entity Ranking (XER) track in 2008. Of the two tasks in the INEX 2008 XER track, we submitted one run for the entity ranking sub-task and four runs for the entity list completion sub-task. We did not submit any runs for the entity relation search task.

Introduction

Entity ranking has recently emerged as a research field that aims at retrieving entities as answers to a query. Many approaches to entity ranking have been proposed, and the performances of most of them were evaluated by the INEX 2007 XER track [2] on the INEX Wikipedia XML test collection [3]. This resulted in many advances to this research field; however, little attention has been put on the impact of the different types (or classes) of topics on the entity ranking performance.

For INEX 2008, our aim was to develop a method for topic difficulty prediction in the research field of entity ranking. Our approach is based on the generation of a topic classifier that can classify the INEX XER topics from a number of features extracted from the topics themselves (also called a-priori features) and possibly from a number of other features calculated at run time (also called a-posteriori features). The main goal is to apply the topic difficulty prediction to improve the effectiveness of our entity ranking system that was evaluated as one of the best performing XER systems at INEX 2007 [2, 9].

Topic Difficulty Classification

The classification of topics into groups is based on how well the participant systems in the INEX 2007 XER track answered to the topics. For each topic, we calculate the topic difficulty using the Average Average Precision (AAP) measure [4, 5]. AAP is the average of the average precisions of all the systems reported for a given topic: the higher the AAP, the easier the topic.

We define two methods for grouping topics into classes depending on the number of groups we want to build, either two or four classes to experiment with two different types of classes. For grouping the topics into two classes (Easy and Difficult), we use the mean AAP metric as a splitting condition: if AAP for a given topic is superior to the mean AAP (calculated across all topics) then the topic is classified as Easy otherwise it is classified as Difficult. For grouping the topics into four classes (Easy, Moderately_Easy, Moderately_Difficult, and Difficult), we use the mean AAP and the standard deviation around the mean as a splitting condition:

if AAP >= (mean AAP + stDev) **then** Easy topic
if AAP >= (mean AAP) **and** AAP < (mean AAP + stDev) **then** Moderately_Easy topic
if AAP >= (mean AAP - stDev) **and** AAP < (mean AAP) **then** Moderately_Difficult topic
if AAP < (mean AAP - stDev) **then** Difficult topic

The above two or four classes of INEX 2007 XER topics are then used as a basis for evaluating our automatic feature-based topic classification algorithm.

Our XER Approach

Our approach to identifying and ranking entities combines: (1) the full-text similarity of the entity page with the query; (2) the similarity of the page's categories with the categories of the entity examples; and (3) the link contexts found in the top ranked pages returned by a search engine for the query. We focus on the entity list completion sub-task, where a few examples of relevant entities are provided in the topic description as a form of relevance feedback information.

We developed an XER system that was used and evaluated as one of the best performing systems on the INEX 2007 XER test collection [2, 9]. Our XER system involves the following modules:

1. The *topic module* takes an INEX topic as input and generates the corresponding full-text query and the list of entity examples.
2. The *search module* sends the query to a search engine[3] and returns a list of scored Wikipedia pages. The assumption is that a good entity page is a page that answers the query.
3. The *link extraction module* extracts the links from a selected number of highly ranked pages, together with the information about the paths of the links (XML paths). The assumption is that a good entity page is a page that is referred to by a page answering the query.
4. The *linkrank module* calculates a weight for a page based (among other things) on the number of links to this page. The assumption is that a good entity page is a page that is referred to from contexts with many occurrences

[3] We used Zettair, an open source search engine: http://www.seg.rmit.edu.au/zettair/

of the entity examples. A coarse context would be the full page that contains the entity examples. Smaller and better contexts may be elements such as paragraphs, lists, or tables, which may be determined either statically or dynamically [6]. The main drawback of the static approach is that it requires a pre-defined list of element contexts which is totally dependent on the document collection. The advantage is that, once defined, the list-like contexts are easy to identify. On the other hand, the main advantage of the dynamic approach is that it is independent of the document collection, however the possible drawback is that narrow contexts containing only one entity example are never identified. In this work we plan to investigate a novel dynamic approach for determining the contexts that addresses the above issue.

5. The *category similarity module* calculates a weight for a page based on the similarity of the page categories with the categories attached to the entity examples. The assumption is that a good entity page is a page associated with a category close to the categories of the entity examples. The category similarity measures may utilise lexical similarity between category names [7], or, as we plan to investigate in this work, different category weighting rules that could be applied on the sets of categories directly attached to both the example and the answer entities.
6. The *full-text module* calculates a weight for a page based on its initial search engine score.

The global score $S(t)$ for an answer entity page is calculated as a linear combination of three scores, the linkrank score $S_L(t)$, the category similarity score $S_C(t)$, and the full-text score $S_Z(t)$:

$$S(t) = \alpha S_L(t) + \beta S_C(t) + (1 - \alpha - \beta) S_Z(t) \quad (1)$$

where α and β are two parameters whose values can be tuned differently depending on the entity retrieval task.

Details of the global score and the three separate scores can be found in previous publications [6, 7]. In this work we are interested in automatically adapting the values of α and β parameters to the topic, depending on the topic class. By predicting the optimal values for α and β parameters that correspond to each class of topic difficulty, we aim at improving the performance score of our system over the current best performance score that uses pre-defined static values for the α and β parameters.

Topic Difficulty Prediction

Our methodology for predicting topic difficulty is based on generating a classifier to classify topics in two or four classes (as described above). The classifier is built using features extracted from the INEX XER topic definition. We use the open source data mining software Weka [10] developed by the University of Waikato. Weka is a collection of machine learning algorithms for data mining tasks that, given a training set of topics, can generate a classifier from the topics and their associated features.

From the specific structure of the INEX XER topics we developed several a-priori and a-posteriori features, of which we needed to choose those features that best correlated with the topic difficulty. We generated many classifiers, each one associated with a random subset of the 46 INEX 2007 XER topics. We then manually analysed all the decision trees generated by Weka to identify the features that were actually used by the generated classifiers. As a result, we could extract nine features that correlated well with the topic difficulty prediction [8].

We tried many different mostly random combinations of training and testing topic subsets, but we found that, because of the relatively small sizes of the training subsets available for the INEX 2007 XER track, the precision of the correctly classified instances was between 65% and 82%.

To improve the precision, we used a well known approach that combines five decision trees, each generated from slightly different topic subsets. This is known as Random Forests [1] and was used in query prediction by Yom-Tov et al. [11]. Before implementing the combined classifier, we carefully built the training topic subset for each individual predictor so that the included topics were representative of different features. For the final combined classifier we also had to build a testing topic subset that did not include any of the training topics.

The final topic difficulty prediction algorithm was built using a simple voting system (which is the reason why we needed an odd number of classifiers). For building a two-class classifier the voting algorithm is trivial: for a topic we get a prediction from the five classifiers and count the number of predictions as Easy and the number of predictions as Difficult; the majority gives the prediction for the final classifier. For example, if the predictions from the five classifiers are [diff, easy, easy, diff, diff], the combined prediction is Difficult. The combined classifier resulted in a precision of 94% on our testing topic subset which was much better than what we could achieve with a single classifier [8].

Discussion

Our objective with topic difficulty prediction was to improve the performance of our XER system by dynamically tuning the values for system parameters according to the predicted topic class. Specifically, for each of the INEX 2008 topics, we aim at adapting the values for the α and β system parameters in order to improve the best overall performance score without using prediction. Of the four runs we submitted in the entity list completion sub-task of the INEX 2008 XER track, two runs do not use prediction, while the other two use two-class and four-class prediction.

On the INEX 2007 XER test collection, we found that two-class prediction of topic difficulty was performing better than the baseline (our last year best run), although the difference in performance was not statistically significant [8]. On the other hand, the four-class prediction of topic difficulty resulted in decreased performance compared to the baseline, which was mainly due to the fact that the topic prediction algorithm was specifically designed for two-class rather than for four-class prediction. These results were promising, however the small size of the training and testing topic subsets we used in INEX 2007 did not allow for

very conclusive evaluation. We therefore intend to use topic difficulty prediction in our system and apply it on the INEX 2008 XER test collection. We believe that the larger number of topics used in the INEX 2008 XER test collection will confirm the significance of our findings and highlight the positive impact of topic difficulty prediction on entity ranking.

Acknowledgements

Most of this work was completed while Vladimir Naumovski was doing his internship at INRIA in 2008.

References

1. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
2. A. P. de Vries, A.-M. Vercoustre, J. A. Thom, N. Craswel, and M. Lalmas. Overview of the INEX 2007 entity ranking track. In *Proceedings of the Sixth INEX workshop, Schloss Dagstuhl, Germany, 2007*, volume 4862 of *Lecture Notes in Computer Science*, pages 1–23, 2008.
3. L. Denoyer and P. Gallinari. The Wikipedia XML corpus. *SIGIR Forum*, 40(1):64–69, 2006.
4. S. Mizzaro. The good, the bad, the difficult, and the easy: Something wrong with information retrieval evaluation? In *Proceedings of the 30th European Conference on Information Retrieval (ECIR'08)*, volume 4956 of *Lecture Notes in Computer Science*, pages 642–646, 2008.
5. S. Mizzaro and S. Robertson. HITS hits TREC: Exploring IR evaluation results with network analysis. In *Proceedings of the 30th ACM SIGIR conference on Research and development in information retrieval (SIGIR'07)*, pages 479–486, Amsterdam, The Netherlands, 2007.
6. J. Pehcevski, A.-M. Vercoustre, and J. A. Thom. Exploiting locality of Wikipedia links in entity ranking. In *Proceedings of the 30th European Conference on Information Retrieval (ECIR'08)*, volume 4956 of *Lecture Notes in Computer Science*, pages 258–269, 2008.
7. J. A. Thom, J. Pehcevski, and A.-M. Vercoustre. Use of Wikipedia categories in entity ranking. In *Proceedings of 12th Australasian Document Computing Symposium (ADCS'07)*, pages 56–63, Melbourne, Australia, 2007.
8. A.-M. Vercoustre, J. Pehcevski, and V. Naumovski. Topic difficulty prediction in entity ranking. Submitted for publication, 2008.
9. A.-M. Vercoustre, J. Pehcevski, and J. A. Thom. Using Wikipedia categories and links in entity ranking. In *Proceedings of the Sixth INEX workshop, Schloss Dagstuhl, Germany, 2007*, volume 4862 of *Lecture Notes in Computer Science*, pages 321–335, 2008.
10. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.
11. E. Yom-Tov, S. Fine, D. Carmel, A. Darlow, and E. Amitay. Juru at TREC 2004: Experiments with prediction of query difficulty. In *Proceedings of the Thirteenth Text Retrieval Conference (TREC 2004)*, 2004.

The University of Amsterdam (ILPS) at INEX 2008

Wouter Weerkamp, Jiyin He, Krisztian Balog, and Edgar Meij

University of Amsterdam, ISLA, Kruislaan 403, 1098SJ Amsterdam, The Netherlands,
{w.weerkamp,j.he,k.balog,e.j.meij}@uva.nl

Abstract. We describe our participation in the INEX 2008 Entity Ranking and Link-the-Wiki tracks. We provide a detailed account of the ideas underlying our approaches to these tasks. For the Link-the-Wiki track, we also report on the results and findings so far.

1 Introduction

This year the Information and Language Processing Systems (ILPS) group of the University of Amsterdam participated in two INEX tracks: Entity Ranking and Link-the-Wiki. For the Entity Ranking track our main emphasis was on developing a general language modeling framework to model the tasks at hand. The models that are developed for both task offer us plenty of opportunity to experiment with various ways of estimating components of the models. We describe the components of the models and the way we estimated these. Of main interest to us was the use of category information in the various components (e.g., by estimating entity similarity in list completion), combining different entity representations, and query modeling. In our participation in the Link-the-Wiki track our main aim was to explore different features that indicate links between Wikipedia pages, as well as to develop a generative approach to automatic link generation. We submitted runs designed to compare the influence of different features on link generation.

In this paper, we describe our participation for the tracks mentioned above, in two largely independent sections: Section 2 on our entity ranking track participation and Section 3 on our work in the link-the-wiki track. Finally, we conclude in Section 4.

2 Entity Ranking

The Enitity Ranking track of this year's INEX features three tasks: *entity ranking*, *list completion*, and *entity relation search*. In our participation, we focus on the first and second tasks, leaving entity relation search for coming years. Both tasks (entity ranking and list completion) are aimed at retrieving entities from a semi-structured document collection. The document collection at hand is Wikipedia, and an entity by definition is a Wikipedia article.

The entity ranking task aims at retrieving entities (Wikipedia pages) given a certain topic and Wikipedia category: the goal is to identify the Wikipedia pages that are relevant given the topic and fit within the given category. The list completion task is slightly different from the previous task, and aims at adding entities of the same type to a small

sample seed set of entities. Again, we also have the topic and category available, but as additional information we get a list of examples: one or more entities that fulfill the constraints.

In our participation we use a generative language modeling approach to model both tasks, since this offers us a theoretically sound and understandable way of dealing with the problem at hand. A large portion of this paper is directed to the modeling of entity ranking and the estimation of the various components this model offers us. We submitted a total of six runs, again with a focus on the entity ranking task (four runs).

The remainder of this section introduces the modeling of the entity ranking task in Section 2.1, and of the list completion task in Section 2.2. Next, we discuss the estimation of the various components of both models in Section 2.3, and we conclude with some notes on the experimental setup and submitted runs in Sections 2.4 and 2.5, respectively. Since evaluation results are not available at the time of writing, we skip the results, analysis, and conclusions.

2.1 Modeling Entity Ranking

Entities are ranked by their probability of being relevant given a query q and a set of categories C, that is $P(e|q, C)$. We assume that q and C are conditionally independent, moreover, it is also assumed that each of the categories $c \in C$ are mutually independent. Formally:

$$\begin{aligned} P(e|q, C) &= P(e|q) \cdot P(e|C) \\ &= P(e|q) \cdot \prod_{c \in C} P(e|c). \end{aligned} \tag{1}$$

To estimate the probability of an entity given the query, $P(e|q)$, we apply Bayes' rule, then drop the denominator, $P(q)$, which is a constant for all entities, and thus, does not influence the ranking:

$$P(e|q) \propto P(q|e) \cdot P(e). \tag{2}$$

Here, $P(q|e)$ expresses the probability that q is generated by entity e, and $P(e)$ is the *a priori* probability of an entity being relevant (independent of the query). For the sake of simplicity, $P(e)$ is assumed to be uniform, and is not included in the equations from now onwards.

We infer an entity model for each entity e, such that the probability of a term given the entity model is $P(t|\theta_e)$. This model is then used to predict how likely the entity would produce query q. Each query term is assumed to be sampled identically and independently. Thus, the query likelihood is obtained by taking the product of the individual term probabilities across all terms in the query:

$$P(q|\theta_e) = \prod_{t \in q} P(t|\theta_e)^{n(t,q)}, \tag{3}$$

where $n(t, q)$ denotes the number of times t is present in q. Putting together our choices so far (Eqs. 1, 2, and 3) we obtain the following:

$$P(e|q, C) = \prod_{t \in q} P(t|\theta_e)^{n(t,q)} \cdot \prod_{c \in C} P(e|c). \tag{4}$$

For computational reasons, we move to the log domain, and use the following formula for ranking entities:

$$\log P(e|q,C) \propto \left(\sum_{t \in q} P(t|\theta_q) \cdot \log P(t|\theta_e) \right) + \sum_{c \in C} \log P(e|c). \quad (5)$$

Note that $n(t,q)$ has been replaced with $P(t|\theta_q)$, where θ_q is referred to as the *query model*. This allows us more flexible weighting of query terms. Three important components remain to be defined: the entity model θ_e, the query model θ_q, and the probability of an entity given a category $P(e|c)$. These will be introduced in the following sections.

2.2 Modeling List Completion

The list completion task is modeled similarly to the entity ranking task, with the addition that the probability of the entity is also conditioned on a set of example entities, E. We assume that example entities are conditionally independent from the query and the categories, as well as mutually independent from each other. Formally:

$$P(e|q,C,E) = P(e|q,C) \cdot \prod_{e' \in E} P(e|e'). \quad (6)$$

Again, we perform this computation in the log domain:

$$\log P(e|q,C,E) = \log P(e|q,C) + \sum_{e' \in E} \log P(e|e'). \quad (7)$$

The estimation of $P(e|q,C)$ has already been discussed in Section 2.1. A new component to be defined is the probability of an entity e given entity e': $P(e|e')$. In other words, this probability expresses the similarity of two entities.

2.3 Estimating the Components

In this section we detail how various components of the models introduced in the previous sections are estimated. Specifically, we discuss the implementation of the entity model θ_e, the query model θ_q, the probability of an entity given a set of categories $P(e|C)$, and finally, the probability of an entity given another entity $P(e|e')$.

Entity Model The entity is represented as a multinomial probability distribution over terms. To estimate $P(t|\theta_e)$ we smooth the empirical entity model with the background collection to prevent zero probabilities:

$$P(t|\theta_e) = (1 - \lambda_e) \cdot P(t|e) + \lambda_e \cdot P(t) \quad (8)$$

Since entities correspond to Wikipedia articles, this way of modeling an entity is identical to constructing a smoothed document model for each Wikipedia page. The choice of the smoothing parameter λ_e is discussed in Section 2.4.

Query Model Our baseline query model $P(t|\theta_q)$ is set to $n(t,q) \cdot |q|^{-1}$, where $n(t,q)$ is the number of occurrences of term t in query q, and $|q|$ is the query length. Essentially, the probability mass is distributed uniformly across query terms. Since this representation of the query is quite sparse, we would like to add more terms to the original query. By mixing new terms and original query terms, we end up with the following equation:

$$P(t|\theta_q) = (1 - \lambda_q) \cdot P(t|\hat{q}) + \lambda_q \cdot P(t|q), \tag{9}$$

where $P(t|\hat{q})$ is the probability of the term given the expanded query.

In [1] various methods are introduced for constructing expanded query models by sampling terms from a set of example documents (complementing the textual query). Based on the information provided with the topic statement, we have three straightforward ways of applying these methods to our current scenario, by (i) treating all entities belonging to the target categories as examples (both for entity ranking and list completion), (ii) employ a blind-relevance feedback approach, in which we perform a baseline run and look at the categories which are assigned to the 10 highest-ranked entities (category feedback for entity ranking), or (iii) using the example entities (only list completion). Specifically, we use the best performing method, maximum likelihood (ML), from [1] to estimate the expanded query model $P(t|\hat{q})$.

Entity-Categories Probability The probability of an entity e given a set of target categories C, $P(e|C)$, is computed as follows. Let cat(e) denote the set of categories e is assigned to. The overlap ratio between cat(e) and the set of target categories C is used as an estimate of $P(e|C)$:

$$P(e|C) = \frac{|\text{cat}(e) \cap C|}{|C|}, \tag{10}$$

where $|C|$ is the size of the set of target categories. We experiment further with this way of estimating $P(e|C)$, by introducing a parameter δ to control the weight of the overlap between the two sets C and cat(e) and dropping the term in the denominator:

$$P(e|C) = \delta \cdot |\text{cat}(e) \cap C|. \tag{11}$$

Based on initial experiments we set $\delta = 6$.

Further, we hypothesize that the target categories for each topic, as used in Eqs. 10 and 11, are not exhaustive. Therefore, in order to amend this set of categories, we apply a simple expansion strategy. We leverage the hierarchical structure of the Wikipedia categories and add the categories below each target and expanded category up to a certain depth. Based on preliminary experiments, we set the maximum depth to three.

Entity-Entity Probability Our model for the list completion task involves the estimation of the similarity between two entities. This is expressed as $P(e|e')$, the probability of an entity e given another entity e' (see Eq. 6). We estimate this probability based on set overlap between the categories assigned to each of the entities. To this end, we employ a standard set-based similarity measure, Dice's coefficient, calculated as follows:

$$P(e|e') = \frac{2 \cdot |\text{cat}(e) \cap \text{cat}(e')|}{|\text{cat}(e)| + |\text{cat}(e')|}, \tag{12}$$

where cat(e) and cat(e') are the set of categories assigned to entities e and e', respectively.

2.4 Experimental Setup

Document Representation Besides representing the entity (Wikipedia page) by its entire textual content (refered to as *full representation*), we opted for a second representation. Assuming that most valuable information on a Wikipedia page is presented at the beginning of the article, we select only the first paragraph of each article. This paragraph is the new representation of the entity (refered to as *paragraph representation*). For reasons of comparability, we use the full representation in almost all cases.

Document Preprocessing Document preprocessing consisted of removing stopwords only. Besides the "standard" English stopwords, we added several Wikipedia-specific stopwords to the stopword list (e.g. *disambiguation*, *category*, and *stub*).

Smoothing Parameter For the smoothing parameter λ_e in Eq. 8, we set λ_e equal to $\frac{|e|}{\beta+|e|}$, where $|e|$ is the length of the entity in number of terms. Essentially, the amount of smoothing is proportional to the length of the entity (and is like Bayes smoothing with a Dirichlet prior [2]). If there is very few content available in the entity then the model of the entity is more uncertain, leading to a greater reliance on the background probabilities. We set β to be the average entity length, i.e. $\beta = 409$ for the full representation and $\beta = 42$ for the paragraph representation.

Query Modeling Parameter For the construction of the new query model (Eq. 9), we need to set λ_q and decide on the number of terms in the new query. For the entity ranking task, in which we select our expansion terms from the category feedback approach, we set $\lambda_q = 0.5$ and select the 20 terms with the highest probability. For the list completion task we set $\lambda_q = 0.2$ and again select the top 20 terms to be included in the new query.

2.5 Submitted Runs

The following lists our six submitted runs and the configuration used for each run (note that all runs use the full representation, unless stated otherwise).

6_UAms_ER_T_baseline: Our baseline run using Eq. 3.

3_UAms_ER_T_overlap: Overlap run using Eq. 5; we estimate $P(e|C)$ as in Eq. 11, and use an expanded category set C up to depth three.

4_UAms_ER_T_cat-exp: Expanded overlap run; similar to run *3_UAms_ER_T_overlap*, except that we model the query according to Eq. 9, where expansion terms are selected using the category feedback method. We select the top 2 categories and use the entities within these categories as examples.

1_UAms_ER_T_mixture: Mixture run; we construct two runs using Eq. 5, one on the full representation and one on the paragraph representation. Each run estimates $P(e|C)$ as in Eq. 11, and uses an expanded category set C up to depth three (paragraph representation) or two (full representation). Both runs are combined using a linear rank combination with a weight of 0.1 for the paragraph representation and 0.9 for the full representation.

The remaining two runs are used for the list completion task:

5_UAms_LC_T_baseline: Our baseline run using Eq. 7; we model the query according to Eq. 9 and select expansion terms from the provided example entities.

2_UAms_LC_T_dice: Our overlap run; similar to run *5_UAms_LC_T_baseline*. We estimate $P(e|e')$ using Eq. 12 and $P(e|C)$ as in Eq. 11, and use an expanded category set C up to depth two.

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. The 2006 Wikipedia collection is used as the development and test data, which contains the ground truth of linked documents. This year's LTW track consists of two sub-tasks, namely, the file-to-file (f2f) link generation task and the anchor-text to Best Entry Point (BEP) link generation task. We participated in the f2f task and submitted three runs. The task is formulated as follows: a set of 6600 Wikipedia articles are randomly picked from the collection as topic pages; the participants are supposed to discover at most 250 incoming and 250 outgoing links between the topic pages and the rest of the collection. In the following sections, we introduce our approaches for both incoming links and outgoing links, followed by the description of the runs we submitted, as well as the experimental settings we applied to our runs.

3.1 Outgoing Links

Our approach to identifying outgoing links can be seen as a two-step procedure: anchor text detection (where to start a link) and target identification (which page should be linked). Although in the f2f task, the exact position of the anchor text is not required, the identified anchor texts strongly indicate the target pages to be linked.

Anchor Likelihood Ratio For anchor text detection, we introduce the anchor likelihood ratio measure (ALR) which we use to rank the n-grams in a text on being an anchor text. Since the Wikipedia articles are well-structured and interconnected, the existing links provide an indication of the patterns of linked pages. Mihalcea and Csomai [3] proposed the link probability measure which measures the likelihood of a word sequence being an anchor text by calculating the ratio between the number of times a word sequence is used as an anchor text and the number of times this word sequence occurs in the collection. The likelihood estimation is a reasonable measure and proved

to be useful. However, the value of the likelihood is a continuous number between 0 and 1, which needs a threshold to determine wether the word sequence is an anchor text or not. By modifying this measure, we try to model it without a magic threshold.

For a given word sequence w, we assume that it is sampled from two different underlying models: the anchor model θ_A and the background collection model θ_C. For selecting the model for generating the word sequence, we take the likelihood ratio between the two models, which is formulated as:

$$ALR_w(\theta_A||\theta_C) = \frac{P(w|\theta_A)}{P(w|\theta_C)} \tag{13}$$

Given Eq. 13, it is obvious that the larger the value of the likelihood ratio, the more likely it is that the word sequence w is generated by the anchor model. Particularly, when $ALR_w > 1$, it expresses that the anchor model is preferred over the collection model, and therefore we can obtain a non-magic threshold at $ALR_w = 1$.

Target page identification - Title Field Evidence For target page identification, we follow a language modeling approach. In Wikipedia, the title of a page is the main concept of the page and is usually the same as or similar to the anchor text linked to it. Therefore, a straightforward way of modeling this relationship would be to assume that a given anchor text can be generated by the language models that generate the titles. Thus the problem boils down to estimating the probability that the given anchor text A is generated from a given title model θ_t by applying the Bayes' Theorem.

$$P(\theta_t|A) = \frac{P(A|\theta_t)P(\theta_t)}{P(A)}, \tag{14}$$

where the $P(\theta_t)$ is assumed to be uniformly distributed and $P(A)$ is the same for each anchor and is usually dropped since it does not affect ranking. For estimating the probability $P(A|\theta_t)$, we assume each term in the anchor text to be independent and estimate the maximum likelihood (ML) of the anchor term w generated by the title model:

$$P(A|\theta_t) = \prod_{w \in A} P(w|\theta_t) \tag{15}$$

To avoid zero probabilities, we smooth $P(w|t)$ with the background collection of all page titles C_T using the Jelinek-Mercer method to obtain $P(w|\theta_t)$:

$$P(w|\theta_t) = (1 - \lambda) \cdot P(w|t) + \lambda \cdot P(w|C_T), \tag{16}$$

where

$$P(w|\theta_t) = \frac{n(w,t)}{\sum_{w'} n(w',t)} \tag{17}$$

In this equation, $n(w,t)$ and $n(w',t)$ are the number of times terms w and w' occur in a candidate target page's title t.

Target page identification - Topic Page Content Evidence Since most Wikipedia pages are short, it is very likely to end up with equal probabilities for different target candidates, especially in cases where disambiguate pages are involved. In order to solve this disambiguation problem, we try to incorporate an additional evidence source, the topic page content evidence. The assumption behind this is that if a candidate target page is the real target page for the given topic page, the content of the topic page should somehow relate to the terms in the title field of the candidate target page. Based on this assumption, we model the problem as the probability that the language model of the topic page θ_d generates the title of the candidate target page t. Similarly as before, we apply Bayes' Theorem to estimate the probability that a specific model θ_d is selected.

$$P(\theta_d|t) = \frac{P(t|\theta_d)P(\theta_d)}{P(t)}, \tag{18}$$

where $P(t)$ is ignored for ranking, and $P(\theta_d)$ is assumed to be uniformly distributed. The same ML estimate is applied to calculate $P(t|\theta_d)$:

$$P(t|\theta_d) = \prod_{w \in t} P(w|\theta_d) \tag{19}$$

Again, JM smoothing is applied to avoid zero probabilities.

3.2 Incoming Links

Our approach for identifying incoming links is quite straightforward. We experiment with two types of methods: The first method is based on exact matching of titles. We get the pages that contain the exact matches of the title of the topic page and select randomly 250 pages as linked pages. In this case, we simply ignore the context of the matched terms and assume that the existence of a link is context-independent. The second method is a rank-based approach. In this case, we perform a retrieval run on the candidate source pages, and use the title of the topic page as query. The top 250 pages are selected as the linked pages. Further thresholding on these 250 pages is also explored, and we describe this later in detail in our submitted runs section.

3.3 The LTW Runs

We submitted three runs to the LTW track. Each of the runs tries to explore the different research questions.

ltw01 : We use this run as our baseline run.
Outgoing links: We select word sequences as anchor texts whose ALR is larger than 1; use the anchor text as query on the title field of the Wikipedia collection and retrieve target pages; use the top-ranked page as the target page.
Incoming links: We use the title of the topic page as query and retrieve the top 250 pages from the collection as the source pages that are linked to the given topic.

ltw02 :We compare this run with ltw01. For outgoing links, we use this run to test if re-ranking would help disambiguate the target pages. For incoming links, this run tests how much the term linked with a topic page is related with the content of the source page.
Outgoing links: We select anchors with ALR larger than 1; retrieve the target page whose title matches (exact or partial) the anchor text; re-rank the target pages according to the likelihood of the target title in the topic page (i.e., Eq. 18), and select the top-ranked page as the target page.
Incoming link: use the topic title to find exact matches in the collection, select randomly 250 pages as the incoming source page.

ltw03 : This run checks an assumption: the linked pages should be semantically close if the links are not generated automatically by Wikipedia (i.e., country, year, etc.). This run does not try to identify anchor text at all.
Outgoing links: We use the topic title as query to retrieve 250 candidate target pages, rank them by cosine similarity to the topic page, and select the target pages whose similarity is larger than 0.15.
Incoming links: We use the same strategy as that of the outgoing links, but select the source pages whose similarity is larger than 0.026.

Here, 0.15 is the average cosine similarity between linked pages that are sampled from the collection; 0.026 is the threshold for "exceptionally" high similarity between two random pages. Different thresholds mean we would like outgoing links to emphasize on precision, and incoming links to emphasize on recall.

3.4 Experimental Settings and Results

In this section, we discuss the experimental settings in generating our runs and present the evaluation results, followed by a discussion of the initial observations.

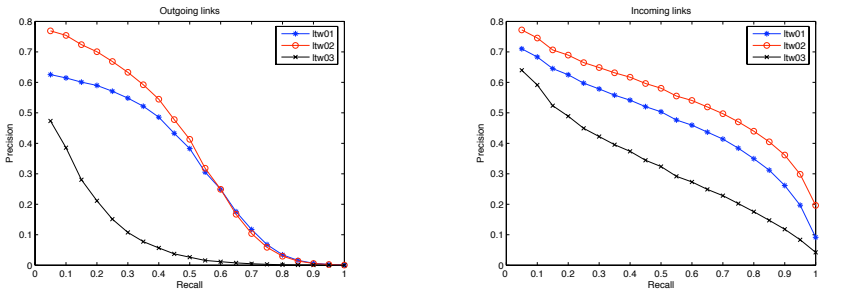
We use the content only Wikipedia pages as input. All the XML-tags are removed and the XML-structures of the documents are ignored in our experiments. For preprocessing, we use a Porter stemmer for both topic pages and target pages without stop words removal. All the topic pages are virtually deleted from the collection. For all retrieval experiments, we use the Lemur toolkit [1]. Smoothing parameter λ (Eq. 16) is heuristically set to 0.1 for all experiments (i.e. the background statistics have little impact).

Table 1 lists the evaluation results of the submitted runs in terms of MAP. Figure 1 shows the precision-recall plots for both incoming and outgoing links. In terms of MAP, we see that for both outgoing links and incoming links, run *ltw02* has better performance than *ltw01*. For outgoing links, the re-ranking with content evidence does help improve the performance, especially in early precision. This shows in 1(a). For incoming links, the fact that *ltw02* is better than *ltw01* suggests that simple title match works better than a rank-based method. Run *ltw03* performs consistently worse than the other two runs. However, early precision for this run is not that bad, suggesting that the similarity between pages could be a reasonable feature.

[1] http://www.lemurproject.org

Runs	MAP of outgoing links	MAP of incoming links
ltw01	0.2879	0.4800
ltw02	**0.3474**	**0.5249**
ltw03	0.1041	0.3345

Table 1. Results of submitted runs



(a) Precision-Recall plot for outgoing links (b) Precision-Recall plot for incoming links

Fig. 1. Precision-Recall plots for the submitted runs

4 Conclusions

We described our approaches, submissions, and initial results of this year's INEX participation. For the Entity Ranking track our focus was on the entity ranking and list completion tasks, and our chief aim was to develop a general language modeling framework to model these. Given the models we developed, we are left with plenty of choices on how to estimate the various components these models offer. For most of the components we applied simple options, that mainly make use of the category information that is available in Wikipedia. More elaborate ways of estimating the components of our models are left to future work and depend on the results of this year's participation. Results and conclusions are postponed in this paper due to the lack of evaluation results at the time of writing.

As to the Link-the-Twiki track, we submitted three runs for the File-to-File task designed to examine different features for file-level link generation. For outgoing links we based our runs on a two-step procedure: anchor text detection and target page identification. For anchor text detection, we use the anchor likelihood ratio (ALR), and for target identification, we apply a language modeling approach with different sources of evidence (i.e. title field and topic page content). Results show that the title field evidence alone is an important feature, and early precision can be improved by adding content evidence. For incoming links, we apply two very simple methods: exact matching and retrieval using the title of the topic page. We also submitted a run that tests if simple similarity between pages is sufficient as an indication of a link. The result shows that this is a reasonable feature, but on its own it is not powerful enough for link detection. Future work focuses on exploring more robust ways to model these features.

References

1. Balog, K., Weerkamp, W., de Rijke, M.: A few examples go a long way: constructing query models from elaborate query formulations. In: SIGIR '08. (2008)
2. Mackay, D.J.C., Peto, L.: A hierarchical dirichlet language model. Natural Language Engineering **1**(3) (1994) 1–19
3. Mihalcea, R., Csomai, A.: Wikify!: linking documents to encyclopedic knowledge. In: CIKM '07: Proceedings of the sixteenth ACM conference on information and knowledge management, New York, NY, USA, Acm (2007) 233–242

The Interactive Track at INEX 2008

Nils Pharo[1], Ragnar Nordlie[1] and Khairun Nisa Fachry[2]

[1]Faculty of Journalism, Library and Information Science, Oslo University College, Norway
nils.pharo@jbi.hio.no, ragnar.nordlie@jbi.hio.no
[2] 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 2008 *interactive track*. We introduce the methods used for data collection and the tasks performed by participants. Data collection is not yet completed thus it will be presented in more detail in the final proceedings.

Introduction

The INEX interactive track (iTrack) was run for the first time in 2004, repeated in 2005 and again in 2006/2007 (due to technical problems the tasks scheduled for 2006 were actually run in early 2007). 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 behaviour, 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.

In 2008, we wanted to preserve as much of the "common effort" quality of the previous years as possible. We invited the participants to participate in a minimum experimental effort using the system and data provided and described below. Within the framework of the track, participants could then design their own investigations under certain constraints, such as:

- The collection of documents was the same as the one used for the INEX ad hoc retrieval task, i.e., in 2008 the Wikipedia collection.
- The IR system developed for the 2006 track was made available for the participants to use, either alone or in comparison with participants' own system(s).

- Each participating site was responsible for recruiting a minimum of 8 (but preferably more) test persons to participate in the study as searchers.
- The participants should make their data available to all participating groups, and describe their collection process and experimental procedure so that the data can be interpretable for others.

In all, 7 institutions stated that they were interested in taking part in the 2008 iTrack experiments:

- Renmin University of China
- Kyungpook National University
- University at Albany, State University of New York
- University of Otago
- Microsoft Research
- University of Amsterdam
- Oslo University College

Tasks

For the 2008 iTrack we decided to design the experiment with two categories of tasks, from each of which the searcher should select one task. The original intention was to also give the searchers the opportunity to perform one self-generated task, but it was unfortunately not possible to implement this in the IR system. The two categories of tasks consist of fact-findings tasks (category 1) and research tasks (category 2). The tasks were generated to represent information needs that we believe are typical for Wikipedia-users. In addition we wanted the task to be complex enough not to be solvable from one individual article. In order to decrease learning effect, the order of task categories performed by searchers was rotated.

The fact-finding tasks:

1. As a frequent traveler and visitor of many airports around the world you are keen on finding out which is the largest airport. You also want to know the criteria used for defining large airports.
2. The "Seven summits" are the highest mountains on each of the seven continents. Climbing all of them is regarded as a mountaineering challenge. You would like to know which of these summits were first climbed successfully.
3. In the recent Olympics there was a controversy over the age of some of the female gymnasts. You want to know what the minimum age for Olympic competitors in gymnastics.

The research tasks

1. You are writing a term paper about political processes in the United States and Europe, and want to focus on the differences in the presidential elections of France and the United States. Find material that describes the procedure of selecting the candidates for presidential elections in the two countries.
2. Every year there are several ranking lists over the best universities in the world. These lists are seldom similar. You are writing an article discussing and comparing the different ranking systems and need information about the different lists and what criteria and factors they use in their ranking.
3. You have followed the news coverage of the conflict between Russia and Georgia over South Ossetia. You are interested in the historic background for the conflict and would like to find as much information about it as possible. In particular you are interested in material comparing this conflict with the parallel border conflict between Georgia and Abkhazia.

System design

The system used is a java-based retrieval system built within the Daffodil framework [1], which resides on a server at and is maintained by the University of Duisburg. The search system interface is quite similar to the one used in Task A of the 2005 and 2006 tracks.

The system returns elements of varying granularity based on the hierarchical document structure. The elements are grouped by document in the result list and up to three high ranking elements are shown per document. When a searcher chooses to examine a document the system shows the entire full text of the document with background highlighting for high ranking elements. In addition to this it shows a Table of Contents drawn from the XML formatting. To help searchers select query terms a box appears showing terms related to the current query, using mouse-over searchers can view the top-three contexts of the related terms. The searchers can also right-click on related terms to retrieve the top-ten contexts.

Document corpus

The document corpus used is the same as the one used in the 2006 itrack. It consists of more than 650,000 articles formatted in XML. In all, the corpus contains 4,6 GB of encyclopedia articles extracted from Wikipedia[1].

[1] For more information see Denoyier & Gallinari's SIGIR Forum article at http://www.acm.org/sigs/sigir/forum/2006J/2006j_sigirforum_denoyer.pdf

Online questionnaires

Before the experiment, the searcher is given a pre-experiment questionnaire, which collects demographics questions such as searcher's age, education and searching experience. Each search task is preceded with a pre-task questionnaire, which collects searcher's perceptions of the search task. After each task, the searcher is asked to fill out a post-task questionnaire. The post-task questionnaire contains questions to learn about the searchers use of and their opinion on various features of the system. The experiment is closed with a post-experiment questionnaire, which asks searcher's general opinion of the search system. The questionnaires data are logged in a database.

Relevance assessments

The system was designed to have searchers assess the relevance of each item they looked at. These could be the full article or article elements. We have chosen to use the relevance scale used in the 2006 interactive track. This is based on work by Pehcevski [2] and it balances the need for information on the granularity of retrieved elements, the degree of relevance and is fairly simple and easy to visualize [3]. The searchers were not forced to perform relevance judgments, but in the instructions they were told to "select an assessment for each viewed piece of information with regards to how you consider it to be of help in solving the task."

Logging

All search sessions are logged and saved to a database. The logs register the events in the session, the actions performed by the searcher as well as the responses from the system.

Data analysis

At the time of writing data collection was still going on and we expect to give a summary of the data collected at the workshop in Dagstuhl.

References

[1] 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.

[2] 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.

[3] 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.

Link-the-Wiki: Performance Evaluation based on Frequent Phrases

Mao-Lung (Edward) Chen
Faculty of IT
Queensland University
of Technology
Brisbane, Australia
chen@student.qut.edu.au

Richi Nayak
Faculty of IT
Queensland University
of Technology
Brisbane, Australia
r.nayak@qut.edu.au

Shlomo Geva
Faculty of IT
Queensland University
of Technology
Brisbane, Australia
s.geva@qut.edu.au

Abstract. In this paper, we discuss our participation to the INEX 2008 Link-the-Wiki track. We utilized a sliding window based algorithm to extract the frequent terms and phrases. Using the extracted phrases and term as descriptive vectors, the anchors and links are recognized efficiently. Results show that we were able to find the relevant links successfully.

1 Introduction

With the information boom on the internet, there are many encyclopaedia-like websites gathering and sharing knowledge. One of the leading website is Wikipedia, which is a repository written and contributed by peer anonymous internet users. With the rich articles and features in Wikipedia, the INEX organization collected the documents and articles into INEX Wikipedia corpus, which is presented in XML format. The corpus is large in size and useful for various ranges of information retrieval and data mining / text mining researches.

One of the research tracks organized by INEX is Link-the-Wik, which was introduced on 2006 [1]. The objective of this track is to automatically discover the hyperlinks among Wikipedia web pages.

This Link-the-Wiki track offers many interesting challenges. One of them is related to the size and nature of the Wikipedia data corpus. This corpus has more than 659,000 documents. The file size is more than 5GB in XML files. The challenge includes performance on large dataset, handling high dimensional, complex and noise-full data source.

This research utilises frequent phrases for link discovery. Firstly, this research attempts to reduce the size, complexity and dimensionality of the dataset by extracting the descriptive and frequent terms and phrases from the corpus. Secondly, this research discovers the hyperlinks between Web pages, according to the extracted frequent phrases and terms.

This report is going to discuss the sproposed approach and the four stages of this research. In the Data Pre-Processing section, data cleaning step including stop-words removal and stemming will be reviewed. In the Frequent Phrase Extraction section, the extracting algorithm will be introduced and explained. The rationale of both incoming and outgoing links will be discussed in the Links Discovery section. The Experiments and Discussion section will look at the results of these experiments. The last section of this paper is the Conclusions and Future Work, which summaries this research and offers some future extensions and applications of this research.

2 Overview of the Proposed Approach

Figure 1 illustrates the proposed approach undertaken in this research. It includes four main stages including data preparation, frequent phrase recognition, link discovery and validation.

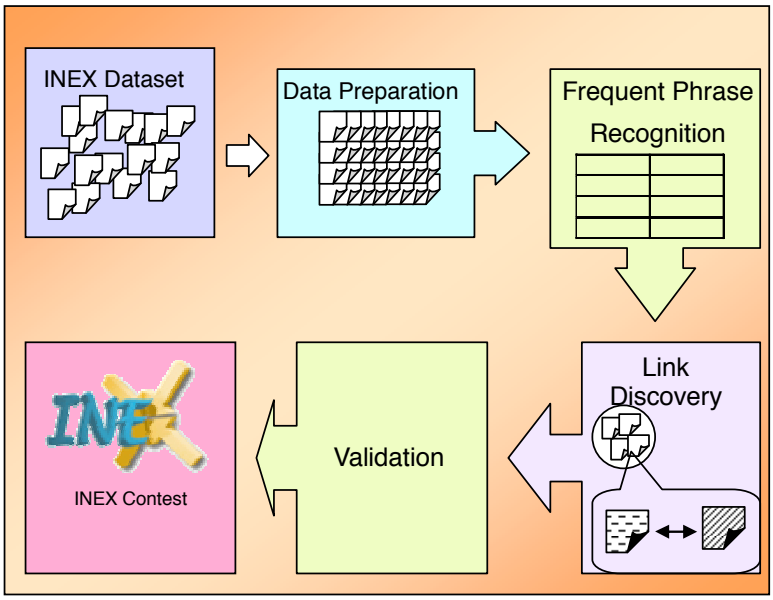


Fig. 1. The proposed approach

2.1 Data Preparation

In the first stage of this approach, data cleaning, transformation and preparation are performed. The Wikipedia documents in the INEX corpus require a series of data cleaning process to get them ready for data mining.

All the 659,000 documents of the Wikipedia corpus are incorporated into a relational database. By gathering all the documents into data tables in a database, all the Wikipedia articles will be well aligned. Relational databases, such as Microsoft SQL Server, Oracle and MySQL are appropriate selections to reside the data.

After the articles have been arranged into database, the following pre-processing would be ideally performed and take the advantages of database. By running loops on the data records to handle each document, data parsing, word stemming and stop-words removal will be accomplished at this stage.

2.2 Frequent Phrase Recognition

Initial data preparation steps including stop-word removal and stemming are able to eliminate a certain amount of noise and reduce the size of the corpus.. However, the database is still very large in size. In order to further reduce the size and complexity, the second stage in this research employs an algorithm to recognise and extract the frequent phrases. A frequent phrase extraction algorithm has been developed to access the database, to extract the frequent phrases from each single document and to store the extracted phrases into database again for each document. In other words, this frequent phrase recognition step is to extract the descriptive phrases as vectors from the individual documents. It is hoped that after this step, database size and article complexity would be remarkably decreased.

2.3 Link Discovery

With each document stored as frequent terms and phrases, the link discovery step becomes straightforward. Each orphan document is processed to recognize the appropriate anchors according to the existing frequent phrases. Moreover, the links of the recognized anchors are filtered and selected according to the frequent phrases present in database for each document.

3 Data Pre-processing:

Similar to a data mining task, data pre-processing is the first step of the proposed approach. In this research, all data is organized into a database. The first pre-processing step was to eliminate the XML tags from the input XML document and

transfer it into a plain text article. In addition, any word which is less than and equals to 3 characters was deleted during parsing.

The next step is stop-word removal. There were some difficulties encountered in using the standard and common stop-word lists to identify the stop-words. These lists cover a wide range; as a result, some of the meanings were lost or changed after the stop-words removal. For example, the word "new" is covered in these lists. For some articles which have the phrase "New York", it became only "York" with the removal of the word "New". Apparently, "New York" is totally different from "York". The solution to this problem was to manually review the list of stop-words. If a keyword that can be a part of a meaningful phrase should be excluded from the stop-words list.

The last step of data pre-processing is to stem the words. This research employed a well-known stemming algorithm by Porter M.F. [2]. The Porter Stemmer algorithm can remove the suffix from words in English. It is widely employed in information retrieval and text mining researches and applications.

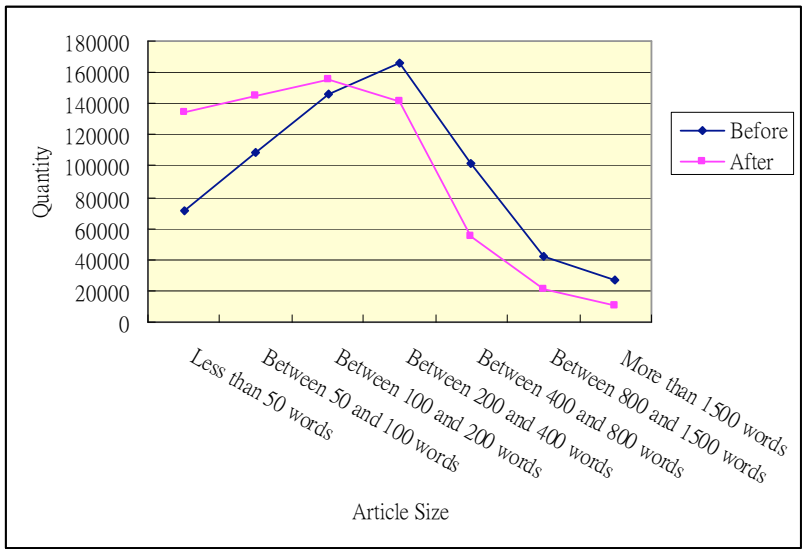


Fig. 2. Article size before and after pre-processing

As shown in figure 2, the article size was effectively reduced by pro-processing. After pro-processing, the distribution of article size was drift to left and distributed more evenly than before pre-processing. The average size of articles was reduced almost 40% after pre-processing as shown in table 1. It was average 389 words in a document before pro-processing. It has condensed to average 234 words in a document.

Table 1. The average article size befor and after pre-processing

	Before	After	Reduced
Average Article Size	389	234	39.8%

4 Frequent Phrase Extraction

A sliding-window based algorithm is used to extract the frequent phrases from each single document. This algorithm recognizes and extracts the frequent terms and phrases from each document independently in the corpus.

Let D be the set of documents in the Wikipedia corpus. D At this point we assume that each document is independent from each other.

$$D = \{ d_1, d_2, d_3, \ldots\ldots, d_n \} \quad (1)$$

Considering each document independently, a set of frequent phrases from each document is extracted. Frequent phrases are extracted from a document at the level of sentences and paragraphs. Figure 3 shows the process of frequent phrase extraction in which a document is modeled as a set of sentences. The algorithm applies a window on a number of words (several window sizes are used during the experiments). Using the moving window, 1-term, 2-terms to n-terms frequent phrases are extracted (several n-sizes are used in experiments). Output of this algorithm is a set of 1-term to n-terms frequent phrases which belongs to that particular document.

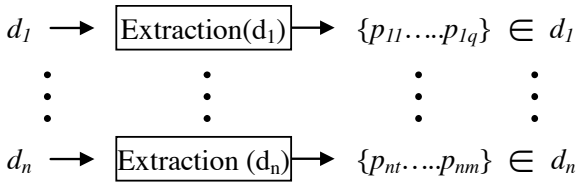


Fig. 3. The inputs and outputs of frequent phrase extraction

5 Link Discovery

The link discovery task was to recognize the anchors in a set of orphan documents and the appropriate incoming and outgoing links to these orphan documents via these anchors. An incoming link is the link that from the relevant documents in Wikipedia documents link to this orphan document. In contract, an outgoing link is a link which anchors in this orphan document and refers to a relevant document. We utilized the extracted frequent phrases to recognize the anchors and identify both in-coming and out-going links.

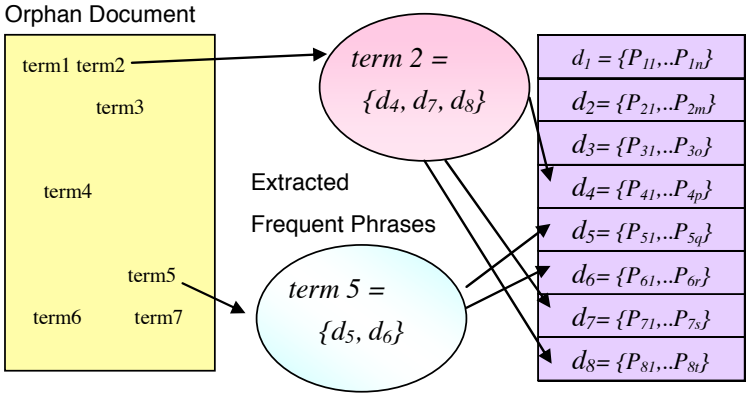


Fig. 4. The link discovery design with extracted frequent phrases.

The link discovery was designed by exploiting the extracted frequent phrases as shown in figure 4. Each orphan document is scanned for a common term or phrase identified in the corpus. Each term (or combinations of the terms in the window) of an orphan document is matched with the extracted frequent phrases. If a term (or phrase) is matched with a frequent phrase, the link can be created by tracking back to the original document. The sections below explain the detail process of identifying outgoing and incoming links.

5.1 Outgoing Links

The first task in identifying outgoing links of an orphan document is to recognize the anchors in the document which are phrases. A phrase is composed of multiple single terms and is a set of element terms. Consider the following example. Assume that the orphan document has an anchor that is "*Australian Open Tennis Championship*". This 4-terms phrase, P_1, is a set of 4 elements, including t_1=

"*Australian*", t_2= "*Open*", t_3= "*Tennis*" and t_4 is "*Championship*". The first challenge of outgoing link discovery is how to identify "Australian Open Tennis Championship" as an anchor phrase present in the orphan document.

$$P_1 = \{t_1, t_2, t_3, t_4\} = \{\text{“Australian”, “Open”, “Tennis”, “Championship”}\} \quad (2)$$

Let us first consider the outgoing link discovery without the assistance of the Frequent Phrase List. The link discovery algorithm considers t_1 ("*Australian*") individually and search the link for t_1. As shown in figure 5 (left part), the program can only pick up *t1 ("Australian")*, *t2 ("Open")*, *t3 ("Tennis")* and *t4 ("Championship")* individually. Without the Frequent Phrase List, all these terms are independent from each other and there exist no relationship among them.

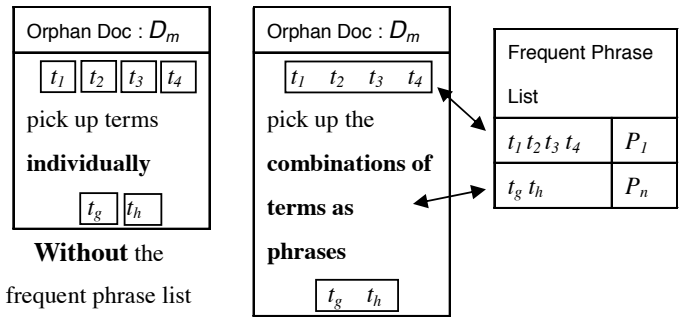


Fig. 5. Comparing with / without the assistance of frequent phrase list.

In contrast, with the assistance of extracted frequent phrases, the procedure recognizes the anchors $\{t_1, t_2, t_3, t_4\}$ as a single phrase. As shown in figure 5 (right part), $\{t_1, t_2, t_3, t_4\}$ ("*Australian Open Tennis Championship*") in the orphan document D_m is recognized according to P_1 in Frequent Phrase List. In this example, t_g and t_h would also be identified as a phrase P_n. In other w rods, the relationships among the individual terms are identified and stored in the Frequent Phrase List. In this practice, the procedure achieved a simulation of natural language, and recognizes phrase anchors.

After the anchors have been recognized, the next task is locating the documents which contain information about this anchor. For example, the articles containing information about previous winners of "Australian Open Tennis Championship" would be a good candidate. By exploiting the Frequent Phrase List, this link discovery procedure executes a series of queries against the documents which contain the query phrase.

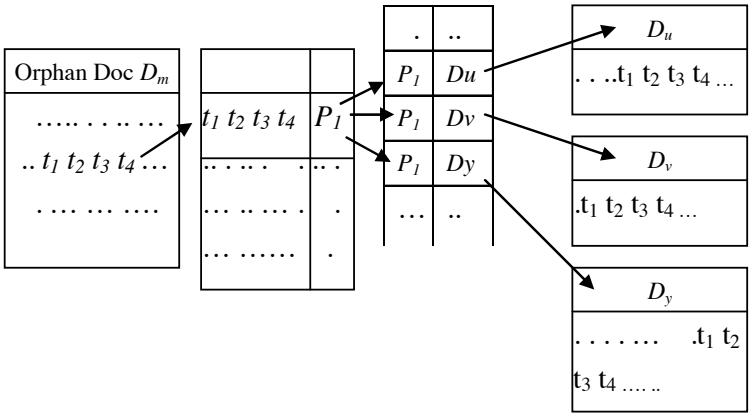


Fig. 6. Finding the outgoing links for anchor P_1

As shown in figure 6, the link discovery procedure first obtains the query phrase (anchor) P_1, and filters the list of documents. In this example, there are 3 documents returned by this query, including D_u, D_v, D_y. For example, if P_1 is the "*Australian Open Tennis Championship*" and the D_u may be an article regarding the "*The history of Australian Open Tennis Championship*".

With the sufficient information from previous steps, the phrases anchors would be recognized and the links for those anchors would be identified.

5.2 Incoming Links

The incoming link discovery uses the same concept as outgoing link discovery, but the direction is reversed. The first task is identifying anchors in the orphan document. The frequent *n*-terms phrases are extracted from the orphan document. As shown in figure 7, the frequent phrases, P_1, P_2 , P_3 and P_4 are extracted and viewed as descriptive vectors of this particular document D_m. The descriptive vectors are describing the topics of this orphan document. For instance, the possible frequent phrases from D_m are "*Australian Open Tennis Championship*", "*Melbourne Park*", "*hard court*", "*Grand Slam*". The combination of these frequent phrases represents and describes this orphan document to some extent.

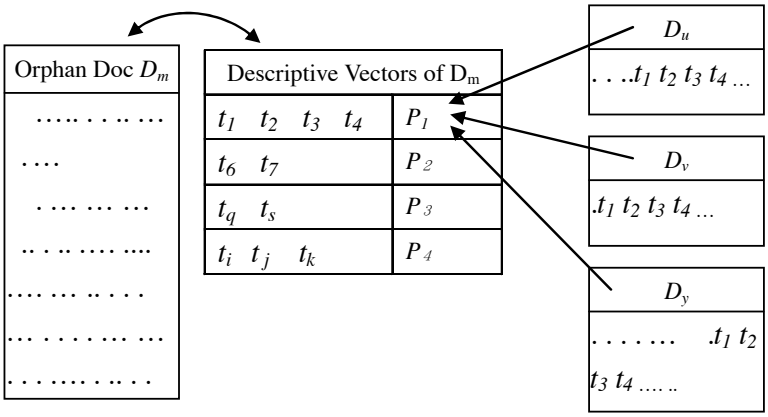


Fig. 7. Finding the incoming links for document D_m

The next step of incoming link discovery is to scan through the Wikipedia corpus and find out the articles which have information about the Australian Open Tennis Championship. These articles become the incoming links to this orphan document. In figure 7 for example, P_1 ("*Australian Open Tennis Championship*") is one of the descriptive vectors of document D_m. The last step of creating an incoming link is to store the information of incoming document ID $\{D_u, D_v, D_y\}$ to the orphan document D_m.

5.3 Links Discovered

After completing the link discovery procedures of incoming links and outgoing links, this research recognized sufficient and high quality links.

Table 2. The quantity of links discovered

Discovered Links	Minima	Average	Maxima
Overall	37	398	1133
Small docs (less than 500 words)	37	111	196
Medium docs (500 ~ 2000 words)	90	302	573
Large docs (more than 2000 words)	303	631	1133

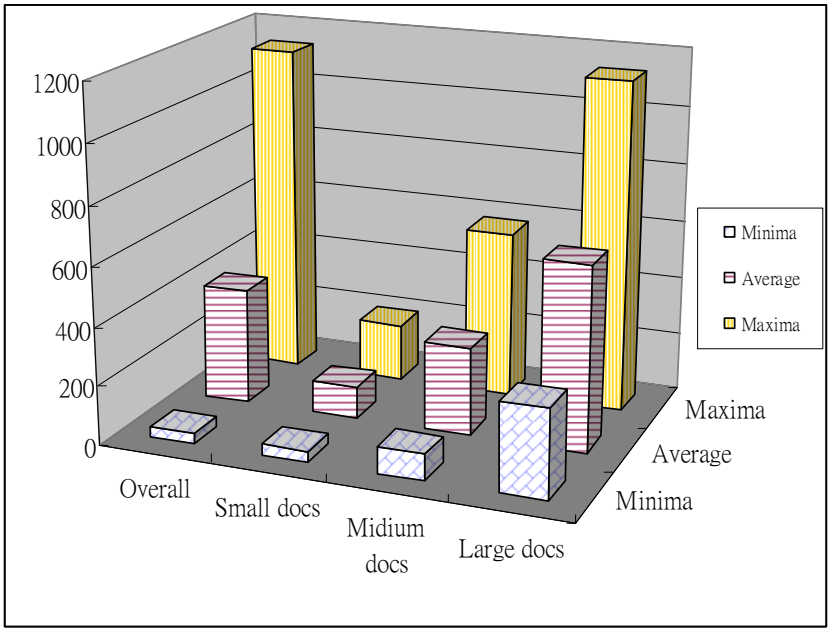


Fig. 8. The quantity of links discovered

As shown in table 2 and figure 8, the average of links discovered from small, medium and large documents are 111, 302 and 631, respectively. The INEX Link-the-Wiki contest can accept up to 250 incoming and 50 outgoing links for each orphan document. These illustration shows that the use of frequent phrases in identifying incoming and outgoing links are able to locate the sufficient quantity of links discovered.

6 Experiments and Discussion

Frequent *n*-terms phrases were extracted from the Wikipedia corpus to represent the original documents. The dimensionality and size was apparently reduced by the frequent phrase extraction. The original Wikipedia corpus was more than 5GB, while the total file size of extracted phrases was only 1.2GB. Table 3 gives some instances of recognised frequent phrases with frequency. Please note that the words are stemmed as the outcome of Port Stemmer [2].

Table 3. Some instances of the extracted frequent phrases

Doc ID	Freq	Phrase	Category
Doc 1	7	europ	1
Doc 1	9	violin	1
Doc 1	4	music instrument	2
Doc 1	6	standard pitch	2
Doc 1	12	finger position	2
Doc 1	4	violin mak techniques	3
Doc 1	5	beethoven violin concerto concert	4
Doc 1	6	correct violin string tun	4
Doc 1	3	tradit itali cremona stradivari pattern.	5

Table 4 shows that when the article size (word count) was increased, the average frequency of extracted phrase in that particular document was also raised as well. This shows that the extracted phrases were sufficient enough to describe the original document.

Table 4. The average frequency of every phrase

Document Size (Word Count)	Average Frequency
Less than 200	3
Between 200 and 400 words	8
Between 400 and 700 words	16
Between 700 and 1000 words	29
Between 1000 and 3000 words	62
More than 3000 words	214

By investigating the extracted Frequent Phrase List, some interesting observations were made. For example, as shown in table 5 and figure 9, the comparison between total phrases and unique phrases revealed the nature of English language. When looking at the 1-term phrases (single terms), the unique words are approximate 270,000 words. In other words, a dictionary with about 270,000 words would explain almost everything in this world. However, the 2-terms and 3-terms phrases may be more accurate to describe the meanings in English. Because the draw of 2-terms and 3-terms phrases seems going to a centralized trend, it reveals the nature of English that 2-terms and 3-terms phrases are the most descriptive and representative.

Table 5. The comparison between total phrases and unique phrases

	1-term	2-terms	3-terms	4-terms	5-terms
Total Phrases	6,410,434	3,782,563	2,504,056	1,100,244	410,025
Unique Phrases	270,826	2,127,492	2,023,640	948,361	359,144

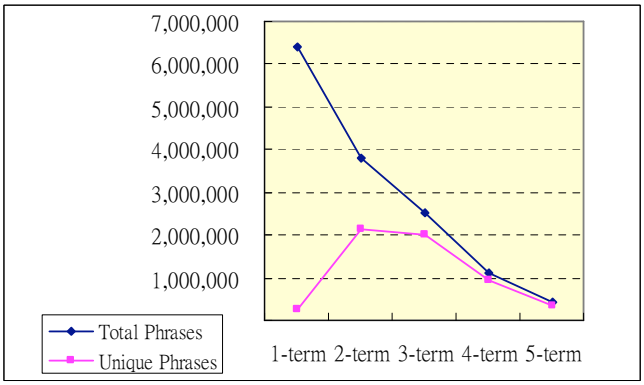


Fig. 9. The comparison between total phrases and unique phrases

7 Conclusions and Future Work

After conducting a series of experiments, results were found to support the hypothesis and assumptions made in the research. For example, the complexity and dimensions were effectively reduced by extracting the frequent phrases. The descriptive information collected from frequent phrases was sufficient to undertake the Link-The-Wiki link discovery tasks.

There are some possible future extensions of this research. From the perspective of text mining, the recognition of frequent phrases is a difficult issue.

On the other hand, hyperlink is a particular feature of hypertext and web pages. The hyperlinks discovered in the research were almost as meaningful as manually maintained. In the future researches, the precision of automatic link discovery would be improved. As a result, the generic link discovery method would benefit the huge amount of websites.

References

1. Trotman, A., Geva, S.: Passage Retrieval and other XML-Retrieval Tasks. Proceedings of SIGIR 2006 Workshop on XML Element Retrieval Methodology, Seattle, Washington, USA (2006) 48-50
2. Porter, M.F.: An algorithm for suffix stripping. Automated Library and Information Systems **14** (1980) 130-137
3. Kostoff, R.N., Tshiteya, R., Pfeil, K.M., Humenik, J.A.: Electrochemical power text mining using bibliometrics and database tomography. Journal of Power Sources **110** (2002) 163-176
4. Myat, N.N., Hla, K.H.S.: A Combined Approach of Formal Concept Analysis And Text Mining For Concept Based Document Clustering. IEEE/WIC/ACM International Conference on Web Intelligence 2005 (2005) 4
5. Girju, R., Badulescu, A., Moldovan, D.: Learning semantic constraints for the automatic discovery of part-whole relations. 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1. Association for Computational Linguistics, Edmonton, Canada (2003)
6. Hideo, J., Mark, S.: Retrieving descriptive phrases from large amounts of free text. 9th international conference on Information and knowledge management. ACM, McLean, Virginia, United States (2000)
7. Parisut, J., Worapoj, K.: Dimensionality reduction of features for text categorization. 3rd conference on IASTED International Conference: Advances in Computer Science and Technology. ACTA Press, Phuket, Thailand (2007)
8. Beil, F., Ester, M., Xu, X.: Frequent term-based text clustering. 8th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, Edmonton, Alberta, Canada (2002)
9. Yanjun, L., Soon, M.C.: Text document clustering based on frequent word sequences. 14th ACM international conference on Information and knowledge management. ACM, Bremen, Germany (2005) 293-294
10. Shen, D., Chen, Z., Yang, Q.Z., H., Zhang, B., Lu, Y., Ma, W.: Web-page classification through summarization.: Research and development in information retrieval, Sheffield: 27th ACM Int. SIGIR Conference. (2004) 242-249
11. Lei, Z., Debbie, Z., Simeon, J.S., John, D.: Weighted kernel model for text categorization. 5th Australasian conference on Data mining and analystics - Volume 61. Australian Computer Society, Inc., Sydney, Australia (2006)

CMIC@INEX-2008: Link-the-Wiki Track

Kareem Darwish[1]

[1] Cairo Microsoft Innovation Center,
Bldg B115, Smart Village
Km. 28 Cairo/Alexandria Desert Rd.
Abu Rawash, Egypt
kareemd@microsoft.com

Abstract. This paper describes the runs that I submitted to the INEX 2008 Link-the-Wiki track. I participated in the incoming File-to-File and the outgoing Anchor-to-BEP tasks. For the File-to-File task I used a generic IR engine and constructed queries based on the title, keywords, and keyphrases of the Wikipedia article. My runs performed well for this task achieving the highest precision for low recall levels. For the Anchor-to-BEP task, I used a keyphrase extraction engine developed in-house and I filtered the keyphrases using existing Wikipedia titles. Unfortunately, my runs performed poorly compared to those of other groups. I suspect that this was the result of using many phrases that were not central to articles as anchors.

Keywords: Document Linking, keyphrase extraction.

1 Introduction

This paper presents the experiments I conducted at the Cairo Microsoft Innovation Center (CMIC) for the INEX Link-the-Wiki track. I participated in the outgoing Anchor-to-BEP (A2B) and the incoming File-to-File (F2F) tasks only. For the A2B task, the task was reduced to an Anchor-to-File task by setting all the best entry points to 0. The focus for the A2B task was on the identification of possible anchors via performing keyphrase extraction on the text of the orphan pages. The keyphrase extraction algorithm that I used attempted to find all possible phrases, but neglected to determine if the keyphrases are central to the page. Such a determination of centrality is crucial for identifying good anchors. For the F2F task, I used generic information retrieval techniques without any special processing for Wikipedia articles.

This paper is organized as follows: Section 2 presents my keyphrase extraction technique and survey existing techniques; section 3 presents my methodology for the A2B and F2F tasks and reports on the results; and Section 4 concludes the paper.

2 Keyphrase Extraction

Identifying a word sequence consisting of one or more words that represents a *valuable concept* in text is an important NLP problem. Such valuable concepts, which are henceforth referred to as keyphrases, are often called keywords (if they are single words), collocations and multi-word expressions, and are assumed to obey "semantic non-compositionality, syntactic non-modifiability, and non-substitutability of components by semantically similar words" [2, 4]. Conventionally, keyphrases represent the central concepts in an article, and hence, a sequence of words can be a keyphrase in one article and not in another. Another application is identifying salient words or phrases that can serve as hypertext to link from one article to another. Depending on the desired level of linking, a sequence of words may not have to be central to the article, which was my target of the work presented in this paper. Perhaps the fundamental difference between the two aforementioned applications is that the first is concerned with the top *n* valuable word sequences and the other is concerned with "all" such word sequences.

Subsections 2.1 and 2.2 describe related work on keyphrase extraction and my keyphrase extraction algorithm respectively.

2.1 Related Work

Much effort has gone in defining what keyphrases and there variants are [2, 4]. There are many approaches to keyphrase extraction including approaches that use phrase occurrence counts and part of speech patterns and word collocations, in which words that co-occur with a mean distance that has low variance [4]. Other approach are based on supervised learning in which a classifier is trained on features such as phrase location in a text segment, a phrase term frequency and document frequency [6]. Another approach is based on constructing a directed graph where the nodes represent tokens from a reference corpus and weighted links between nodes indicating the count of subsequent occurrences in text. After constructing the graph, graph walks over the highest weighted links are used to extract keyphrases [3]. The list of approaches listed above is by no means exhaustive, but provides a flavor of the most popular approaches.

2.2 Keyphrase Extraction System

I developed a keyphrase extraction technique that uses supervised machine learning in which a support vector machine (SVM) classifier is trained on the following features:

1. The probability of sequence occurrence. Keyphrases are expected to have a high probability of occurrence. The probability is computed using a language model that is trained on a reference corpus. For this work, I trained the language model on the Link-the-Wiki Wikipedia corpus.
2. The unigram occurrence probability of head and tail words. Head and tail words are typically expected to be valuable words, which would indicate that they have a low occurrence probability. The probability is computed using a language model that is trained on Link-the-Wiki corpus.
3. The sequence probability of words between head and tail words. These words are assumed to connect between the head and tail words and hence should have high probability of occurrence. For example, for the keyphrase "Department of Energy" the connect sequence is just "of" and is expected to be a common sequence. Again the probability is computed using a language model that is trained on the Link-the-Wiki corpus.
4. The probability of Part-of-Speech (POS) sequence being a keyphrase. The probability is computed using a language model that is trained on a list of POS tagged keyphrases. The POS tagging was done using an in-house POS tagger.
5. The percentage of digits.
6. The percentage of words with upper case letters.
7. The percentage of words that are a part of noun phrase chunk. The chunking was done using an in-house chunker.
8. The number of words in a sequence.

My keyphrase extractor can be tuned to be recall or precision oriented. For the submitted runs, I tuned the system to be more precision oriented, because a user would generally be willing to tolerate missing hyperlinks but would generally not tolerate incorrectly assigned hyperlinks. My system achieves 40% recall when tuned to be approximately 99% precise, as measured on a reference corpus. An important feature that was omitted is a feature that measures the importance of the sequence in the article. Such a feature can be the term frequency of the term, some combination of the term frequency and inverse document frequency, or some other feature such as the binomial log likelihood ratio [1].

3 Approach to Link-the-Wiki and Results

For the F2F task, I used the Indri search toolkit for indexing and searching the Wikipedia articles. I used Indri with stop-word removal and no stemming or blind relevance feedback. Indri combines inference network model with language modeling [5]. I submitted two runs, namely CMIC_F2F_01 and CMIC_F2F_02. I used three items to construct queries, namely the titles of Wikipedia articles (with the phrase operator if a title was longer than 1 word), the keyphrases extracted from the articles, and the top 20 terms from each article as ranked by term frequency only. For CMIC_F2F_01 run, I constructed the queries from the titles and the keyphrases. As for the CMIC_F2F_02 run, I constructed the queries using titles, keyphrases, and top 20 terms. The resultant mean average precision for the CMIC_F2F_01 and CMIC_F2F_02 runs was 0.46 and 0.51 respectively. I suspect that adding more than 20 top terms, perhaps the entire content of the article, would have produced better results. It is also noteworthy that CMIC_F2F_02 achieved the highest precision among all the submitted runs for low recall levels (recall < 0.25), which suggests that my approach is more precision oriented and more suitable for generating a good small list of suggestions.

For the A2B task, I submitted one run, namely CMIC_LTW_01. For the run, I extracted the keyphrases in the orphan article and I filtered the keyphrases using the titles of the articles in Wikipedia articles that I was allowed to link to. The filtering involved allowing a keyphrase to match any title that was either an exact match or one that subsumes the keyphrase completely. For example, in article 100011 entitled Otago, the keyphrase "Firth of Clyde" was linked to the article 144233 entitled "Firth of Clyde", and the keyphrase "Free Church of Scotland" was allowed to link to articles 554606 and 909535 entitled "Free Church of Scotland" and "Free Presbyterian Church of Scotland" respectively. Unfortunately, my results were dismal with a mean average precision of 0.05. I suspect that my runs performed poorly because many of the keyphrases that were chosen to be anchors were not central to the articles and were hence deemed irrelevant by assessors.

4 Conclusion

This paper presented the CMIC runs to the INEX Link-the-Wiki track. I did well in the F2F task, but dismally in the A2B task. For the F2F, using generic information retrieval techniques in combination with keyphrase and key word extraction produced acceptable results with a mean average precision of 0.51, which is a little over 10% less than the best submission to the track (QUT9_GPXF2FnameInOut – mean average precision of 0.57). Further, my best submission achieved the highest precision levels, compared to all the other submissions, for low levels of recall (recall $\leq$ 0.25). This is desirable because one would want to link an orphan Wikipedia article to a small number of articles and precision for those articles needs to be as high as possible.

Perhaps better selection of keyphrases and using more keywords from the documents could have resulted in better results and warrants further investigation.
For the A2B task, my runs lagged significantly, mostly because I over generated anchors, many of them were not central to the articles. My keyphrase extraction algorithm needs to be modified to account for the centrality of the extracted keyphrase to articles. This can be achieved by retraining my classifier using an extra feature such as term frequency, inverse document frequency, binomial log ratio, or some other measure of centrality.

References

1. Dunning, T. E.: Accurate methods for the statistics of surprise and coincidence. Computational Linguistics, 19(1):61–74, (1993)
2. Evert, S.: The Statistics of Word Cooccurrences: Word Pairs and Collocations. Ph.D. Dissertation, Institut für maschinelle Sprachverarbeitung, University of Stuttgart, (2004)
3. Hammouda, K., Kamel, M.: Efficient Phrase-based Document Indexing for Web Document Clustering. IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 10, pp. 1279-1296, (2004)
4. Manning, C., Schutze, H.: Foundations of Statistical Natural Language Processing. Cambridge, MA: MIT Press, (1999)
5. Metzler, D., Croft, W. B.: Combining the Language Model and Inference Network Approaches to Retrieval. Information Processing and Management Special Issue on Bayesian Networks and Information Retrieval, 40(5), 735-750, (2004)
6. Turney, P. D.: Coherent keyphrase extraction via web mining. In Proceedings of IJCAI, Acapulco, Mexico, 434-439, (2002)

Stealing Anchors to Link the Wiki

Philipp Dopichaj, Andre Skusa, and Andreas Heß

Lycos Europe GmbH
Carl-Bertelsmann-Str. 29
P. O. Box 315
33311 Gütersloh
Germany
{philipp.dopichaj,andre.skusa,andreas.hess}@lycos-europe.com

Abstract. This paper describes the Link-the-Wiki submission of Lycos Europe. We try to learn suitable anchor texts by looking at the anchor texts the Wikipedia authors used. Disambiguation is done by using textual similarity and also by checking whether a set of link targets "makes sense" together.

1 Introduction

In this paper, we describe the Link-the-Wiki submission of Lycos Europe. Details about INEX and the Link-the-Wiki track are given elsewhere in these proceedings, so we do not repeat them here. In the following, we use *new text* to refer to the text which should be linked (conceptually, this is a text entered by a user of the platform without any links; the aim of the system is to support the user to find suitable links). We use *anchor text* or *anchor* to refer to the link label, that is, the clickable part of the text that links to a *target page*.

Our approach to the Link-the-Wiki task is based on that described by Itakura and Clarke [2]: All existing anchor texts from the training collection are indexed along with their link targets, and the new text is scanned for these anchor texts to find links.

The main difference is that we try to select the best-matching target dynamically whereas Itakura and Clarke use a static mapping from anchor text to target – the target is always the page most frequently referenced by the anchor. For example, in a text about computers, the anchor *Apple* is more likely to refer to the page *Apple Computers* than to the page *Apple Records*. We use heuristics based on text similarity and link structure to determine which of the potential targets is the most likely real target.

Finding outgoing links is done in the following steps:

1. The potential anchor texts are identified. The chosen anchor texts do not overlap, and each anchor text has one or more potential targets associated with it.
2. For each potential anchor text, a ranking of the potential targets *in the context of the new text* is performed. Furthermore, general statistical information obtained at indexing time – like absolute frequency – is used.

Our main focus is finding outgoing links, as opposed to finding incoming links from existing documents to the newly-added content. Outgoing links are determined in two main steps that will be described in the following sections:

1. Finding the parts of the texts that should serve as links to other documents (*anchor texts*).
2. Finding the correct target pages for the anchor texts in case of ambiguities.

The second point means that even if a given anchor text is known to refer to *some* other document, it is not necessarily known to *which* article it refers.

2 Preparations for Finding Outgoing Links

This section describes how potential anchor texts are found in the new text and also what index structures are needed to support this.

2.1 Finding Potential Anchor Texts

The first step toward identifying links in a new document is to find potential anchors; this is done by searching for occurrences of the training anchors in the new text. We give preference to longer anchor texts: For example, in the example text from figure 1a, we have the sequence *Mac OS X v10.2*. Potential anchors include *Mac*, *Mac OS*, and *Mac OS X v10.2*; here, the last one is the longest anchor text, so it is selected. In case of overlapping anchor texts, the anchor occurring earlier is selected.

Apple bundled a similar program, Sherlock 3 , with Mac OS X v10.2 .

(a) Input text.

[[Apple]] bundled a similar program, [[Sherlock 3]] , with [[Mac OS X v10.2]] .

(b) Selected anchors.

Apple:	Apple Computer, APPLE, Apple Records, Apple (album), Apple II family, Malus, Apple Store (retail), Apple (super mario), Yabluko, Apple I
Sherlock 3:	Sherlock 3
Mac OS X v10.2:	Mac OS X v10.2

(c) Possible targets of the selected anchors

[[Apple Computers|Apple]] bundled a similar program, [[Sherlock 3]] , with [[Mac OS X v10.2]] .

(d) Final linked text, with the *Apple* anchor directed to *Apple computers*.

Fig. 1: Processing of an input text from the Wikipedia article *Karelia Watson*.

Using word boundaries as implemented in the Java regex package for anchor detection does not work for two reasons:

- Due to the idiosyncrasies of the INEX Wikipedia collection, spurious spaces are inserted or removed around markup, even in the middle of words, so word boundaries cannot be trusted.
- Anchors may only partly cover a given word; this is bad style, but there are instances where *child* as part of *children* is linked to the corresponding article. For other languages like German, compound words can be formed without spaces, so this might happen more frequently.

Although the second case is rare – especially in the English version of Wikipedia used for INEX –, the first reason is sufficient to justify the decision not to analyze word boundaries.

The result of this stage is a collection of non-overlapping anchor texts that might be turned into links. Based on the training data set, we know for each anchor set the possible target pages as well as the absolute frequency of references to a certain target page under the given name. We now have to develop a ranking of the targets for every potential anchor.

2.2 Reducing the Size of the Anchor Index

Our approach requires statistics about the existing links in the training collection. We examine every link in the collection and store the anchor text along with the target page's ID. Then, we count the number of occurrences for each anchor text/target page pair to see how often a given anchor text is used to refer to the given page.

This information is sufficient input for our approach, but to both keep the index size small and remove spurious entries, we remove all anchor text/target page pairs with one of the following properties:

- The length of the anchor text is less than 5 or greater than 60. Very long anchors include anchors like *Best Writing, Story and Screenplay Based on Factual Material or Material Not Previously Published or Produced*; they mostly refer to very specific page titles that are unlikely to occur in normal text. Short anchors are removed because they are usually ambiguous and they can lead to false positives.
- The anchor text refers to ten or more different pages. This implies that the anchor text is very general like, for example, *her father*.
- The anchor text occurs less than five times in the collection.

The numbers used were chosen in a rather ad-hoc fashion; further research is required to determine whether these numbers are good (or even whether the filtering is needed at all). We will test this once the results and evaluation tools are available.

3 Link Target Disambiguation

In many cases, anchor texts refer to only one possible target, like *Sherlock 3* in the example in figure 1c. However, the anchor text *Apple* from the same example shows that there is not always a one-to-one mapping of anchor texts to target pages, so the link detector has to make a choice. Furthermore, it may be necessary to remove spurious anchors.

One obvious problem is that anchor texts are frequently only sensible in the context in which they occur; for example, the anchor text "her father" refers to different persons depending on who "her" refers to. Since low-level information about the document frequency of terms is not available in our setup, we could not use Itakura and Clarke's formula for selecting anchors to index, so we implemented the simple heuristics from section 2.2.

The remainder of this section is based on the following values that influence the choice of which targets to use for a given anchor:

1. The rank of this target for this anchor, based on the total number of references;
2. the rank of the target page when doing a full-text search for the new article's title; and
3. the rank of the target page when doing a full-text search for the new article's full text (optional).

We chose to use a linear combination of these factors to obtain the final rank of a target.

3.1 Analysis of Anchor/Link Frequency

In absence of any other information, the link finder can still look at the *prior probability* of a given anchor text referring to a given target. This information can be obtained by analyzing the frequencies of the different target pages for a certain anchor text. For example, in the INEX collection, the anchor *Apple* refers to *Apple Computer* 399 times, to *APPLE* 83 times and to *Apple Records* 65 times, so in absence of any further information, *Apple Computer* is most likely the correct target.

3.2 Analysis of the Target Text

Simply using the frequency of targets in the training collection, however, does not take into account the context provided by the new document: for example, the text of the document should already give a strong indication whether the article is about computers or music. Thus, a straightforward approach is to calculate the *textual similarity* of the new text and the possible targets; if the new text and a target have a high similarity, it is likely that they are about the same general topic (like computers or music).

In our implementation, we implement this by doing a single full-text search for the complete new text respectively its title on an index that comprises the

Table 1: Target distribution of the anchor *Apple* (case sensitive).

Rank	Count	Target page
1	399	Apple Computer
2	83	APPLE
3	65	Apple Records
4	7	Apple (album)
5	2	Apple II family
6	2	Malus
7	1	Apple Store (retail)
7	1	Apple (super mario)
7	1	Yabluko
7	1	Apple I

full texts of all articles in the test collection. This results in a single ranked list of articles that are somehow related to the new text; for every anchor text that is found in the new text, the highest-ranked article from this list is chosen.

3.3 Analysis of the Link Structure

According to our observation, it is likely that the documents that are linked from the same source document are connected. This is because these pages typically share a main topic, so if two topics are mentioned (or pages are referenced) on the same source page, these topics are more likely to be connected than two randomly chosen topics. We can exploit this to find the correct link target among a set of candidates; for every such set, we determine how many links to the target pages for the *other* anchor texts exist. The more links exist, the more likely the target is to be the correct link target for this anchor.

Figure 2 demonstrates that the pages linked from a single page tend to be heavily connected. We can see that *APPLE* is not connected to the pages that are actually referenced from the source page at all and that *Apple Records* only has one link, whereas *Apple Computer* has many links in this cluster of pages.

The link analysis will not work properly if there is a very low number of targets (or, more generally, if the potential targets are mostly unconnected). In this case, the link finder should select potential targets even if they are isolated. The exact mechanism and threshold for this are the subject of future research.

3.4 Combination of these Approaches

Of course, it is possible to not only use these approaches in combination, but also to combine the evidence to obtain better quality. Since each of the approaches can be used to find a ranked list of possible targets for a given anchor text, we chose to use a weighted combination of the different ranks as the basis for the final decision. Given the example rankings from table 2, and the weights $w_1 = 1$ (anchor/link frequency), $w_2 = 5$ (text similarity), and $w_3 = 2$ (link analysis)

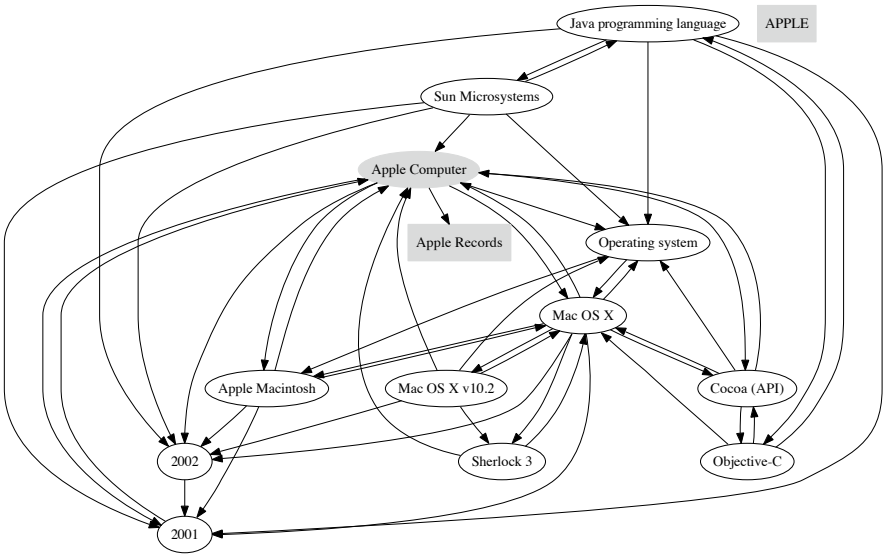


Fig. 2: The link network for pages linked from the page *Karelia Watson*. Our focus is on the shaded items, which are potential targets for the anchor text *apple*. The *Apple Records* and *APPLE* pages (in rectangles) are not linked from this page, but shown here for demonstration.

results in a final value of 12 for *Apple Computer*, of 13 for *Apple Records*, and of 24 for *Apple I*. Thus, in this case, the link target *Apple Computers* has the lowest combined rank and is selected as the final target. (Note that a higher weight value decreases the influence of the corresponding factor.)

Table 2: Example for combining the different aspects of target rankings.

Target	Anchor/link freq.	Text sim.	Link analysis
Apple Computer	1	2	1
Apple Records	2	1	3
Apple I	3	3	2

Since we did not finish the implementation of link-based target disambiguation in time, we only submitted runs using anchor/link frequency and text similarity. For text similarity, we search the full text of all articles for occurrences of the title of the new page to be linked. From a quality point of view, it would probably be better to search for the complete body text of the new article – otherwise we implicitly assume that the concept is already mentioned in the existing articles, although it does not have an article of its own. Unfortunately, the cost for doing this was prohibitive on our setup, so we had to settle for searching for the titles only. We used the different combinations of text similarity–anchor/link frequency weight, from equal weights for both (run LycosA2B-1-1), a weight of 5 for one and 1 for the other (runs LycosA2B-1-5 and LycosA2B-5-1). Furthermore, we submitted runs using only one of the two factors (runs LycosA2B-0-1 and LycosA2B-1-0).

At the time of writing, no results or evaluation tools are available, so we cannot say which of these approaches works best. We plan to also evaluate additional runs using the link-based target disambiguation once the assessments and evaluation tools become available.

3.5 Limitations

One base limitation of our work is that we assume that the collection already contains a large number of related articles. As Huang et al. [1] note, this assumption does not hold for a batch upload of related articles where links between the articles are at least as important as links to or from the collection. Another potential problem is that the anchor texts that have been used by the authors might not be meaningful (for example, "click here").

We believe, however, that the approach can work well in the right circumstances. We plan to use it on a community platform about German history, with the anchors from the German Wikipedia as a training set. The results from preliminary tests are quite promising.

4 Finding Incoming Links

Our current implementation for finding incoming links is simplistic: we simply search for the new document's title in the full-text index to determine a ranked list of candidate sources. (Note that no phrase search if performed, so in effect the results may contain pages where the terms from the title occur out of order.) Next, the title of the new document is searched for in each candidate's text, and the first occurrence is added to the list of links, ordered by the search rank. Finally, all pages where the title is *not* found – this may happen if the title comprises several words – are added to the end of the list.

5 Future Work

Our submission should be regarded as a first attempt at the problem; in particular, resolving ambiguities and avoiding the "discovery" of general terms like "father" as links is still unsolved. In future work, we plan to address this by taking more factors into account.

Acknowledgements

The research presented in this paper was partially funded by the German Federal Ministry of Economy and Technology (BMWi) under grant number 01MQ07008. The authors are solely responsible for the contents of this work. We thank our colleagues at Lycos Europe who gave valuable feedback.

References

1. Darren Wei Che Huang, Yue Xu, Andrew Trotman, and Shlomo Geva. Overview of INEX 2007 link the wiki track. In *Proc. INEX 2007*. Springer, 2008.
2. Kelly Y. Itakura and Charles L. A. Clarke. The University of Waterloo at INEX2007: Adhoc and Link-the-Wiki tracks. In *Proc. INEX 2007*. Springer, 2008.

Context Resolution Strategies for Automatic Wikipedia Linking

Michael Granitzer[1], Christin Seifert[2], and Mario Zechner[2]

[1] Knowledge Management Institute
Graz University of Technology
Inffeldgasse 21a, 8010 Graz
mgranitzer@tugraz.at,
http://kmi.tugraz.at/
[2] Know-Center Graz
Inffeldgasse 21a, 8010 Graz
{mzechner,cseifert}@know-center.at,
http://www.know-center.at/

Abstract. Automatically linking Wikipedia pages is done mostly by two strategies: (i) a content based strategy based on word similarities or (ii) a structural similarity exploiting link characteristics. In our approach we focus on a content based strategy by finding anchors using the title of candidate Wikipedia pages and resolving matching links by taking the context of the link anchor, i.e. its surrounding text, into account. Best-entry-points are estimated on a combination of title and content based similarity. Our goal was to evaluate syntactic title matching properties and the influence of the context around anchors for disambiguation and best-entry-point detection. Results show, that the whole Wikipedia page provides the best context for resolving links and that simple inverse document frequency based scoring of anchor texts is also capable of achieving high accuracy.

Key words: INEX, Link-the-Wiki, content based approach, similarity analysis

1 Introduction

This paper outlines the approach taken by the Know-Center Graz in the Link-the-Wiki Track of INEX 2008. While the task itself is outlined in detail in a previous chapter, we restrict ourselves to illustrate our approach and to discuss properties found to be relevant. Our approach has been evaluated on the 659,413 Wikipedia pages, wherefrom every link from or to a page in the test set as well as all pages of the test set itself have been excluded. In the following we refer to the corpus without the test set as the *Wikipedia corpus*. The two runs are distinguished as *file-to-file run*, having 6.600 test documents and *anchor-to-bep run* having 50 topics. A wiki page having all links removed from, is called an *orphan page*.

Basically we restricted ourselves to use standard retrieval techniques combined with efficient string matching technique as used in information extraction frameworks like GATE [2]. As shown in last years runs [6] and in related work from Wu and Weld [8] using such content based approaches yields reliable results. This fact and the applicability of standard IR and IE tools served as motivation for the presented work.

In our approach we identified the following steps for identifying links between Wikipedia pages:

1. *Source Document Identification*: In a first step the source of a link has to be determined. While the orphan wiki page determines the link source for out-links, in-link detection requires efficient search strategies for fast detection of source document candidates. Based on results from INEX 07, our approach utilizes searching for Wikipedia titles in order to determine source document candidates for in-links.
2. *Anchor Identification and Ranking*: The second step includes detection of the anchor position in the source document. We identify anchors by annotating source documents on the word level using a gazetteer list. The list is created from titles and/or anchor texts of links in the Wikipedia corpus. Each link is assigned a score based on the similarity of the surrounding text, called *anchor context* in the following, with the target page in the Wikipedia corpus. We use different document decomposition strategies for determining the *anchor context* based on the assumption that they influence the quality of the score.
3. *Target Document Identification*: The third step includes detection of the link target. Gazetteer matching already provides a list of target documents for each anchor. However, anchors may overlap. For example "United States of America" may have two overlapping links, one for "United States" and one for "United States of America". Those overlapping links have to be resolved here. Furthermore, in the file-to-file linking runs links have to be merged since they have to be provided on the document instead of the anchor level.
4. *Best-Entry-Point Identification*: As a last step the best-entry-points in the target document have to be identified. Again we focused on using the anchor context and calculate the similarity between the anchor context and document parts in the target document. The target document parts have been again identified using different decomposition strategies. In addition, if a target document part contains the title or parts of the title of the source page, we increased the similarity significantly. Thus, document parts containing the title are preferred as best-entry-point.

Our work aims to provide answers on issues like syntactic matching (i.e. on the word level like case sensitivity, part-of-speech tags, title matching vs. title and anchors etc.), influence of document decomposition strategies (i.e. sentences, automatically generated topics and document) on determining the anchor context as well as scoring strategies for removing high frequent, noisy links like for example "The" or "Are". Since we use standard information retrieval and information extraction technology our approach may be easily put into practice; a

valuable aspect for industries. We hope that our approach provides contrast to other approaches and shed light into automatic linking strategies for Wikipedia as well as its use in practical settings.

2 Document Indexing

The Wikipedia corpus itself is indexed using the open source search engine Lucene [5], with standard stop word removal. For each Wikipedia page the title and all anchors of links pointing to this page are extracted and stored for usage as gazetteer list. A finite state machine (FSM) is filled from this gazetteer. Transitions between states of the FSM are words occurring in a gazetteer entry, while states are distinguished into final states or intermediate states. Final states contain the URL of the Wikipedia page and if upon matching such a final state is reached, an annotation pointing to the particular Wikipedia page is added. In this way gazetteer matching allows us to annotate word sequences with hyperlinks for a large number of possible link targets at reasonable speed.

Orphan pages are preprocessed using the OpenNLP toolkit [1], whereby preprocessing includes tokenization, sentence detection and part-of-speech tagging. In a last step, the FSM is applied to annotate possible links. For resolving the context of an anchor we are relying on either the complete document, the sentence an anchor occurs in as well as a automatically detected, topical coherent block of sentences around the anchor. This automatic topic segmentation of a page is done using the well known C99 segmentation algorithm [3] and sentences are obtained by the sentence detection algorithm.

3 Linking Strategies

For a given orphan page d_o, our system determines a set of n possible in-links $I = \{< l_1, s_1 > \ldots < l_n, s_n >\}$ and a set of m possible out-links $O = \{< l_1, s_1 > \ldots < l_m, s_m >\}$. Each out-link/in-link is assigned a score s_i determining the confidence of the system in generating such a link. One link is - as defined in the LTW result set specification - a quadruple $l_h =< s_h, t_h, sp_h, b_h >$ where for link l_h s_h denotes the source page, t_h the target page, sp_h the span determining start and end of the link in the source document and b_h the best-entry-point in the target document.

In the following we present how the different properties have been determined and thereby differentiate between out-link and in-link generation. While both follow the same principle approach there are slight implementation differences for keeping link generation computational feasible.

3.1 Out-link Generation

Out-link generation starts with preprocessing the orphan document d_o as outlined in section 2. Gazetteer matching returns the set of possible out-links O,

whereby for each link l_i we know its source s_i, its target t_i and its span sp_i. For each link we determine the anchor context, defined as the text surrounding the link source. In our experiments we distinguish between sentence, automatically detected topics and the complete document as anchor context. All nouns of the anchor context are extracted and fed into the retrieval backend as Boolean OR query. To speed up this potentially large OR query we restrict the result set to pages pointed to by all links in the anchor context simply by adding all link target identifiers (i.e. the filename of the page) as AND query part. Thus, for all links having a span sp in the current anchor context we are receiving a score s. In particular the query is formulated as

$$(ID = t_1 \ \ OR \ldots OR \ \ ID = t_n) \ AND \ (w_1 \ \ OR \ \ w_2 \ \ \ldots \ \ OR \ \ w_k)$$

with $\{w_1 \ldots w_k\}$ as the nouns of the anchor context and t_k as unique identifier for the k^{th} link target and "$ID =$ " specifying the search on the metadata field containing the unique identifiers of a Wikipedia page. Formally, the score (named *anchor context score* in the following) returned is obtained from standard Lucene ranking as

$$s_i = coord_{w,i} * norm(w) * \sum_{t \in w} \frac{\sqrt{tf_{t,i}} * idf_t^2}{norm(i)} \tag{1}$$

where

- $tf_{t,i}$ is the frequency of term t in document i
- $idf_t = 1 + \log \frac{\#D}{\#D_t + 1}$ is the inverse document frequency with $\#D$ as the number of documents in the corpus and $\#D_t$ the number of documents containing term t
- $norm(w)$ is the norm of the query calculated as $\sqrt{\sum_k idf_k^2}$
- $norm(i)$ is the length norm of document i, namely the number of terms contained in document i
- and $coord_{w,i}$ is a overlapping factor increasing the score the higher the number of overlapping terms between query and documents are.

The Lucene scoring equation has been proven as reliable heuristic for full text searching. It can be seen as an heuristic version of a cosine similarity between anchor context and target document with emphasize towards the number of overlapping words. This assumption is quite naturally for resolving the context of a key. For example "tree" in computer science will occur more frequently with terms describing data structures than the "tree" in nature. Thus, depending on the position of a link in the document we receive different scores. For linking on the word-level resp. for the best-entry-point task, links are ranked according to their score and the best $n = 50$ links are taken as candidates.

An alternative determination for the score is based on the observation that some pages have very common titles. Such pages are for example "The", "Are". For removing such high frequent anchors we simply took the inverse document frequency of the anchor text as scoring scheme. The rational is that noisy, high

frequent links occur in nearly every document and therefore provide no additional information independent whether they are a true links or not. In particular the score, named *anchor IDF* in the following, is calculated as

$$s_i = \log \frac{\#D}{\#D_a} + 1$$

where $\#D$ is the number of wiki pages in the corpus and $\#D_a$ the number of wiki pages containing the anchor text of the link.

For the file-to-file task links pointing to the same target t but having different spans sp are merged. We distinguish three different merging strategies, namely the highest score of the link, the average score of the link or simply by counting the number of links to a target t.

Our approach raises several questions to analyze. For gazetteer matching we focused on a very high recall by allowing fuzzy matching strategies, assuming that we can disambiguate them efficiently in the following step. In particular the question is how much noise added by the gazetteer can be resolved afterwards. Therefore we considered the following properties upon gazetteer matching:

- Case sensitive vs. case insensitive matching may impact different link categories. Named entities like persons, technical abbreviations (e.g. *AJAX, R*) may be perfectly identified by case sensitive matching, while case insensitive matching prefers more general concepts like *web applications* etc. Considering this parameter may outline differences between the different link categories.
- Filtering gazetteer entries based on identified part-of-speech: Some titles contain numbers or particular syntactic elements like brackets. Filtering gazetteer entries based on their part-of-speech should allow making matching considering those cases by removing non nouns.
- Filling the gazetteer list using page titles only or titles and anchors of a link raises questions towards how much information is added by anchors of links in the Wikipedia corpus and how noisy it is. From a statistical point of view we obtain around 1.7 million gazetteer entries by taking titles and anchors into account, achieving a very high recall. Our question herein is whether the anchor text is capable to disambiguate those additional entries or not.
- Selecting longest common sequence matching links from overlapping anchors: Through gazetteer matching the span sp of links may overlap. Our hypothesis here is that the longer the sequence of words, the more specific a link is. Therefore, those more specific links should be chosen.

The core question in the following disambiguation step is how to determine the anchor context. We investigate three different levels, namely sentence, topic (as a sequence of sentences) or the whole document. We did not assume that the orphan document is structured in any way and thus topics are detected automatically using the above mentioned C99 algorithm. However, analyzing the impact of structures is up to following research. Regarding the scoring of

an link our main interest lies in the difference between the anchor IDF scoring scheme and the anchor context scoring scheme. In particular the question is how much useful context information surrounds a link and whether it is worth to consider this context information.

3.2 In-link Generation

In-link generation is in principle similar to out-link generation with the difference that in a first step we have to determine the source document d_j of a particular link. Again we utilize title matching for doing so, but in contrast to out-link generation the title is used as search string instead of gazetteer matching. Similarly to out-link generation we are determining different contexts, in this case best-entry-point contexts, to assign a score to a link. Again sentences, topics or the whole document serves as context. Given the nouns of this context as sequence $< w_1, \ldots, w_k >$ we are sending the following query to the backend:

$$\text{“}title\text{”}\ AND\ (w_1\ OR\ w_2 \ldots OR\ w_k)$$

where “*title*” indicates a phrase query for the title of the Wikipedia page. Again the score is calculated as outlined in equation 1.
From the result set we obtain a ranked list of possible link source candidates. If the context is different than the whole document, merging strategies are required to merge the ranked lists of the different contexts. Similarly to out-link generation we utilized the highest scoring source candidate, the average score of a source candidate or a simple counting scheme. Taking the n best source candidates is either the input for determining the best-entry-points or gives us already the result for the file-to-file linking task.

Using more fine grained contexts than the document level follows the intuition that a link points to a Wikipedia page because the author wants to address a particular aspect of the relationship and not all aspects of the target page. By considering those fine grained similarities we tried to address this aspect.

3.3 Best-Entry-Point Detection

Either in-link or out-link generation provides a list of best matching links including target page, source page and the span of a link. In the final step, best-entry-points are determined again based on document decomposition. Our hypothesis is that the best-entry-point in the link target has to be similar to the anchor context. Furthermore, if the title of the source page is contained in the link target, those parts of the target document are preferred entry points. Since we obtain a score for each entry point, results are ranked and the best five entry points are taken as result.

In particular, similarity is calculated using a simple vector space model with local TFIDF weighting. Given the link target t, the textual content of the target

is preprocessed and decomposed into segments $t_{r,1} \dots t_{r,k}$. Segments are either sentence or topics. After filtering out all non-noun words, each segment is converted into a term vector. The weight of a term is calculated according to the TFIDF scheme, but based on the extracted segments, as:

$$w_{r,l} = tf_{r,l} * \log(1 + \frac{\#R}{\#R_l}) \tag{2}$$

where $w_{r,l}$ is the weight of term l in segment r, $tf_{r,l}$ is the number of times a term l occurs in segment r divided by all terms in segment s, $\#R$ is the number of segments in the target document and $\#R_l$ is the number of segments containing term l.

Similarly to the target segments, the anchor context in the source document - denoted as a - is also converted into a term vector by filtering all non-nouns and applying equation 2.

The ranking of best-entry-points is obtained by calculating the cosine similarity between anchor context $\overrightarrow{a}$ and all target segments $\overrightarrow{t}_{r,1} \dots \overrightarrow{t}_{r,k}$ and rank them accordingly. Segments containing the title of the anchor page are favored by increasing the similarity as follows:

$$s(\overrightarrow{a}, \overrightarrow{t}_{r,i}) = \begin{cases} title \in t_{r,i}: & (1 + \frac{\overrightarrow{a} \cdot \overrightarrow{t}_{r,i}}{\|\overrightarrow{a}\| * \|\overrightarrow{t}_{r,i}\|})/2 \\ title \notin t_{r,i}: & \frac{\overrightarrow{a} \cdot \overrightarrow{t}_{r,i}}{\|\overrightarrow{a}\| * \|\overrightarrow{t}_{r,i}\|} \end{cases}$$

Best entry points are returned as starting point of the text segment since we assume that a reader does not want to start reading in the middle of a sentence or paragraph.

4 Implementation and Evaluation Details

As outlined above, Lucene [5] has been used as search backend and OpenNLP [1] for preprocessing. All algorithms are developed in Java, including the gazetteer component. Since our approach, at least for out-link detection, heavily relies on gazetteer matching the question is whether a gazetteer with low runtime and low memory resource consumption is feasible. In our FSM approach the gazetteer with titles and anchors consisted of around 1.7 million entries and used up around 800 MB main memory. Additionally, gazetteer entries may be distributed using distributed computing techniques like Map & Reduce [4] and thus scaling up is possible in our approach.

Runtime behavior also satisfies interactive requirements. On a dual core laptop with 4GB of main memory file-to-file runs took around 64 minutes using the more complex anchor context scoring - that is around 1.7 documents per second. After finding the link candidates, best-entry-point matching does not increase runtime complexity. Thus, the overall process can be seen as computational tractable and scalable.

During the development we did internal benchmarking for file-to-file and anchor detection using the TREC evaluation program trec_val. Those results differ strongly from the released preliminary results [3] so we assume that there has been an error in our submission format. However, both numbers are outlined in this section. Furthermore, results will be corrected in the post-proceedings after the release of the evaluation tool.

The runs can be differentiated in file-to-file in-link/out-link generation, out-link anchor detection and best-entry-point detection. File-to-file runs are evaluated on the 6.600 topics defined by the organizers. Out-link anchor detection and best-entry-point detection are run on the 50 topics defined by the participants. After the development of our algorithms we analyzed the different algorithmic properties by taking the available ground truth of the file-to-file and anchor detection runs. This allowed evaluation of all runs but the best-entry-point runs.

For out-link generation we distinguished between the title only vs. title and anchor matching, case sensitive vs. case insensitive matching (CS), longest common sequence (LCS) matching, different document segmentation level (DSL) as well as the two different ranking schemes, namely the anchor IDF and the anchor context score. Results of the internal runs including the official map of selected runs for file-to-file out-link generation are depicted in table 1, for anchor generation in the best-entry-point run in table 2. Figure 1 compares our out-link runs to the best runs of the other LTW track participants.

Table 1. Results for Out-link Generation File-to-File

Title Only	LCS	CS	Scoring	Segmentation	MAP_{intern}	$MAP_{official}$
true	false	false	anchor context	document	0.548	0.1129
true	true	false	anchor IDF	NA	0.5038	0.1407
true	false	true	anchor context	document	0.471	NA
true	true	true	anchor IDF	NA	0.4508	NA
false	true	true	anchor IDF	NA	0.4392	NA
true	false	true	anchor context	topic	0.4258	NA
false	true	false	anchor IDF	NA	0.4215	NA
false	false	true	anchor context	document	0.3827	NA
false	false	true	anchor context	topic	0.3809	NA
false	false	true	anchor IDF	NA	0.3478	NA
true	false	true	anchor context	sentence	0.3369	NA

Overall our findings for out-link generation can be summarized as follows:

– *Gazetteer Matching*: Using only nouns in the matching process did not have strong impact in our experiments and therefore detailed results have

[3] Preliminary results on file-to-file and anchor linking released on 28^{th} of November

Table 2. Results for anchor out-link generation

Title Only	LCS	CS	Scoring	Segmentation	$MAP_{official}$
true	false	false	anchor context	document	0.2370
true	false	false	anchor context	topic	0.2039
true	true	false	anchor IDF	NA	0.2044

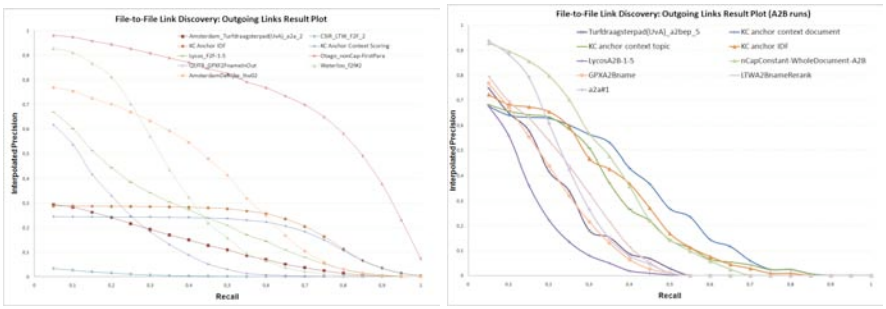


Fig. 1. Precision-Recall curve of our outlink file-to-file runs (left) and anchor-to-bep runs (right) compared to the best runs of other participants.

been omitted here. Taking only the longest common sequence of overlapping matches has an impact on the anchor TFIDF scheme but not on anchor context scoring. Our assumption is, that anchor context scoring is able to disambiguate overlapping matches while this is not the case for anchor TFIDF. Overall, case insensitive matching yields to better results than case insensitive matching.

- *Document Segmentation and Scoring Schemes*: Surprisingly using the whole wiki page as context worked out to be best followed by using automatically detected topics. Considering sentences as anchor context did not perform well. However, all except of the document based anchor context scoring stayed behind simple anchor IDF matching. Also, noise added by considering title & anchors during the matching step could not be resolved by either scoring scheme yielding to the conlcusion to exclude anchor texts in the matching step.
- *Merging Strategies*: Maximum and average merging seem to work well compared to just counting the score. However, more runs on different parameter variations have to be done in order to get a more concrete picture on the influence of merging strategies.

For in-link generation different document decomposition strategies have been analyzed. In addition we compared whether using the title as OR instead of a phrase query increases or decreases accuracy. Anchors are again identified by matching the title of the orphan page. Title matching has been done case insensitive. Results for file-to-file runs are outlined in table 3 and for anchor

generation of in-links in table 4. Figure 2 compares our in-link runs to the best runs of the other track participants.

Table 3. Results for in-link generation file-to-file

Title as OR Query	Segmentation	MAP_{intern}	$MAP_{official}$
false	document	0.6355	0.5300
false	sentence	0.5938	0.5369
false	no context	0.5938	NA
true	document	0.5066	NA
true	sentence	0.4088	NA
true	no context	0.4088	NA

Table 4. Results for in-link anchor generation

Document Segmentation	$MAP_{official}$
Document	0.1685
Sentence	0.1663
Topic	0,1391
No context	0,1386

Results show that again using the whole document as context yields the best results, while sentences or topics as context perform similar to only searching for titles. Making an OR query out of the title did worsen results significantly.

After the release of the evaluation tool we plan to evaluate the differences between our internal and the official runs and evaluate best-entry-point results. Furthermore, some parameter combinations have not been evaluated till now and by using significance testing we plan to test the influence of different parameter combinations as well as take a more detailed look into the influence of particular parameters onto specific topics.

5 Conclusion

Based on the usage of standard IR and IE technology our results seem to be promising. However, numbers presented here have to be taken with care since internal benchmarks differ from official results and we assume that there are errors in our submission format. Besides this, the preliminary results point out that simple scoring strategies like our anchor IDF yield to reliable results while not being able to disambiguate the context of links as good as taking the whole document as anchor context into account. Regarding the context of a link our experiments showed that taking the whole page as context turns out to be best

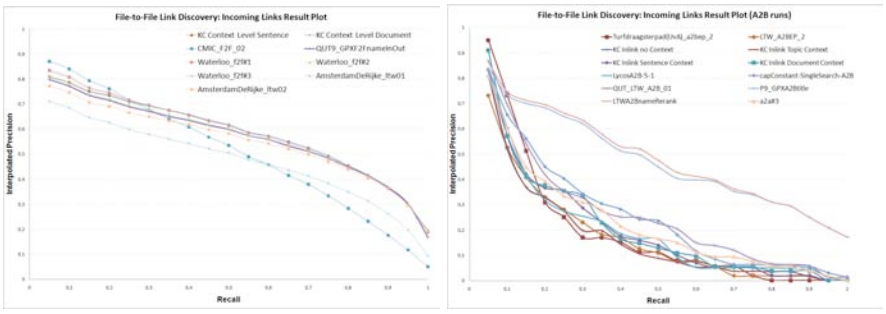


Fig. 2. Precision-Recall curve of our in-link file-to-file runs (left) and anchor-to-bep runs (right) compared to the best runs of other participants.

suited for outlink as well as for in-link detection. Although results look promising title matching and the subsequent scoring scheme seem to be too tightly coupled. For example without filtering out overlapping title matches by taking only the longest common sequence anchor TFIDF performs worse.

Such interwoven parameters are an indication for the need to decouple the different linking steps and the use of machine learning approaches in each step. Using machine learning seems to be very promising as shown in very recent work [7]. Also, we completely ignored the pre-given structure of Wikipedia pages like section and paragraphs. Maybe those structures allow a better definition of anchor contexts and best-entry-point contexts.

We think that two important aspects should be covered by the Link-the-Wiki track next year. One is the automatic labeling of the link types between Wikipedia pages. For example the page Berlin linking to Germany marks a part-of relationship while a link between Berlin and Capital marks a is-a relationship. Developing methods for automatically identifying such relationships for Wikipedia links may have a huge practical but also theoretical impact in boosting new technology like semantic wikis. The second possible extension regards linking to documents outside the Wikipedia, i.e. determining external links. Again we think there is a practical impact and yielding to new search paradigms with Wikipedia in its core.

Acknowledgement

The Know-Center is funded within the Austrian COMET Program - Competence Centers for Excellent Technologies - under the auspices of the Austrian Ministry of Transport, Innovation and Technology, the Austrian Ministry of Economics and Labor and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG. We also want to thank the organizer of the Link-the-Wiki track for their effort in making the track possible.

References

1. T. & Bierner G. Baldridge, J.; Morton. Opennlp: The maximum entropy framework. Web Site http://maxent.sourceforge.net/about.html, 2001. , last visited June 2008.
2. Kalina Bontcheva, Hamish Cunningham, Diana Maynard, Valentin Tablan, and Horacio Saggion. Developing reusable and robust language processing components for information systems using gate. In *DEXA '02: Proceedings of the 13th International Workshop on Database and Expert Systems Applications*, pages 223–227, Washington, DC, USA, 2002. IEEE Computer Society.
3. Freddy Y. Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 26–33, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
4. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.
5. Erik Hatcher and Otis Gospodnetic. *Lucene in Action (In Action series)*. Manning Publications, December 2004.
6. Darren Wei Che Huang, Yue Xu, Andrew Trotman, and Shlomo Geva. Overview of inex 2007 link the wiki track. *Focused Access to XML Documents*, LNCS 4862:373–387, 2007.
7. David Milne and Ian H. Witten. Learning to link with wikipedia. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge mining*, pages 509–518, New York, NY, USA, 2008. ACM.
8. Fei Wu and Daniel 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.

Wikisearching and Wikilinking

Dylan Jenkinson, Kai-Cheung Leung, Andrew Trotman

Department of Computer Science
University of Otago
Dunedin
New Zealand

{dylan, kcleung, andrew}@cs.otago.ac.nz

Abstract. The University of Otago submitted three element runs and three passage runs to the Relevance-in-Context task of the ad hoc track. The best Otago run was a whole-document run placing 7th. The best Otago passage run placed 13th while the best Otago element run placed 31st. There were a total of 40 runs submitted to the task. The ad hoc result reinforced our prior belief that passages are better answers than elements and that the most important aspect of the focused retrieval is the identification of relevant documents. Six runs were submitted to the Link-the-Wiki track. At time of writing the results had not been published.

1. Introduction

Otago participated in the Relevance-in-Context task of the ad hoc track submitting six runs, three passage and three element runs. The passage runs compared the Otago 2007 algorithm to a previous algorithm examined by Otago, the Kullback-Leibler model, and to whole document retrieval. The result suggest that whole document is better than passage retrieval and that there is little difference between the other two algorithms.

Otago also participated in the Link-the-Wiki track, preferring a variant of the Itakura & Clarke algorithm for outgoing links, and searching for the orphan title for documents that should link to the orphan. At the time of writing the results for the track had not been published.

2. Wikisearching

2.1. Passages

2.1.1. The Otago 2007 Algorithm

The approach taken by Otago at INEX 2007 [1] was two step. First, relevant documents were identified using BM25. Second, all the occurrences of all the search

terms with a document were identified (stemming with Porter's algorithm) and a fixed sized window of 300 words placed on the centroid. The centroid was defined as the mean of the term locations within the document, or alternatively the mean of those within one standard deviation of the true mean.

2.1.2. The Kullback-Leibler Algorithm

In earlier experiments at Otago, Huang et al. [2] examined techniques for identifying relevant passages within a relevant document and converting those into elements by taking the smallest element that fully enclosed the passage. Of the passage selection methods examined, the Kullback-Leibler model was the most effective:

$$\mathrm{KL(W|Q)} = \sum_{\mathrm{t}\in\mathrm{Q}} \mathrm{P(t|W)log}\left(\frac{\mathrm{p(t|W)}}{\mathrm{p(t|D)}}\right)$$

where W is a window within a document, D, and t is a search term of query, Q, and

$$p(\mathrm{t}|W) = \frac{tf_W + 0.5}{|\mathrm{W}| + 1}$$

and

$$p(\mathrm{t}|D) = \frac{tf_D + 0.5}{|\mathrm{D}| + 1}$$

where tf_D is the number of occurrences of t in D and $|D|$ is the length of document D (and likewise for tf_W with respect to the window, W).

Several strategies for choosing the window were examined. The sliding non-overlapping window of size 400 words was shown to be effective on the INEX IEEE document collection (measured with MAep and iMAep).

Itakura and Clarke [3] suggest that methods of identifying elements from passages are not as effective as methods of identifying elements directly. This is, in part, because the conversion from a passage to an element usually involves increasing the size of the passage and this extra text is expected to be non-relevant (by the passage retrieval algorithm). That is, the conversion from a passage to an element is unlikely to affect recall but is likely to decrease precision. If this is the case then the prior reported result of Huang et al. is understated and a comparison of Kullback-Leibler to Otago 2007 is necessary to progress our work.

2.2. Elements

2.2.1. The Beigbeder Algorithm

Beigbeder [4] proposes a method of scoring elements based on fuzzy proximity. If a document contains one occurrence of one search term, then the fuzzy proximity (fp) to term occurrence t, for location p is

$$fp = max\left(\frac{k-(p-t)}{k}, 0\right)$$

If the document contains more than one term occurrence of the same term then the fuzzy proximity is defined as the fuzzy proximity to the closest term occurrence (that is, *max*(*fp*) with respect to that term). If the document contains multiple search terms then the fuzzy proximity is defined as the minimum fuzzy proximity to all search terms.

The fuzzy score of an element in a document is computed as the sum of fuzzy proximity scores for each term in the element, normalized by the length of the element. However, as the documents are hierarchically structured, if a search term occurs in the title of a section then the fuzzy proximity of a term in the element to the search term in the title is defined as 1.

2.2.2. Small Improvements

Beigbeder's algorithm treats all terms as equal whereas it is usual for scoring algorithms to weight terms differently. The algorithm is thus extended to include some aspect of the strength of a search term (IDF). The IDF weighted fuzzy proximity, *fp'* is given by

$$fp' = IDF * max\left(\frac{k-(p-t)}{k}, 0\right)$$

the variant of IDF chosen is

$$IDF = \frac{N-n+0.5}{n+0.5}$$

where *N* is the number of documents in the collection and *n* is the number of documents in which the term occurs.

Problematically, if a search term is missing from the document then the fuzzy proximity to that term is always zero and so no part of the document is considered relevant (due to the *min*() function). Using the sum of fuzzy proximity weights in place of the minimum overcomes this problem.

The Beigbeder algorithms is of general interest as it is a method of identifying relevant elements as a function of term proximity, and can be extended to identify relevant passages. A comparison of the original Beigbeder algorithm and the Otago variant; as well as to the Otago passage runs will help answer the question of whether passages or elements are the best result to the Relevance-in-Context task.

2.3. Documents

At INEX 2007 an RMIT University ad hoc submission demonstrated that a full-document run could be more effective at focused retrieval than a focused run [5].

Geva and Winters[1] suggest this is because the F measure of recall and precision pre-selects choosing whole documents as 100% recall within a document can be easily realized. Whole document runs were, therefore, submitted for comparison to the focused retrieval runs.

2.4. Otago ad hoc 2008 Runs and Results

Three runs were submitted to the Relevance-in-Context passage task. In all cases documents were identified using BM25 (k_1=1.2, k_3=7.0, b=0.75) and then one passage was identified for each document in the top 1500 documents. The rank order of the final results was BM25. Stemming was not used.

WHOLEDOC_PASSAGE: The whole document was returned as the passage.

DYLAN_200: A fixed sized window of 200 words was placed on the centroid of the search terms within the document. The standard deviation method was used to compute the centroid.

SW_KL_200: The Kullback-Leibler method with a sliding window of 200 words was used to identify a relevant passage.

Three runs were submitted to the Relevance-in-Context element task, BM25 was used to identify the top 1500 documents, one element was identified, and the results re-ranked based on the Beigbeder score. For these experiments k=200.

WHOLEDOC: The whole document was returned as an element (this run is identical to WHOLEDOC_PASSAGE and was submitted as a sanity check).

BEIGBEDER_ORIG: Elements were scored using Beigbeder's algorithm.

BEIGBEDER_IDF: Elements were scored using the IDF weighed version of Beigbeder's algorithm. Due to a bug in our code we actually implemented the product of the sum of the IDF and *fp* scores in place of the sum of the product.

2.5. Wikisearching Results

The results are presented in Table 1 where it can be seen that WHOLEDOC and WHOLEDOC_PASSAGE do, indeed, score the same thus passing the sanity check. The passage algorithms are superior to the element algorithms with the Kullback-Leibler approach bettering the Otago 2007 approach by a very small amount. The IDF enhancement to Beigbeder's algorithm increases the precision substantially, but not sufficiently to better the passage runs.

[1] Private communications

Table 1. Ad hoc Relevance-in-Contest task results

Run	Type	MAgP
WHOLEDOC_PASSAGE	Passage	0.192
WHOLEDOC	Element	0.192
SW_KL_200	Passage	0.183
DYLAN_200	Passage	0.182
BEIGBEDER_IDF	Element	0.149
BEIGBEDER_ORIG	Element	0.107

3.0 Wikilinking

The Link-the-Wiki task, first included in INEX in 2007, requires participants to automatically identify hypertext links between documents in the Wikipedia. The user model is that of a user who creates a new Wikipedia entry and would like to link that entry to pre-existing entries in the Wikipedia (and *vice versa*).

The production of a new article can be simulated by taking an existing Wikipedia document and removing all trace of it from the collection. Link identification software can then be applied to the collection and the orphaned document. A comparison of the automatically generated links to the original collection gives some measure of the quality of the link detection system – that is, the original links are considered to be the gold-standard by which systems are compared.

Exactly this approach was taken in the INEX 2007 Link-the-Wiki track, and was used again for document-to-document linking in 2008. In 2008, 6600 documents (about 1% of the document collection) were randomly selected and orphaned for document-to-document link detection.

New in 2008 is the anchor-to-BEP linking task, in which the task is to identify the best orphan anchor from which to link from and the best-entry-point (BEP) in the target document from which to link to. Unlike document-to-document linking, anchor-to-BEP linking requires manual assessment because the Wikipedia documents are typically not *a priori* marked-up in this way. For 2008, 50 anchor-to-BEP documents were suggested by task participants and were orphaned for the experiment. A limit of 50 anchors per document was imposed (for practical reasons) and at most each anchor could link to 5 locations in the Wikipedia.

Two separate problems exist with identifying links, the identification of outgoing links (from the orphan to the collection) and the identification of incoming links (from the collection to the document).

3.1. Outgoing Links

Although the Otago runs in 2007 were adequate, those of Itakura & Clarke [6] were substantially better – effort was, therefore, spent investigating methods of improving their technique. It should be noted that the Itakura & Clarke algorithm relies on a pre-existing heavily interlinked document collection (such as the Wikipedia). In the case

where no prior links exist in the collection the techniques of Geva [7] which were also successful in INEX 2007 can be used.

3.1.1. The Itakura & Clarke Algorithm

The Itakura & Clarke algorithm relies entirely on pre-existing links between documents within the document collection. Of the link types available in the collection, only the <collectionlink> type is utilized because the other link types do not link between two documents in the collection (for example, a <wikipedialink> links from a document in the collection to a document in the Wikipedia that is not in the INEX collection).

Initially a list of all the links within the document collection is created. This is generated by parsing each document in the collection and extracting the anchor text of the link and the target document id.

Next and from the output of the previous stage, a list of target documents is created for each unique anchor text in the collection. For a given anchor text in the collection, the most frequent target is most likely to be the correct target.

For each anchor text / target pair a strength value (γ) is constructed

$$\gamma = \frac{np}{af}$$

where *np* is the number of documents that link from the anchor to the target and *af* is the number of documents in which the anchor text occurs.

An orphaned document is then parsed and the first location of each anchor in the pre-generated list is located. For overlapping anchors (for example, “Lennon” and “John Lennon”) the longest possible anchor is chosen as a longer anchor is more likely to be correct than a short anchor. A limit of 250 anchors per document was enforced by the Link-the-Wiki track definition.

3.1.2. Small Improvements

After implementing the Itakura & Clarke algorithm verbatim a small number of improvements were identified.

The algorithm defines the anchor text as all text occurring between the tags, converted to lowercase, and including punctuation. Anchor texts often contain punctuation at the end thus creating a distinction between “John Lennon” and “John Lennon.”. We stripped punctuation from the anchors thus conflating these two cases.

Anchor texts beginning at the start of a sentence are capitalized for grammatical reasons so the algorithm converts the text into lower case. Unfortunately this results in a distinction between “unfinished music” and “Unfinished Music” (the two part experimental work by John Lennon and Yoko Ono). Geva [7] identifies the importance of case in link detection so the case conversion step was dropped.

Finally, over-weighting γ for capitalized terms in the orphan will help identify proper noun conflicts (such as Unfinished Music). A capitalization constant, Π, is added to γ where terms in the orphan were found capitalized.

Figure 1 compares the improvements to the original algorithm using the INEX 2007 Link-the-Wiki topics. The line labeled “Waterloo” is the Itakura & Clarke run as

submitted. Removing punctuation (Alphanumeric) from the anchor list improves the algorithm, removing case folding (Case Sensitive) leads to further improvements. Weighting (Weighed) includes punctuation removal, case sensitivity, and weighted γ, and was the best experimental run on the 2007 orphans.

Figure 2 shows the effect of Π on precision, a value of 0.3 is best for early precision, but a value of 0.1 holds the precision longer resulting in the highest mean average precision.

3.1.3. Best Entry Points

Several studies have shown the best entry point for Wikipedia documents is the start of the document. [1, 8]. No further investigation was performed on BEPs.

3.1.4. Multiple Targets

The Link-the-Wiki task specification for 2008 allowed at most 5 targets for each anchor point. The Itakura & Clarke algorithm was, consequently, extended so that the γ value was computed for not just the most common target, but also for all targets of an anchor text. The γ values represent the probability of the target document being the correct target; consequently choosing the top five documents (by γ) for each anchor text satisfies the track requirements.

3.2. Incoming Links

The best Otago run at INEX 2007 achieved an excellent early precision (P@5) score of 0.751. The experiments described in this section were conducted in an effort to improve the overall performance (MAP) and were conducted on the 2007 Link-the-Wiki oprhans.

3.2.1. The Otago 2007 Algorithm

The algorithm for detecting incoming links relies on a simple theme extraction technique used to identify the semantic content of the document.

For each unique term (excluding stop words) in the orphaned document the Otago 2007 algorithm [1] computes the actual frequency of that term, *af*

$$af = \frac{tf}{dl}$$

where *tf* is the number of occurrences of the term in the orphan and *dl* is the length of the orphan (in terms); to the expected frequency, *ef*

$$ef = \frac{cf}{df * ml}$$

where *cf* is the number of occurrences of the term in the collection, *df* is the number of documents containing the term and *ml* is the mean length of a document. Ranking the terms in the orphan by ratio of *af* to *ef (st)*,

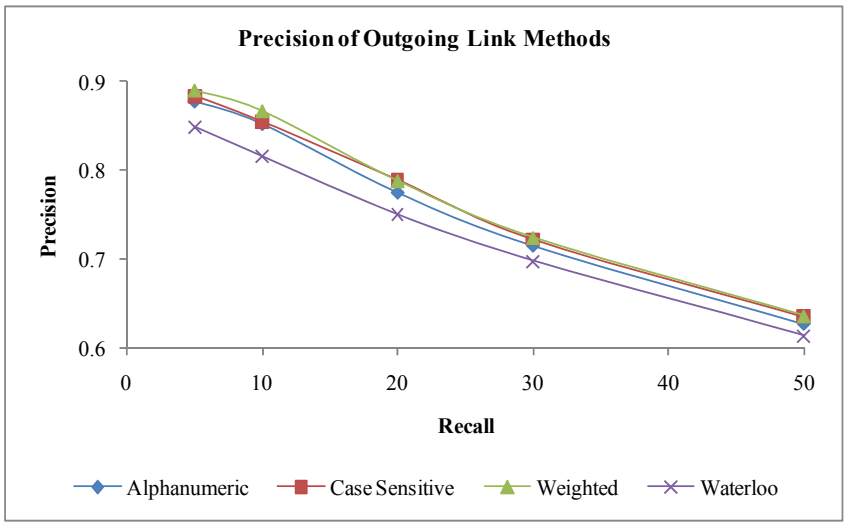


Fig. 1. Small improvements on the Itakura & Clarke algoritm (Waterloo) are seen when punctuation is removed (Alphanumeric), when case folding is removed (Case Sensitive) and when uppercase anchors are preferred over lowercase anchors (Weighted).

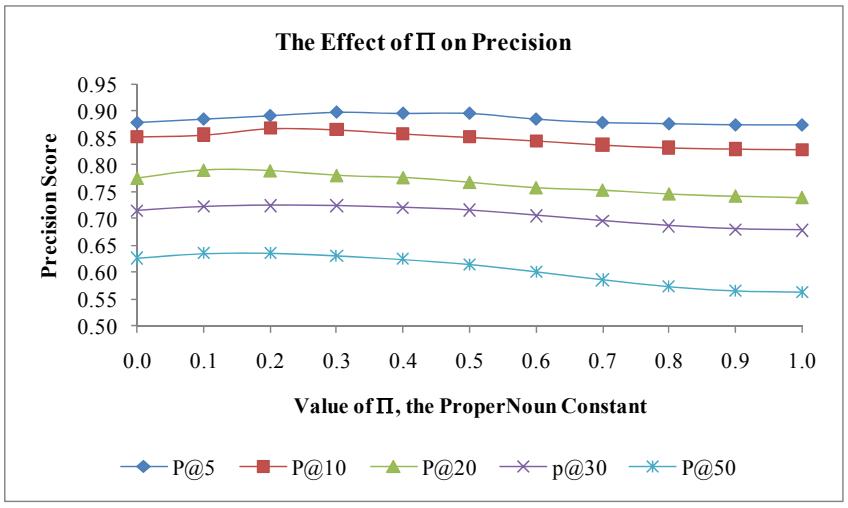


Fig. 2. Effect of varying Π on the precision. Small value of Π (0.3) is best for early precision but a very small score (0.1) holds the precision higher longer (best for MAP).

$$st = \frac{af}{ef}$$

provides a list of terms in order of occurrence relative to expected occurrence. If this ratio is larger than 1 the term occurs in the document more often than expected, if it is less than 1 it occurs less frequently than expected. The top ranked terms are representative themes of the document and are used to construct queries. The results of these queries are documents relevant to the themes of the orphan and therefore these documents should link to the orphan.

3.2.2. Improvements – Multiple Searches

For INEX 2007, queries were constructed by taking the top *n* terms from the *st*-ordered term-list and performing a query, extracting the top *n* * 50 results and then concatenating them to the list of results until a total of 250 results were found. That is, for *n*=2, three searches were performed, the first identifying the top 100 results and the second identifying the next 100 results, and the last identifying the remaining 50 results. There was no theoretic justification for this approach; it was motivated by time constraints. It is of note, however, that it was not an unsuccessful approach.

By merging the results of each separate query on the rsv (in this case the BM25 score) good targets that match other than the top theme will be placed high in the results list. This approach might also place documents that are good matches for non-key themes high in the results list because of a high rsv with respect to a non-key term.

To alleviate this problem the BM25 score for each search term can be weighted. The strength of a term with respect to the orphan has already been computed (*st*) and so that is a reasonable value to choose.

The best Otago run at INEX 2007 used two searches of 4 terms each, producing a total of 250 results in the results list. Using merging and weighted merging on the 2007 orphans the best number was 2.

The results are shown in Table 2. The best runs submitted to INEX 2007 (by any participant) achieved a score of 0.484 and is listed for comparative purposes. The best Otago run at INEX 2007 achieved a score of 0.339 which is better than the score achieved by result merging (0.319) but not as good as the 0.350 achieved by weighted result merging.

Figure 3 shows the early precision scores for the same three techniques. Of particular interest is that although the MAP score for weighted merging is highest, the early precision scores of the Otago 2007 run are highest.

Table 2. MAP scores for different approaches to multiple searches. The weighted merging of queries containing 2 terms each achieved a better score than the best Otago 2007 run, however not as good as the best run submitted by any institute.

Run	MAP
Top INEX 2007 run	0.484
Weighted merge	0.350
Otago 2007	0.339
Merged	0.319

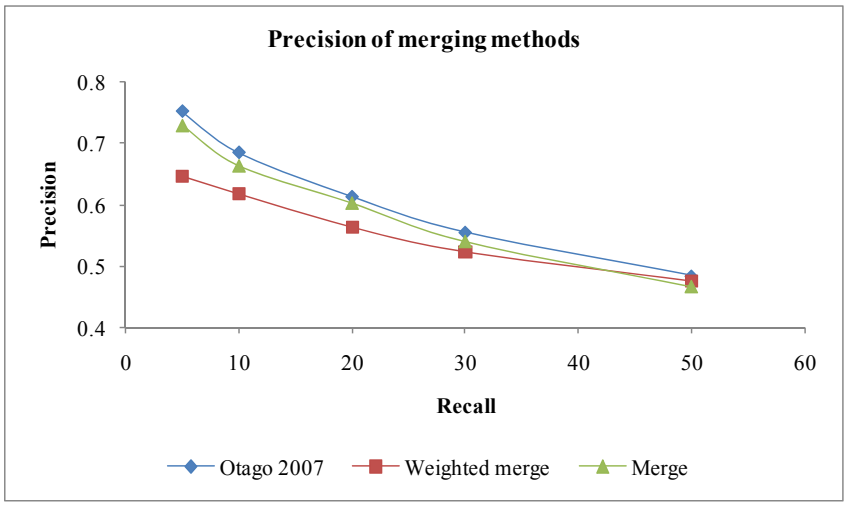


Fig. 3. Early precision scores for the three merging techniques. Although the MAP of weighted merge is highest, the early precision of Otago 2007 is highest.

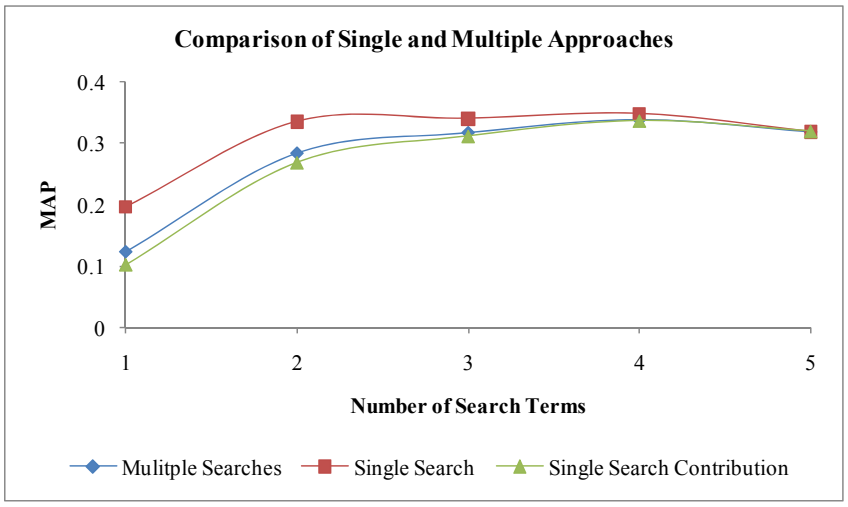


Fig. 4. A comparison of the multiple search technique to the single search technique suggests that the single search technique is best.

3.2.3. Improvements – Single Searches

With the multiple search technique the contribution of each separate search to the final precision score is unclear. It is also unclear whether or not a better approach is to simply perform one search with the given number of terms and to use the top 250 results.

Two experiments were conducted: in the first, n search terms were used and $n * 50$ results were retrieved; in the second, n search terms were used but the full 250 results were retrieved. The first experiment computes the contribution of the first search to the multi-search whereas the second compares multi-searching with single-searching. The results were compared to the multiple search technique without merging and without weighting.

Figure 4 shows the contribution of the first search is a substantial proportion of the final result of the multiple search approach. It also shows the superiority of the single search technique when the full 250 results are retrieved. The improvements decrease as the number of terms per query increases to 5 as the number of documents retrieved per query in the multiple query approach tends to the full 250.

3.2.4. Weighted Search Terms

The experiments examining multiple searches showed that MAP could be improved if the search terms were weighted by *st*. Improvements are therefore expected in the single search approach if the individual search terms in a single query are weighted. The weights could be taken from the *st* score, but we chose to learn weights using Genetic Algorithms [9].

Trotman [10] and later Robertson et al. [11] modify the term frequency component of BM25 to include a separate weight for each structure within a document. We use their approach to weight term frequencies based not on the structure, but on the position of the term in the query (where query terms are sorted in decreasing *st* score). The new term frequency score use in the BM25 equation, *tf*, is given by

$$tf = tft * c_q$$

where *tft* is the true term frequency of the term in the document; and c_q is a constant weight for a term at position q in the query, varying from 0 to 1.

If the weight of c_q is 0 then the search term will be discarded from the query. If it is 1 then the true term frequency will be used, otherwise the influence of the term frequency will be linearly scaled by c_q. Good values for c_q are expected to decrease as a function of distance from the start of the query, reaching 0 when adding new terms creates an ambiguous query.

Experiments were conducted to learn weights for queries of lengths between 2 and 10 search terms[2]. The population size was 50, crossover rate was 0.9, mutation rate was 0.05, and reproduction rate was 0.05. The learning was run for 10 generation. Elitism was used. Many iterations of the learning were conducted and the best weights of the best run were recorded.

[2] In the case of a single search term the weight has a scaling effect which does not affect the relative rank order of the results; and so has no effect on MAP.

For the best MAP score achieved for queries ranging from 2 to 10 search terms, Table 3 shows the weights that were learned. It can be seen that the first two terms are responsible for the majority of the performance.

Figure 5 shows that weighting search terms results in an increase in precision for all tested cases (with the exception of a single search term). It should be noted that the experiments over-fit the weights to the orphan documents; unfortunately there is an insufficient number of orphans (in the 2007 set) to conduct a traditional learn / validate / evaluate experiment.

Table 3. Best learned weights for different queriy lengths

Search Terms	Weights (from first to last term)
2	0.96, 0.95
3	0.99, 0.96, 0.04
4	0.97, 0.73, 0.05, 0.06
5	0.95, 0.83, 0.14, 0.1, 0.01
6	0.89, 0.97, 0.44, 0.41, 0, 0.06
7	0.8, 0.95, 0.75, 0.29, 0, 0.07, 0.25
8	1, 0.88, 0.14, 0.05, 0, 0.22, 0.08, 0.19
9	0.87, 0.81, 0.36, 0.26, 0, 0.22, 0.29, 0.2, 0.01
10	0.9, 0.99, 0.77, 0.55, 0.35, 0.08, 0.19, 0.16, 0, 0.19

Table 4. MAP scores of the runs using terms from different parts of the document

Run	MAP
Title	0.410
Overview	0.143
Document	0.080
Otago 2007	0.339
Weighted merge	0.350

3.2.5. Other Sources of Search Terms

The experiments thus far suggest that the best approach is to perform a single search using a small number (two or three) highly representative search terms to identify document that should point to the orphan. The approach to identifying terms involved identifying document themes by simple text processing techniques. Wikipedia documents, however, are structured and include a title as well as a brief overview of the content of the document. These document structures might be used as a method of identifying good representative document-thematic terms, or the whole document (as seen by others [12]) might be used.

The title of the Wikipedia document is held between <name> tags. These were processed to remove duplicate search terms and stop words, and then used as queries.

The overview of the Wikipedia document occurs as an untitled section before the first titled section. It was extracted by using all text before the first <title> tag of the document, stop words and duplicate terms removed and used as the query.

The full-text of the Wikipedia document can easily by extracted by removing all XML tags from the document, removing stop and duplicate words, and used as the query.

Figure 6 shows the effect on early recall of the different techniques. Selecting terms from the whole document is better than using the title which is better than the overview which in turn is better than the whole document. However, the result is somewhat different when the MAP scores are compared; Table 4 presents the MAP scores and it can be seen that using the title is better overall than the other approaches, even bettering the weighted merge approach from above.

3.3. Otago Link-the-Wiki 2008 Runs

Problematic and systemic with our experiments is the tradeoff of early precision with mean average precision. The best method to choose is dependent on the metric being used to score the runs. MAP was used in 2007 and we assume its use in 2008.

3.3.1. File-to-file linking

Three runs were submitted, each used BM25 (k_1=0.421, k_3=242.61, b=0.498)

capConstant-SingleSearchWeighted: outgoing links were identified using the Otago version of Itakura & Clarke with Π = 0.1. Incoming links were identified using the weighted merge method with 4 search terms and weights of 0.97, 0.73. 0.05 & 0.06.

capConstant-TitleOnly: outgoing links were identified using the Otago version of Itakura & Clarke with Π = 0.1. Incoming links we were identified using the title of the orphan.

nonCap-FirstPara: outgoing links were identified using the Otago version of Itakura & Clarke without Π. Incoming links were identified using the outline of the orphan.

3.3.2. Anchor-to-BEP linking

capConstant-SingleSearch-A2B: same as capConstant-SingleSearchWeighted.

capConstant-TitleOnly-A2B: same as capConstant-TitleOnly.

nCapConstant-WholeDocument-A2B: same as nonCap-FirstPara, but using the whole document for the query.

3.4 Wikilinking Results

At time of writing the results of the Link-the-Wiki track for 2008 had not been published.

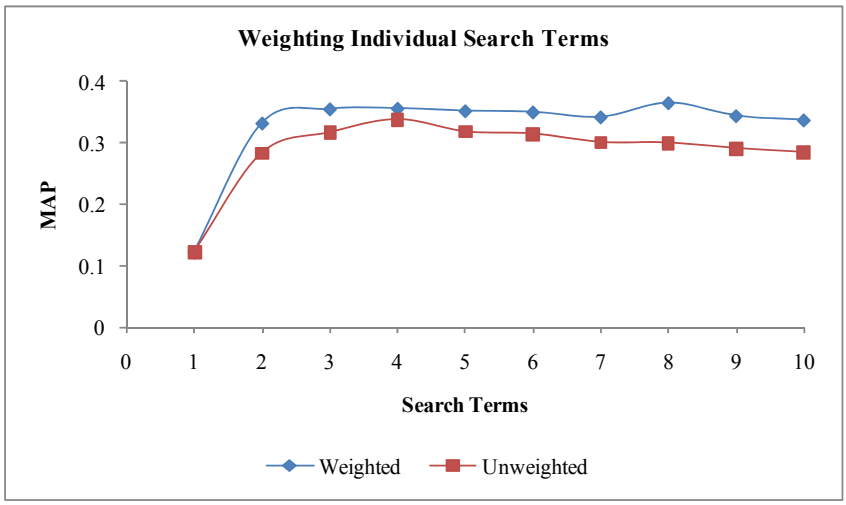


Fig. 5. Effect of weighting individual search terms in the query

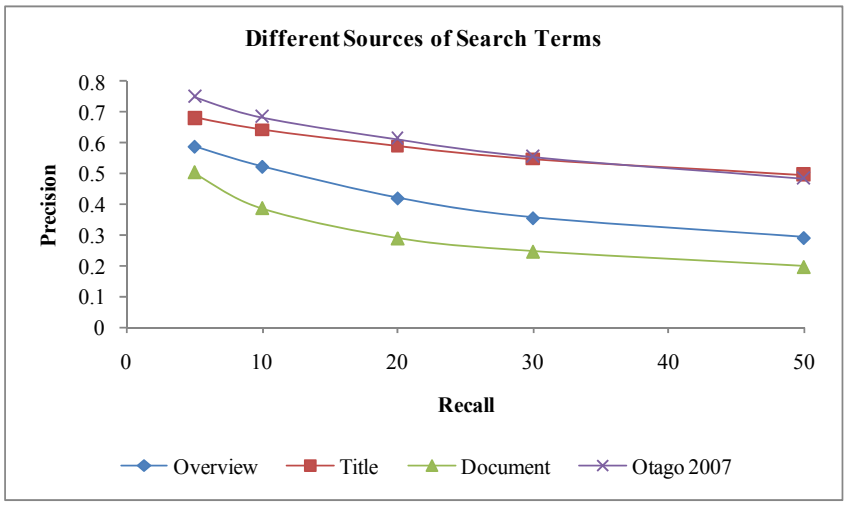


Fig. 6. Different sources of search terms. The title is a more effective source of terms than the overview which is better than the whole document. For early precision the best source was the approach used by Otago at INEX 2007

4. Conclusions

Experiments were conducted to gain insights into effective method of searching for the Relevance-in-Context task. In passage retrieval the Otago 2007 algorithm was compared to the Kullback-Leibler model, and virtually no difference was seen in the performance in the 2008 topics. This suggests the simpler Otago algorithm may be an effective alternative algorithm, especially when efficiency is an issue. In element retrieval the Beigbeder algorithm was compared to an IDF weighted variant and substantial improvements were seen on the 2008 topics – suggesting there is further room for improvement on Beigbeder's work.

In the Link-the-Wiki task the Itakura & Clarke algorithm was used for outgoing links. It was extended by removing punctuation from the anchors, and adding case sensitivity weighting. For incoming links an analysis of the Otago 2007 algorithm suggested that the method of just using the orphan title was effective. At time of writing the results for the Link-the-Wiki track had not been released.

Acknowledgements

Funded in part by a University of Otago Research Grant.

References

1. Jenkinson, D., Trotman, A.: Wikipedia Ad Hoc Passage Retrieval and Wikipedia Document Linking. Focused Access to XML Documents: Proceedings of INEX 2007 (2007) 426-439
2. Huang, W., Trotman, A., O'Keefe, R.A.: Element Retrieval Using a Passage Retrieval Approach. Australian Journal of Intelligent Information Processing Systems **9** (2006) 80-83
3. Itakura, K., Clarke, C.: From Passages into Elements in XML Retrieval. In: Trotman, A., Geva, S., Kamps, J. (eds.): SIGIR 2007 Workshop on Focused Retrieval (2007) 17-22
4. Beigbeder, M.: ENSM-SE at INEX 2007: Scoring with proximity. Preproceedings of INEX 2007 (2007) 53-55
5. Fuhr, N., Kamps, J., Lalmas, M., Malik, S., Trotman, A.: Overview of the INEX 2007 Ad Hoc Track. Focused Access to XML Documents: Proceedings of INEX 2007 (2007) 1-23
6. Itakura, K.Y., Clarke, C.L.: University of Waterloo at INEX2007: Adhoc and Link-the-Wiki Tracks. Focused Access to XML Documents: Proceedings of INEX 2007 (2007) 417-425
7. Geva, S.: GPX: Ad-Hoc Queries and Automated Link Discovery in the Wikipedia. Focused Access to XML Documents: Proceedings of INEX 2007 (2007) 404-416
8. Kamps, J., Koolen, M., Lalmas, M.: Where to Start Reading a Textual XML Document? : 30th SIGIR (2007)
9. Holland, J.H.: Adaptation In Natural and Artificial Systems. Univ. Michigan Press (1975)
10. Trotman, A.: Choosing Document Structure Weights. IP&M **41** (2005) 243-264
11. Robertson, S., Zaragoza, H., Taylor, M.: Simple BM25 extension to multiple weighted fields. 13th CIKM. (2004) 42-49
12. Fachry, K.N., Kamps, J., Koolen, M., Zhang, J.: Using and Detecting Links in Wikipedia. Focused Access to XML Documents: Proceedings of INEX 2007 (2007) 388-403

CSIR at INEX 2008 Link-the-Wiki Track

Wei Lu[1], Dan Liu[1], Zhenzhen Fu[1]

[1] Center for Studies of Information Resources (CSIR),
School of Information Management, Wuhan University, P. R. China
{reedwhu, DanLiu.whu, zhenzhenfu}@gmail.com

Abstract. In this year's Link-the-Wiki track, we totally submitted 4 runs, 2 for the file-to-file task and 2 for the anchor-2-best entry point task. For the incoming link detection, we use *p(d|t)*, the probability for the given topic file generate a document, to judge which documents are proper link sources for the given topic. For the outgoing link detection, in the file-to-file task, we simply use the document title matching strategy but with some proper reorder. In the anchor-to-best entry point task, we use *p(d|a,t)*, the probability for the given topic file and an anchor name generate a document, to select anchors and link targets for a given topic.

1 Introduction

This is the first time for the Center for Studies of Information Resources to participate in the INEX Link-the-Wiki track. We totally submit 4 runs, 2 for the file-to-file task and 2 for the anchor-to-best entry point task. The difference of each 2 runs lies in the incoming link detection approach.

For the incoming link detection, we express it as *p(d|t)*, the probability for the given topic file generate a document. Run 1 takes topic file title as *t*, and the other run takes the content of the topic file as *t*. The same incoming link detection method is used in the 2 sub tasks.

For the outgoing link detection, in the file-to-file task, we firstly identify all of the candidate anchors by matching each line of the topic file with all the document titles of the Wikipedia collection. Then order them by their incoming link number, and finally select the top 250 anchors. In the anchor-to-best entry point task, we express it as *p(d|a,t)*, the probability for the given topic file and an anchor name generate a document. The anchors are selected by a window-based technique, we select the top 50 anchors with the highest *p(d|a,t)* as final results.

The rest of this paper is structured as followed. Firstly, in section 2, we will briefly introduce the lately related work about link discovery in Wikipedia. Then, we will discuss our experiment setup in section 3. Our detail approaches will be introduced in section 4 and the evaluation results and a brief analyze will be presented in section 5, and finally we will summarize our work and discuss about the future work in section 6.

2 Related Work

Link-the-Wiki track was started last year. Only 4 participants submitted result last year: the Queensland University of Technology, the University of Waterloo, the University of Otago, and the University of Amsterdam [1].

After the INEX Link-the-Wiki track, the Queensland University of Technology and the University of Amsterdam had carried out a much deeper research.

As the track organizer, the Queensland University of Technology firstly summarized all the related work that have been done in INEX 2007's Link-the-Wiki track [1]. And then they improved their approaches used in INEX 2007, especially for the outgoing link found task, while in INEX 2007, their outgoing links were identified by running a window over the topic text and looking for matching page names in the collection [1], however, this time, they reordered those candidate anchors by some principles. For example, numbers and single terms have much less probability as an anchor [2]. They got a much better result when doing experiments with INEX 2007's topics.

In [3], the University of Amsterdam focused on outgoing links and investigated link density, and especially repeated occurrences of links with the same anchor text and destination. They used link density/anchor distance and repeated candidate links to assist link discovery, and performance was improved when using INEX 2007's topics as experiment topics. However, in [4], the University of Amsterdam mainly focused on finding out if Wikipedia's link structure can be exploited to improve ad-hoc information retrieval. After analyzed the relation between Wikipedia links and the relevance of pages, they experimented based on the test collection of INEX 2006 and found that Wikipedia's link structure really can help improve the effectiveness of ad-hoc retrieval.

Beyond the INEX, there are also lots of related research work has been done about automatic link generation. In [1, 2 and 3], the authors have summarized these related work, most of them are research about either automatic link generation in the World Wide Web or link detection and usage in Wikipedia. For details please refer to [1, 2, and 3].

3 Experiment Setup

3.1 Pre-process

In the pre-process stage, we mainly aimed at extracting useful information which may be helpful in the following experiment. After removed all the topic files, including 6600 for the file to file task and 50 for the anchor to best entry point task, we parsed the remaining xml documents and extracted nearly all the links. The links in the Wikipedia corpus mainly consist of 4 kinds: collection link, Wikipedia link, outside link and unknown link, as table 1 shows.

Table 1. The four main kinds of links of the Wikipedia corpus.

Link Type	Number	Percentage
Collection Link	17019137	77.4%
Wikipedia Link	176022	0.8%
Outside Link	858994	3.9%
Unknown Link	39310068	17.9%

Collection link is like:

```
<collectionlink
xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple"
xlink:href="21139.xml">
North America
</collectionlink>
```

Collection link is link between documents of Wikipedia corpus; about 77.4% of the links are collection link. In the pre-process stage, we mainly extracted information from collection link, including the third attribute--`xlink:href`, whose value is the target file of it, and the anchor name, in the above example it is "North America".

Collection link is very useful for analyzing the relationship between Wikipedia pages.

The other 3 kinds of links: wikipedia link, outside link and unknown link takes about 0.8%, 3.9% and 17.9% respectively. Wikipedia link likes in page anchors; outside link links to outside pages, mostly web pages outside the wikipedia website; unknown link is those should have been a useful link, but without proper target yet. For these 3 kinds of links, we only extracted the anchor name of the link. We haven't found any useful information in their attributes.

In all the extracted information, all the anchor names can be candidate anchor for the anchor-to-best entry point task. In our following experiment we use another way to get candidate anchors. However, the attributes of collection link are very useful. For every document in the wikipedia corpus, we find all the names of the anchors which link to the document, the article ids of the anchors, the article titles of the anchors, and save these information into separate text files.

3.2 Index and Retrieval Model

In the experiment, we use Lucene[5] and Lucene-Ex[6] as our search system. Lucene-Ex is an extended edition of Lucene supporting language model and some frequently used smoothing. So, there are two retrieval models in the experiment: the default similarity measure in Lucene, vector space model and language model.

Nick Craswell, David Hawking and Stephen Robertson once proved that: if we take the incoming anchor names as a short expression of the current document and append it to the document content, when retrieving related documents, the retrieval effect will be greatly improved [7]. So, in our experiment we append those incoming link anchor names to the topic file content when indexing.

For the index of the file-to-file task, we indexed the following fields:

(article id, incoming anchor names, content, content and incoming anchor names)

Where incoming anchor names is the sequential connection string of all the incoming link anchor names of the topic file, each separated by a full stop.

For the index of the anchor-to-best entry point task, we indexed the following fields:

(article id, title, incoming anchor names, content, content and incoming anchor names)

The difference between the file-to-file index and the anchor-to-best entry point index is that we take incoming anchor names as a separate field in the indexing for the latter.

4 Our Approaches

Link the wiki track has two sub tasks: file-to-file task and anchor-to-best entry point task. The anchor-to-best entry point task needs to find the anchor for link and the best entry point of the link target file. The following will introduce the approaches we take in the two sub tasks.

4.1 File-to-File Task

The file-to-file task needs to find 250 outgoings and 50 incomings for each topic of the 6600 topics. We submitted 2 runs: LTW_F2F_1 and LTW_F2F_2. The two runs differ from each other just in that their incomings are generated by different approaches. We used the same method to find outgoings.

4.1.1 Outgoing

Outgoing links of file level are generated by author mostly because a certain term in the current topic is just another file's title. So, we suppose that if a title of a document is in the current topic file, then there should be a link from the current topic file to the document. Additionally, we not only investigate the title of the document, but also all those different incoming anchor texts of the document, which are extracted in the pre-processing and can be taken as a somewhat transform of the title of the document.

After finding all those candidate outgoings, we order them by their probability as an outgoing. In the pre-process stage, the number of incoming links of each document has been calculated as shows in table 2. We take those who have more incoming links as having a greater possibility as outgoing links for a given topic. It means that we order those candidate documents by the number of incoming links they already have.

Table 2. Incoming links of the Wikipedia corpus.

Number of incoming links	Number of documents	Percentage
0	43554	6.61%
1-49	574108	87.07%
50-99	21401	3.25%

100-199	10801	1.64%
200-299	3565	0.54%
300-499	2749	0.42%
⩾500	3210	0.49%

4.1.2 Incoming

We assume that if two documents are about similar themes, then they can link to each other. So, we use *p(d|t)*, the probability of a topic file generating a document, to judge whether a document is appropriate as an incoming link.

When calculating *p(d|t)*, we use two different approaches. For run1 we use Lucene-Ex as search engine and language model as retrieval model; and for run2 we use Lucene and its default vector space model. Considering the speed of the Lucene-Ex retrieval system, we actually use the topic title as query and search the "content" field of the index. However, in run2, we use the main content of the topic file as query and search the "content" field of the index. The main content of the topic is generated by parsing the topic content and extract those typical terms.

4.2 Anchors to Best Entry Point Task

The anchor-to-best entry point task needs to find 50 anchors and 5 target file per anchor, totally 250 outgoings, and 50 incomings.

We submitted 2 runs: LTW_A2BEP_1 and LTW_A2BEP_2. The two runs differ from each other just in that their incomings are generated by different approaches.

4.2.1 Outgoing

Comparing with the file-to-file task, the anchor-to-best entry point task needs to determine the anchor of the link. We assume that if an anchor should link to a document just when the anchor and the topic file both relevant to the document. So, we express it as *p(d|a,t)*, the probability of a specified anchor and topic file generating the document. *P(d|a,t)* can be calculated as follows:

$$p(d \mid a,t) = p(d \mid a) * p(d \mid t) \quad (1)$$

Firstly, we choose those candidate anchors. After parse the topic file, exclude those words less than 5 letters, and tokenize the rest words, we use a window method to determine the candidate anchors. The size of the window differs from 1 to 7. There are more than 3000 candidate anchors for most topic files.

Secondly, for every candidate anchor, we calculate *p(d|a)*. This process includes 3 steps. (1)Determine whether there are documents just named as the candidate anchor. If the candidate anchor is exactly the same as the topic or one of the incoming anchor names of the document, then assign 1 to *p(d|a)*. (2) Use the candidate anchor as query to search "incoming anchor names" field of the index and get the top 50 documents. If there are less than 50 documents then do step 3. (3) Use the candidate anchor as query to search "content" field and get the top documents to make up for totally 50 documents.

Thirdly, we calculate *p(d|t)* and *p(d|a,t)*. In this process, we use the main content of the topic file as query to search the "content and incoming anchor names" field. Then, we can calculate *p(d|a,t)*.

At last, we order all anchors by the score of the candidate anchors, choose the top 50 anchors and then order these chosen anchors by their offset. The score of a candidate anchor can be calculated by either the highest target files' score of the anchor or the sum of the top 5 target files' score. In our runs, we use the former method. For anchors with the same offset, it means that they are overlapped, so we choose the anchor has the highest score. If they have the same score, then we choose the longest anchor.

Until now, we have found 50 anchors and each anchor with at most 5 target file links. The best entry point of the anchor in the target file is decided as follows: take the target file as a string, if the index of the anchor is not 0, then assign the index to the best entry point, or just assign it as 0.

4.2.2 Incoming

The approach use in finding incoming is the same as we use in finding incoming for the file to file task. Similarly, for run 1 we use Lucene-Ex as search engine and language model as retrieval model; and Lucene is used in run2.

However, in this task, we not only need to find the target file, but also the offset and length of the link anchor, and the best entry point in the topic file. We set all the best entry point of the topic file as 0. For the offset and length of the link anchor, we simply calculate the index of the topic title in the target link file and its length.

5 Result and Discussion

The primary evaluation result is as table 3. The detailed evaluation results hasn't released yet.

Table 3. The evaluation result of link the wiki track

Run ID	Type	MAP
CSIR_LTW_F2F_1	incoming	0.164555592
CSIR_LTW_F2F_2	**incoming**	**0.294029531**
CSIR_LTW_F2F	outgoing	0.008195767
CSIR_LTW_A2BEP_1	incoming	0.124536794100866
CSIR_LTW_A2BEP_2	**incoming**	**0.157729743721682**
CSIR_LTW_A2BEP	outgoing	0.0215233317243304

For finding incoming, we used two different approaches to calculate *p(d|t)*. The evaluation result reveals that when using the main content of the topic file as query is better than just using the title of the topic file.

For finding outgoings of the file-to-file task, we have ignored the fact that those document which have lots of incomings are mostly those talking about countries, places, years or lists. Though they have lots of incomings, we shouldn't take this as a sign of having a greater possibility as outgoing links for a given topic.

For finding outgoings of the anchor-to-best entry point task, we haven't anticipate this result. And currently, we haven't figure out the reason.

6 Conclusion and Future Work

In this year's Link-the-Wiki track, we submitted 2 runs for each of the sub-task. For the incoming link detection, we expressed it as *p(d|t)*, the probability for the given topic file generate a document. For the outgoing links of the file-to-file task, we simply used the document title matching strategy, and for the outgoing links of the anchor-to-best entry point task, we expressed it as *p(d|a,t)*, the probability for the given topic file and an anchor name generate a document.

From the primary evaluation result, we find that there are many problems in our approaches. In the future, we would try to study the research findings of automatic link generation in the 1990s, and some related research work about measures of the similarity between Wikipedia pages, and see if we can get some inspiration from these research findings when doing with the Link-the-Wiki track.

References

1. Darren Wei Che Huang , Yue Xu, Andrew Trotman. Overview of INEX 2007 Link the Wiki Track. 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: 373-387.
2. Darren Wei Che Huang, Andrew Trotman, Shlomo Geva. Experiments and Evaluation of link discovery in the wikipedia. Proceedings of SIGIR workshop on Focused Retrieval. 2008
3. Junte Zhang, Jaap Kamps. Link Detection in XML Documents: What about repeated links? Proceedings of SIGIR workshop on Focused Retrieval. 2008.
4. Jaap Kamps, Marijn Kollen. The importance of link evidence in Wikipedia. Advances in Information retrieval: 30th European Conference on IR Research (ECIR 2008) , Glasgow, UK, March 30-April 3, 2008: 270-282.
5. Lucene. The Lucene search engine,2007. http://lucene.apache.org/.
6. Lucene-Ex. An extension of Lucene for research use, http://sim.whu.edu.cn/newsim/lucene-ex/index.html.
7. Nick Craswell, David Hawking and Stephen Robertson. Effective Site Finding Using Link Anchor Information. Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. New Orleans, Louisiana, United States, 2001: 250 – 257.

Semi-supervised Categorization of Wikipedia collection by Label Expansion

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, for example, from Wikipedia collection proposed at the INEX XML Mining track. We cope with the case where there is a small number of labeled pages and a very large number of unlabeled ones. Due to the sparsity of the link based structure of Wikipedia, we apply the spectral and graph-based techniques developed in the semi-supervised machine learning. We use the content and structure views of Wikipedia collection to build a transductive categorizer for the unlabeled pages. We report evaluation results obtained with the label propagation function which ensures a good scalability on sparse graphs.

1 Introduction

The objective of the INEX 2008 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 2008, the Wikipedia categorization challenge has been indeed set in the semi-supervised mode, where only 10% 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 [2]. 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 four different aspects (or views) of a Wikipedia page:

Content - the set of words occured in the page.
Structure - the set of HTML/XML tags, attributes and their values in the page. These elements control the presentation of the page content to the viewer. In the extended version, we may consider some combinations of elements of the page structure, like the root-to-leaf paths or their fragments.
Links - the set of hyperlinks in the page.
Metadata - all the information present in the page Infobox, including the template, its attributes and values. Unlike the content and structure, not all pages present infoboxes [4].

We use these alternative views to generate a transductive categorizer for the Wikipedia collection. One categorizer representing the content view is based on the text of page. Another categorizer represents the structural view, it is based on the structure and Infobox characteristics of the page.

Thanks to the semi-supervised setting of the challenge, we test the graph-based semi-supervised methods which construct the similarity graph $W = (w_{ij})$ and apply a function propagating labels from labeled nodes to unlabeled ones. We first build the content categorizer, with weights w_{ij} being the textual similarity between two pages. We then build the structure categorizer, where weights w_{ij} are obtained from the structure and Infobox similarity between the pages. Finally, we linearly combine the two categorizers to get the optimal performance.

2 Graph-based semi-supervised learning

In the semi-supervised setting, we dispose labeled and unlabeled elements. In the graph-based approach [5, 6] to linked documents, one node in the graph represents one page. We assume a weighted graph G having n nodes indexed from 1 to n. We associate with graph G a symmetric weight matrix W where all weights are non-negative ($w_{ij} > 0$), and weight w_{ij} represents the similarity between nodes i and j in G. If $w_{ij} = 0$, there is no edge between nodes i and j.

We assume that the first l training nodes have labels, $y_1, y_2, \ldots, y_l$, where y_i are from the category label set C, and the remaining $u = n - l$ nodes are unlabeled. The goal is to predict the labels $y_{l+1}, \ldots, y_n$ by exploiting the structure of graph G. According to the *smoothness* assumption, a label of an unlabeled node is likely to be similar to the labels of its neighboring nodes. A more strongly connected neighbor node will more significantly affect the node.

Assume the category set C includes c different labels. We define the binary label vector Y_i for node i as $Y_i = \{y_{ij} | y_{ij} = 1 \text{ if } j = y_i, 0 \text{ otherwise}\}$. We equally introduce

the category prediction vector $\hat{Y}_i$ for node i. All such vectors for n nodes define a $n \times c$-dimensional score matrix $\hat{Y} = (\hat{Y}_1, \ldots, \hat{Y}_n)$. At the learning step, we determine $\hat{Y}$ using all the available information. At the prediction step, the category labels are predicted by thresholding the score vectors $\hat{Y}_{l+1}, \ldots, \hat{Y}_n$.

The graph-based methods assume the following:

1. the score $\hat{Y}_i$ should be close to the given label vectors Y_i in training nodes, and
2. the score $\hat{Y}_i$ should not be too different from the scores of neigbour nodes.

There exist a number of different graph-based methods [6]; we test some of them and report on one called the label expansion [5]. According to this approach, at each step, node i in graph G receives a contribution from its neighbors j weighted by the normalized weight w_{ij}, and an additional small contribution given by its initial value. This process can be expressed iteratively using the graph Laplacian matrix $L = D - W$, where $D = diag(d_i)$, $d_i = \sum_j w_{ij}$. The normalized Laplacian $\mathcal{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$ can be used instead of L to get a similar result. The process is detailed in Algorithm 1 below.

Algorithm 1 Label expansion

Require: Symmetric matrix $W, w_{ij} \geq 0$ (and $w_{ii} := 0$)
Require: Labels y_i for $x_i, i = 1, \ldots, l$
Ensure: Labels for $x_{l+1}, \ldots, x_n$
1: Compute the diagonal degree matrix D by $d_{ii} := \sum_j w_{ij}$
2: Compute the normalized graph Laplacian $\mathcal{L} := I - D^{-1/2} W D^{-1/2}$
3: Initialize $\hat{Y}^{(0)} := (Y_1, \ldots, Y_l, \mathbf{0}, \mathbf{0}, \ldots, \mathbf{0})$, where $Y_i = \{y_{ik} | y_{ik} = 1$ if $k = y_i$, 0 otherwise$\}$
4: Choose a parameter $\alpha \in [0, 1)$
5: **while** not converged to $\hat{Y}^{(\infty)}$ yet **do**
6: Iterate $\hat{Y}^{(t+1)} := \alpha \mathcal{L} \hat{Y}^{(t)} + (1 - \alpha) \hat{Y}^{(0)}$
7: **end while**
8: Label x_i by $argmax_j \hat{Y}_i^{(\infty)}$

It has been proved that Algorithm 1 always converges [5]. Indeed, the iteration equation can be represented as follows

$$\hat{Y}^{(t+1)} = (\alpha \mathcal{L})^{t+1} \hat{Y}^{(0)} + (1 - \alpha) \sum_{i=0}^{t} (\alpha \mathcal{L})^i \hat{Y}^{(0)}. \quad (1)$$

Matrix $\mathcal{L}$ is a normalized Laplacian, its eigenvalues are known to be in [-1, 1] range. Since $\alpha < 1$, eigenvalues of $\alpha \mathcal{L}$ are in (-1,1) range. Therefore, when $t \to \infty, (\alpha L)^t \to 0$.

Using the matrix decomposition, we have $\sum_{i=0}^{\infty} (\alpha \mathcal{L})^i \to (I - \alpha \mathcal{L})^{-1}$, so that we obtain the following convergence

$$\hat{Y}^{(t)} \to \hat{Y}^{(\infty)} = (1 - \alpha)(I - \alpha \mathcal{L})^{-1} \hat{Y}^{(0)}. \quad (2)$$

The convergence rate of the algorithm depends on specific properties of matrix W, in particular, the eigenvalues of its Laplacian $\mathcal{L}$. In the worst case, the convergence takes $O(kn^2)$ time, where k is the number of neighbors of a point in the graph.

On the other hand, the score matrix $\hat{Y}$ can be obtained by solving a large sparse linear system $(I - \alpha\mathcal{L})\hat{Y} = (1-\alpha)Y^{(0)}$. This numerical problem has been intensively studied [3], and efficient algorithms, whose computational time is nearly linear in the number of non-zero entries in the coefficient matrix. Therefore, the computation gets faster as the Laplacian matrix gets sparser.

2.1 Category mass regularization

Algorithm 1 generates a c-dimensional vector $\hat{Y}_i$ for each unlabeled node i, where c is the number of categories and each element $\hat{y}_{ij}$ between 0 and 1 gives a score for category j. To obtain the category for i, Algorithm 1 takes the category with the highest value, $argmax_j\ \hat{y}_{ij}$. Such a rule works well when categories are well balanced. However, in real-world data categories are often unbalanced and the categorization resulting from Algorithm 1 may not reflect the prior category distribution.

To solve this problem, we perform the category mass normalization, similarly to [7]. It rescales categories in such a way that their respective weights over unlabeled examples match the prior category distribution estimated from labeled examples.

Category mass normalization is performed in the following way. First, let p_j denote the prior probability of category j estimated from the labeled examples: $p_j = \frac{1}{l}\sum_{i=1}^{l} y_{ij}$. Second, the mass of category j as given by the average of estimated weights of j over unlabeled examples, $m_j = \frac{1}{u}\sum_{i=l+1}^{n} \hat{y}_{ij}$. Then the category mass normalization consists in scaling each category j by the factor $v_j = \frac{p_j}{m_j}$. In other words, instead of the decision function $argmax_j\ \hat{y}_{ij}$, we categorize node i in the category given by $argmax_j\ v_j\hat{y}_{ij}$. The goal is to make the scaled masses match the prior category distribution, i.e. after normalization we have that for all j

$$p_j = \frac{v_j m_j}{\sum_{i=1}^{c} v_i m_i}.$$

Generally, such a scaling gives a better categorization performance when there are enough labeled data to accurately estimate the category distribution, and when the unlabeled data come from the same distribution. Moreover, if there is an m such that each category mass is $m_j = mp_j$, i.e., the masses already reflect the prior category distribution, then the mass normalization step has no effect, since $w_j = \frac{1}{m}$ for all j.

2.2 Graph construction

The label expansion algorithm starts with a graph G and associated weighted matrix W. To build the graph G for the Wikipedia collection, we first reuse its link structure by transforming directed links into undirected ones. We analyze the number of incoming and outcoming links for all pages in the Wikipedia collection. Figure 1 shows the In-Out frequencies for the corpus; note the log scale set for all dimensions.

In the undirected graph, we remove self-links as required by Algorithm 1. We then remove links between nodes with high w_{ij} having different labels in order to fits the smoothness condition. It turns out that the link graph is not totally connected. Figure 2 plots the link graph with the help of the Large Graph Layout package[2]. The graph includes one connected component and about 160 small components covering less than 1% of collection. The right plot in Figure 2 additionally projects the category information on the link graph, where each category is shown by a particular color.

We also look for an approach of building graph G for Wikipedia collection different its link structure. The standard approach [5] is to build the k-NN (Nearest Neighbors) graph by taking the top k weights w_{ij} for each node. Unfortunately, the exhaustive k-NN procedure is infeasible for the Wikipedia corpus. So we build a graph G' by modifying G with randomly sampling of node pairs from Wikipedia and selecting the top k=100 ones per node. Note using the content or structure similarity will produce different versions of G'. In the evaluation section, we report results of tests run on both G and G' graphs.

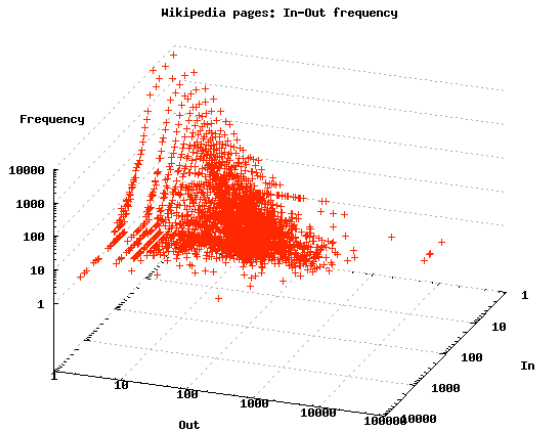


Fig. 1. Wikipedia nodes: In-Out frequencies.

Content matrix To generate a content weighted matrix W, we extract descriptor x_i for node i in the graph by using *"bag-of-words"* model and the *tf-idf* values, (term frequency-inverted document frequency) as $x_{ij} = tf_{ij} \cdot idf_i$, where

- tf_{ij} is the term frequency given by $\frac{n_{i,j}}{\sum_k n_{k,j}}$, where n_{ij} is the number of occurrences of the term in document d_j, and the denominator is the number of occurrences of all terms in document d_j.

[2] http://bioinformatics.icmb.utexas.edu/lgl/.

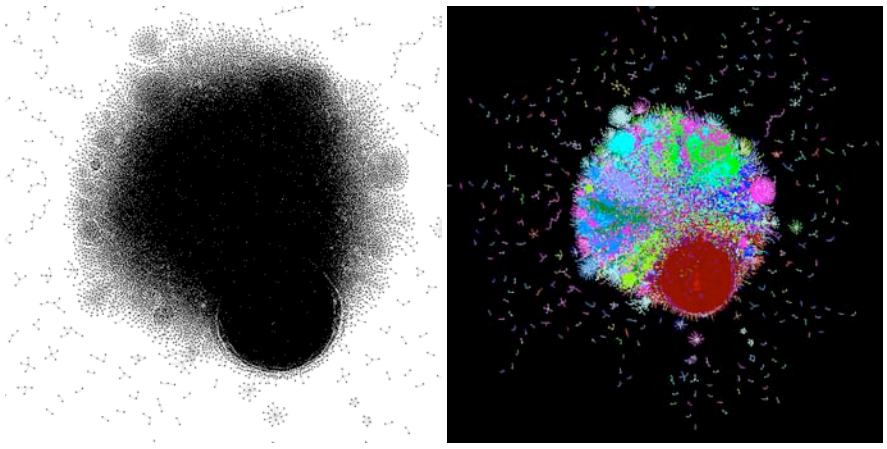


Fig. 2. Wikipedia corpus: the link graph plotted with LGL package.

– idf_i is the inverted document frequency $\log \frac{n}{|\{d_j : t_i \in d_j\}|}$, where n is the total number of documents and $|\{d_j : t_i \in d_j\}|$ is the number of documents where the term t_i occurs.

The tf-idf weighting scheme is often used in the vector space model together with cosine similarity to determine the similarity between two documents.

Layout matrix In the structure graph, node descriptors x_i are generated following the *"bag-of-tags"* approach which is similar to bag-of-words used in the content graph. Instead of words, it uses elements of the page structure. In the HTML formatted pages, the presentation is guided by instruction encoded by HTML tags, attributes and their values. The HTML structure forms a nested structure. The "bag-of-tags" model might have different instantiations, below we report some of them, where the terms form one of the following sets:

1. set of tag names, like `(table)` or `(font)`,
2. set of descendant tag pairs, like `(table,span)` or `(tr,td)`,
3. set of root-to-leaf paths in HTML page, like `(html,body,table,tr,td)`,
4. tag+attribute pairs, like `(table, font)`,
5. tag+attribute+attribute value triples, like `(table,font,times)`.

For any of these sets, we extract descriptors x_i for node i according to the conventional tf-idf weights. We build the weighted matrix W using the *structure similarity* between pages evaluated with "bag-of-tags" model and one of the listed tag sets.

Similarity measures Once we have obtained description vectors x_i for all nodes in graph G, we can get the weighted matrix W by measuring a similarity between two nodes i and j in G. Two possible measures are the following:

1. The Gaussian (RBF) kernel of width σ:

$$w_{ij} = exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right),$$

where the width σ is evaluated from the variance of the descriptors x_i.

2. The cosine function:

$$w_{ij} = \frac{x_i \cdot x_j}{||x_i||\ ||x_j||}.$$

3 Evaluation

The collection used in the INEX XML Mining challenge is composed of n=114,366 pages from the Wikipedia XML Corpus[3]; 10% of these pages have been annotated (l=11,437) with c=15 categories, 90% of pages (u=102,929) are unannotated. Some global characteristics of the corpus is given in Table 1. The word set is composed of all lexemized keywords; neither non-English words not stop were excluded.

Set	Size	Set	Size
Text words	727,667	Tag+attribute pairs	5,772
Infobox templates	602	Root-to-leaf paths	110,099
Infobox tags	1,208	Tag+attribute+value triples	943,422
Tags	1,257	Hyperlinks	636,187

Table 1. Wikipedia collection: some global characteristics.

In all experiments, we measure the accuracy of a transductive categorizer using 10-fold cross validation on the training set (in the presence of unlabeled data). As the baseline method, we used the semi-supervised learning with the transductive SVM [1], with x_i node descriptors being feature values. We also combine content, structure and infobox views, by concatenating the corresponding descriptors. However, direct concatenation of these alternative views brings no benefit. Table 2 reports the evaluation results.

For the label expansion method, we tested the link-based graph G and the sampling-enriched link graph G', with matrices W_c and W_s being generated with content or structure similarity measures, respectively. Using tag+attribute descriptors enriched with infoboxes generates a transductive categorizer whose performance is comparable to the content categorizer. Finally, the best performance is achieved by combining two graphs G' with weights w_{ij} obtained the content and structure similarity. The resulting weighted matrix is obtained as $W = \alpha W_s + (1 - \alpha)W_c$ with the optimal $\alpha = 0.34$ obtained by the cross validation. Table 2 reports the most important evaluation results.

[3] Available from http://www-connex.lip6.fr/ denoyer/wikipediaXML/.

TSVM Method	Accuracy(%)	LP Method	Accuracy (%)	Comment
Content	73.312	G-Content	72.104	Cosine
		G'-Content	75.03	idem
Tag+Attr	72.744	G'-Tag+Attr	72.191	Gaussian, δ=1.5
Paths	59.432	G'-Paths	64.824	idem
Tag+Attr+InfoBox	72.921	G-Tag+Attr+IB	70.287	idem
Content+Tag+Attr+IB	73.127	G'-Tag+Attr+IB	74.753	idem
		G'-Content + G'-TAIB	77.572	α=0.34

Table 2. Performance evaluation for different methods.

4 Conclusion

We applied the graph-based semi-supervised methods to the categorization challenge defined on Wikipedia collection. The methods benefit from the recent advances in spectral graph analysis and offer a good scalability in the case of sparse graphs. From the series of experiments on the Wikipedia collection, we may conclude that the optimal graph construction remains the main issue. In particular, the good choice of the graph generator and node similarity distance is a key to get an accurate categorizer. The use of the Wikipedia link graph offers the baseline performance, while the sampling technique brings a clear improvement. Nevertheless, its impact remains limited as the graph smoothness requirement is satisfied only partially. To better satisfy the requirement, we would need a smarter sampling technique and an extension of the method toward the graph regularization and an advanced text analysis.

5 Acknowledgment

This work is partially supported by the ATASH Project co-funded by the French Association on Research and Technology (ANRT).

References

1. T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. ICML '99*, pages 200–209, San Francisco, CA, USA, 1999.
2. D. Riehle. How and why Wikipedia works: an interview with Angela Beesley, Elisabeth Bauer, and Kizu Naoko. In *WikiSym '06: Proc. Intern. Symp. on Wikis*, pages 3–8, 2006.
3. Y. Saad. *Iterative Methods for Sparse Linear Systems, 2nd Edition*. SIAM, 2008.
4. F. Wu and D. S. Weld. Autonomously semantifying Wikipedia. In *ACM CIKM '07*, pages 41–50, New York, NY, USA, 2007.
5. D. Zhou, O. Bousquet, T. Navin Lal, J. Weston, and B. S. Olkopf. Learning with local and global consistency. In *Advances in NIPS 16*, pages 321–328. MIT Press, 2004.
6. X. Zhu. Semi-supervised learning literature survey. In *University of Wisconsin-Madison, CD Department, Technical Report 1530*, 2005.
7. X. Zhu, Z. Ghahramani, and J. Lafferty. Semisupervised learning using Gaussian fields and harmonic functions. In *Proc. ICML '03*, pages 912–919, 2003.

Probabilistic Methods for Link-based Classification at INEX'08

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

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

Abstract. In this paper we propose a method for link-based classification based on Bayesian networks and report the results obtained of its application to the XML Document Mining Track of INEX'08.

1 Introduction

This is the second year that the University of Granada participates in the Document Mining Track of the INEX Workshop. Our aim is, as in previous editions, providing a solution to the proposed problems on the framework of Probabilistic Graphical Models (PGMs).

The corpus given for 2008 differs slightly on the previous year one [3]. Again, as in 2007, it is a single label corpus (a subset of the AdHoc one [2] but using a different set of 16 categories); however, this year a file with the list of links between XML documents has been added. We will show that those links add relevant information for the categorization of documents.

Given that the 2008 corpus is coming from the same source (Wikipedia) than the 2007 one, we think that it might not be worthwhile to use the structural information of the documents for categorization, because we showed [1] that even using some very intuitive XML document transformations to flat text documents, classification accuracy was not improving, being worse in some of the cases. In this year, then, we have used a more pragmatic approach, directly ignoring structural information by simply removing XML tags from the documents.

2 Linked Files. Study on the Corpus

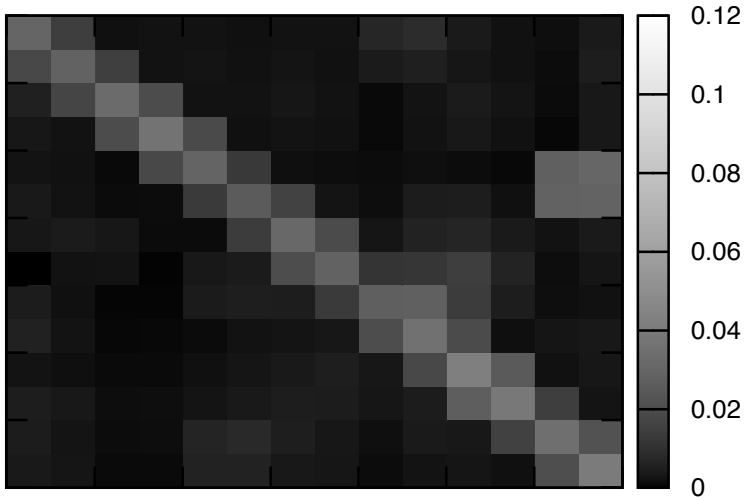
As it was said before, we are given a set of links between document files as additional training information, making some explicit dependencies arise between documents. Thus, it violates the "normal" assumption of traditional classification methods consisting in that the documents are independent of each other. This case of non independent documents can have different forms, and the relationships among documents can be not so regularly arranged (they form a general directed graph, not a tree nor a forest).

Clearly, for the problem of document categorization, that intra-corpus dependences could be ignored, applying "traditional" text categorization algorithms but, as we will show afterwards, the information from linked files can be a very valuable data.

But, how are those links supposed to help in the final process of text categorization? Obviously, not all kinds of links are equal, because they can give different information (even none). A careful review of those different kinds of dependencies represented by hyperlinks (*regularities*) is given by Yang [6], and following her terminology we can state that we are in a "encyclopedia regularity". We reproduce here her definition:

> *One of the simplest regularities is that certain documents with a class label only link with documents with the same class label. This regularity can be approximately found in encyclopedia corpus, since encyclopedia articles generally reference other articles which are topically similar.*

We have plotted, in the following figure, a matrix where the rows and columns are one of the 16 categories. Each matrix value $m_{i,j}$ represents the probability that a document of class i links a document of class j, estimated from the training document collection.



As it can be seen (the matrix has a strong weight in its diagonal), documents of one category tend to link documents of the same category.

3 Proposed method

The method proposed is an extension of a probabilistic classifier (we shall use in the experiments the Naive Bayes classifier but other probabilistic classifiers could also be employed) where the evidence is not only the document to classify, but this document together with the set of linked documents. Note that, in principle, we will try to use only information which is available in a natural way for a text classifier. Thinking in a batch processing of the different documents belonging to the corpus, the information easily available to a system, given a document, is the set of documents it links (not the set of documents that link it). Otherwise, it would be needed to have previously available a list of links between documents (which in this case we have, but it is not very realistic, in our opinion).

Consider a document d_0 which links with documents $d_1, \ldots, d_m$. We shall consider the random variables $C_0, C_1, \ldots, C_m$, all of them taking values in the set of possible category labels. Each variable C_i represents the event "The class of document d_i is". Let e_i be the evidence available concerning the possible classification of each document d_i (the set of terms used to index the document d_i or the class label of d_i). The proposed model can be graphically represented as the Bayesian network displayed in Figure 1.

Our objective is to compute the posterior probability $p(C_0|e)$, where e is all the available evidence concerning document d_0, $e = \{e_0, e_1, \ldots, e_m\}$. It can be proven that this probability can be expressed as follows:

$$p(C_0 = c_0|e) \propto p(C_0 = c_0|e_0) \prod_{i=1}^{m} \left(\sum_{c_i} p(C_i = c_i|c_0) \frac{p(C_i = c_i|e_i)}{p(C_i = c_i)} \right) \qquad (1)$$

As we can observe in equation (1), the posterior probability of C_0 has two components: a part which only depends on the evidence associated to the document d_0 to be classified ($p(C_0 = c_0|e_0)$) and another part related with the information about the class labels of each one of the documents linked with d_0 which can be obtained using its own local evidence ($p(C_i = c_i|e_i)$). This information is combined with the estimated probabilities of a linked document being of class c_i given that the document linking to it is of class c_0.

The posterior probabilities $p(C_0 = c_0|e_0)$ and $p(C_i = c_i|e_i)$ can be obtained using some standard probabilistic classifier, whereas the probabilities $p(C_i = c_i|c_0)$ can be estimated from the training data simply by computing the relative frequencies of the number of times that a document of class c_0 links to a document of class c_i.

Therefore, we can think of the proposed model as a method to modify the results offered by a base probabilistic classifier taking into account the information available about the linked documents and the relationships between categories.

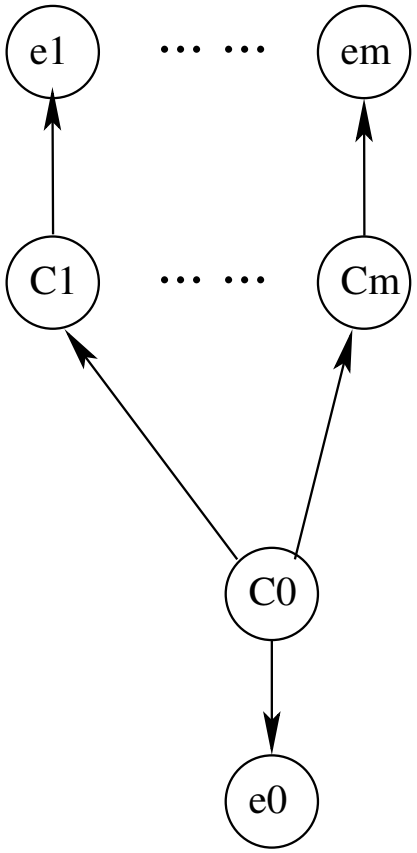


Fig. 1. Bayesian network representing the proposed model.

4 Experimental Results

To make the values comparable with the submitted runs, we have also performed some experiments on the test set in order to show the effectiveness (recall) of our approach. First of all we study the two submitted runs, a baseline (flat text classifier) and our proposal (combined with Naive Bayes):

- A classical Naive Bayes algorithm on the flat text documents: 0.67674 of recall.
- Our proposal using the previous Naive Bayes as the base classifier: 0.68136 of recall.

Although our method improves the baseline, the results achieved are not really significant. In order to justify the value of our model, we are asking now

ourselves which is the predicting power of our proposal, by making some additional computations in an "ideal setting". This "ideal setting" is, for a document being classified, to be surrounded with linked documents whose class membership is perfectly known (and hence we can set for a linked document d_k of category c_i, $p(C_k = c_i|e_k) = 1$ -the true class- and $p(C_k = c_j|e_k) = 0$ -the false categories- $\forall c_j \neq c_i$). Remember that, in previous experiments, a linked file whose category was not known should be first classified by Naive Bayes, and then that estimation (the output probability values) was used in our model.

So, the procedure is the following: for each document to classify, look to the linked files. For each linked file, if it is a training file, use that information (perfect knowledge), and if it is a test file, use also its categorization information taken from the test set labels to have our file surrounded by documents with perfect knowledge. This "acquired" knowledge is obviously removed for the next document classification.

In this "ideal setting" we have made two experiments: one combining naïve Bayes with our model (like the second one of the previous two), and one which combined a "blind classifier" (the one that gives equal probability to each category) with our model. The first should be better than the two previous ones, and the second one could give us an idea of the true contribution to the predictive power of our model, despite the underlying basic classifier used.

- Our proposal in an "ideal setting" using Naive Bayes as a base classifier: 0.69515 of recall.
- Our proposal in an "ideal setting" using a "blind classifier": 0.46500 of recall.

The first experiment provides the desired result: the recall is improved (but not so much). The small improvement could be due, in some part, to the extreme values given in this corpus by the Naive Bayes classifier (very close to 0 and 1). The introduction of these values in the final formula, as the first factor in the final posterior probability of each document, makes difficult to take into account (in the categories of the values close to 0) the information provided by the second factor (the combination of the information given by all the linked files), vanishing in some cases because of the low value of the first factor.

However, the second experiment showed us that, only using linked files information, and ignoring all content information of the document to classify, in this "ideal setting" of knowing the true class of each linked document, our method can reach a 0.46500 of recall. This is a value clearly high, that gives us the idea of the predictive power of using only link information of our model.

5 Conclusions and Future Works

We have proposed a new model for classification of linked documents, based on Bayesian networks. We have also justified the possibly good performance of the model in an "ideal" environment, with some promising results. Regrettably, our results in this track have been very discrete, reaching the final positions and not improving so much the naïve Bayes baseline.

To improve those poor results in the future, we could use a classifier (probabilistic) with a better performance. Such a classifier could be a logistic regression procedure, a higher dependence network or just a SVM with probabilistic output (using Platt's algorithm [5]). The probability assignments should also be "softer", in the sense that several categories should receive positive probability (naïve Bayes tended to concentrate all the probability in one category, zeroing the others and making the information provided by the links not useful, in some way).

Information provided by the inlinks (links received by one file) is also useful. A "symmetric procedure" can be defined with inlinks instead, and being tested. If the inlinks could be available, that information could result in a better performance. Even, a better model could be applied, using an undirected graph model (where we do not consider the direction of the links, and the documents linking a given document or linked by it, without distinction, are the set of *neighbors* of the document). Some of those experiments will probably be included in the final version of this paper although, as we exposed in the introduction, we do not consider that approach to be very realistic.

As future work we would like to study this problem as a collaborative classification problem (see, for instance [4]), and try to apply this method in one of the particular solutions (those that need a "local classifier") that are being given to it.

Acknowledgments. This work has been jointly supported by the Spanish Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía and Ministerio de Educación y Ciencia, under projects TIC-276 and TIN2005-02516, respectively.

References

1. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Probabilistic Methods for Structured Document Classification at INEX'07*. INEX 2007: 195–206, 2007.
2. L. Denoyer, P. Gallinari, *The Wikipedia XML Corpus*, SIGIR Forum **40**(1), 64–69, 2006.
3. L. Denoyer, P. Gallinari, *Report on the XML mining track at INEX 2007 categorization and clustering of XML documents*. SIGIR Forum **42**(1): 22–28, 2008.
4. Prithviraj Sen, Lise Getoor, *Link-based Classification*, Technical Report CS-TR-4858, University of Maryland, Number CS-TR-4858 - February 2007.
5. J. Platt, *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods*,, Advances in Large Margin Classifiers, A. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans, eds., MIT Press, (1999).
6. Y. Yang, S. Slattery, A study of approaches to hypertext categorization, Journal of Intelligent Information Systems, **18**, 219–241, 2002

Document Clustering with K-tree

Christopher M. De Vries and Shlomo Geva

Faculty of Information Technology,
Queensland University of Technology, Brisbane, Australia
chris@de-vries.ws
s.geva@qut.edu.au

Abstract. This paper describes the approach taken to the XML Mining track at INEX 2008 by the Queensland University of Technology. We introduce the K-tree clustering algorithm in an Information Retrieval context by adapting it for document clustering. Many large scale problems exist in document clustering. K-tree scales well with large inputs due to its low complexity. It offers promising results both in terms of efficiency and quality. Document classification was completed using Support Vectors Machines.

Key words: INEX, XML Mining, Clustering, K-tree, Cluster Tree, Balanced Tree, Vector Quantization, Tree Structured Vector Quantization, Text Classification, Support Vector Machine

1 Introduction

This paper describes an approach taken in the XML Mining track at INEX 2008. The track consists of two tasks, classification and clustering. Classification requires labeling documents based on a training set of labeled documents. Clustering requires grouping of similar documents given no other information than the documents themselves. The corpus consisted of 114,366 documents and 636,187 document to document links. Submissions were made for both tasks using several techniques.

We introduce K-tree in the Information Retrieval context. K-tree is a tree structured clustering algorithm [1]. The algorithm has previously been applied in the field of signal processing. It is particularly suitable for large collections due to its low complexity.

Non-negative Matrix Factorization (NMF) was also used to solve the clustering task. Applying NMF to document clustering was first described by Xu et. al. at SIGIR 2003 [2].

Negentropy has been used to measure clustering performance using the labels provided for documents. Entropy has been used by many researchers [3–5] to measure clustering results. Negentropy differs slightly but is fundamentally the same.

The classification task was solved using a multi-class Support Vector Machine (SVM). Similar approaches have been taken by [6, 7].

In Sect. 2, document representation is discussed. Section 3 describes the approach taken to the classification task and the associated results. In Sect. 4, evaluation of document clustering is discussed and negentropy is defined. Section 5 introduces the K-tree algorithm in an Information Retrieval setting and explains its use in the clustering task. In Sect. 6, NMF is discussed in relation to the clustering task. Section 7 reviews the clustering task and discusses the results. The document ends with a discussion of future research and a conclusion in Sects. 8 and 9.

2 Document Representation

Several sources of information were available for documents. A subset of the XML Wikipedia corpus [8] was selected. The documents contain content and structure as XML markup. A link graph had been extracted and labels for 10% of the collection were provided.

Document content was represented with TF-IDF [9] and BM25 [10]. Stop words were removed and the remaining terms were stemmed using the Porter algorithm [11]. The term frequencies in TF-IDF were normalized for document length. BM25 differs from TF-IDF by having two tuning constants $K1$ and b. $K1$ changes the influence of term frequency and b changes the influence of document length. The tuning parameters were set to the same values as used for TREC [10], $K1 = 2$ and $b = 0.75$.

Links were represented as a vector of weighted link frequencies. This resulted in a document to document link matrix. The row indicates the origin and the column indicates the destination document of a link. Each row vector of the matrix represents a document as a vector of link frequencies to other documents. The motivation behind this representation is that documents with similar content will link to similar documents. For example, in the current Wikipedia both car manufacturers BMW and Jaguar link to the Automotive Industry document. The link frequencies were weighted in the same way as TF-IDF. Term frequencies were simply replaced with link frequencies resulting in Link Frequency Inverse Document Frequency (LF-IDF). Link frequencies were normalized by the total number of links in a document.

All of the matrices were culled to reduce the dimensionality of the data. A feature's rank is calculated by summation of its associated column vector. Only the top n features are left in the matrix and the rest are discarded. For TF-IDF and BM25 the features are terms and for LF-IDF they are links to another document. TF-IDF was culled to the top 2000 and 8000 features. BM25 and LF-IDF were only culled to the top 8000 features.

3 Classification Task

The classification task was completed using a SVM and content and link information. This approach allowed evaluation of the different document representations. It allowed the most effective representation to be chosen for the clustering task.

SVMs traditionally create a binary partition through training examples. Instead of splitting the data set into one versus all separation for each label, $SVM^{multiclass}$ [12] was used. This program is an instance of SVM^{struct} which is a modified SVM^{light} to classify structured data. A mutliclass instance is the simplest instantiation SVM^{struct}. More complex structured output such as trees can be used as labels.

A SVM was trained with TF-IDF, BM25 and LF-IDF representations of the corpus. BM25 and LF-IDF feature vectors were concatenated to train on both content and link information simultaneously. Submissions were made only using BM25, LF-IDF or both because BM25 out performed TF-IDF.

3.1 Classification Results

Table 1 lists the results for the classification task. They are sorted in order of decreasing recall. The results for "Vries text only" differ from the official results. The original submission was incorrect. The submission was corrected and scored. Recall is simply the accuracy of predicting labels for documents not in the training set. Concatenating the link and content representations did not drastically improve performance. The difference between BM25 and BM25 concatenated with LF-IDF was 0.005 recall or 0.5%. However, LF-IDF performed reasonably well by itself achieving a recall of 0.62. A reasonable explanation of this outcome is that BM25 dominated the LF-IDF representation. Future work should investigate means to correct this. Examples of this are normalization of features before concatenation and re-weighting BM25 using link information.

Table 1. Classification Results Sorted by Recall

Name	Recall
Expe 1 tf idf TA	0.79
Expe 5 tf idf T5 10000	0.79
Expe 3 tf idf T4 10000	0.79
Vries text and links	0.78
Vries text only	0.78
Boris inex tfidf sim 037 it3	0.74
Boris inex tfidf sim 034 it2	0.73
Boris inex tfidf1 sim 0.38.3	0.73
Expe 4 tf idf T5 100	0.72
Kaptein 2008NBscoresv02	0.7
Kaptein 2008run	0.7
Romero nave bayes	0.68
Expe 2.tf idf T4 100	0.68
Romero nave bayes links	0.68
Vries links only	0.62

4 Document Cluster Quality

Measuring the quality of a clustering solution is difficult. This is particularly true for documents where many high level concepts can be associated. In the XML Mining track, clusters are compared to a known solution. Each document has been assigned a label and therefore purity can be calculated. This makes the evaluation practical, unlike a qualitative evaluation of clusters containing 100,000 documents.

The purity measure for the track is calculated by taking the most frequently occurring label in each cluster. Micro purity is the mean purity weighted by cluster size and macro is the mean. Taking the most frequently occurring label in a cluster discards the rest of the information represented by the other labels. Due to this fact negentropy was defined. It is the opposite of information entropy [13]. If entropy is a measure of uncertainty associated with a random variable then negentropy is a measure of certainty. Thus, the more labels of the same class that occur together the better. When all labels are evenly distributed across all clusters the lowest possible negentropy is achieved.

To define negentropy for a cluster we first need to define some variables. D is the set of all documents in a cluster. X is the set of all possible labels. x_i is the ith label in X. n is the total number of labels in X. c is the number of documents with label x_i in D. t is the total number of documents in D. $p(x_i)$ is the probability for label x_i. $H(D)$ is the negentropy for document cluster D. The negentropy for a cluster falls in the range $0 \leq H(D) \leq 1$ for any number of labels in X. Figure 1 shows the difference between entropy and negentropy. While they are exact opposites for a two class problem, this property does not hold for more than two classes. Negentropy always falls between 0 and 1 because it is normalized. Entropy is only bounded by the number of classes. As the number of classes increases, the difference between the maximum values for negentropy and entropy increase. Negentropy for an entire solution can be calculated in the same manner as micro and macro purity. Macro negentropy is the mean of all clusters and micro is the mean weighted by cluster size.

$$p(x_i) = \frac{c}{t} \tag{1}$$

$$H(D) = 1 + \frac{1}{\log_2 n} \sum_{\substack{i=1 \\ p(x_i)\neq 0}}^{n} p(x_i) \log_2 p(x_i) \tag{2}$$

The difference between purity and negentropy can easily be demonstrated with an artificial four class problem. There are six of each of the labels A, B, C and D. For each cluster in Solution 1 purity and negentropy is 0.5. For each cluster in Solution 2 the purity is 0.5 and the negentropy is 0.1038. Purity makes no differentiation between the two solutions. If the goal of document clustering is to group similar documents together then Solution 1 is clearly better because each label occurs in two clusters instead of four. Additionally, the grouping of labels is larger because they are split less.

Fig. 1. Entropy Versus Negentropy

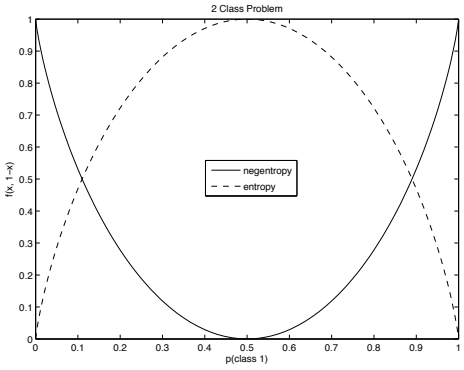


Table 2. Solution 1

Cluster	Label Counts
1	A=3, B=3
2	A=3, C=3
3	B=3, D=3
4	C=3, D=3

Table 3. Solution 2

Cluster	Label Counts
1	A=3, B=1, C=1, D=1
2	B=3, C=1, D=1, A=1
3	C=3, D=1, A=1, B=1
4	D=3, A=1, B=1, C=1

Fig. 2. Solution 1

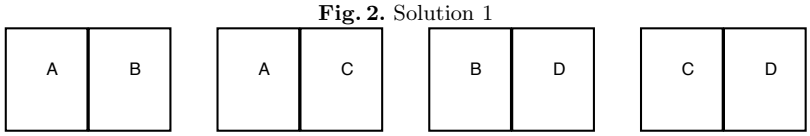
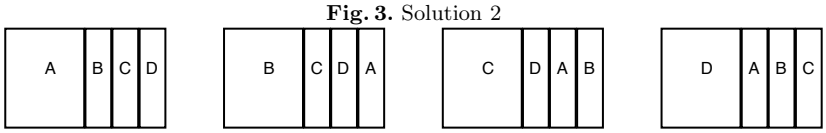


Fig. 3. Solution 2



5 K-tree

The K-tree algorithm is a height balanced cluster tree. It can be downloaded from http://sourceforge.net/projects/ktree. It is also referred to as a Tree Structured Vector Quantizer (TSVQ). It is inspired by the B^+-tree where all data records are stored in the leaves at the lowest level in the tree and the internal nodes form a nearest neighbour search tree. The k-means algorithm is used to perform splits when nodes become full. The constraints placed on the tree are relaxed in comparison to a B^+-tree. This is due to the fact that vectors do not have a total order like real numbers.

B^+-tree of order m

1. All leaves are on the same level.
2. Internal nodes, except the root, contain between $\lceil \frac{m}{2} \rceil$ and m children.
3. Internal nodes with n children contain $n-1$ keys, partitioning the children into a search tree.
4. The root node contains between 2 and m children. If the root is also a leaf then it can contain a minimum of 0.

K-tree of order m

1. All leaves are on the same level.
2. Internal nodes contain between 1 and m children. The root can be empty when the tree contains no vectors.
3. Codebook vectors (cluster representatives) act as search keys.
4. Internal nodes with n children contain n keys, partitioning the children into a nearest neighbour search tree.
5. The level immediately above the leaves form the codebook level containing the codebook vectors.
6. Leaf nodes contain data vectors.

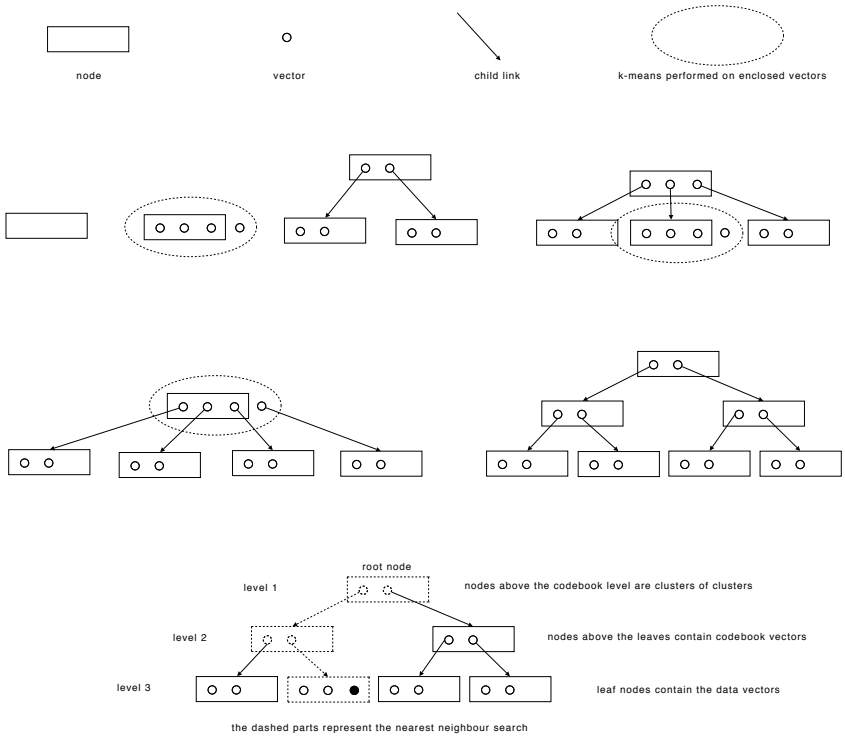
The leaf nodes of a K-tree contain real valued vectors. The search path in the tree is determined by a nearest neighbour search. It follows the child node associated with nearest vector. This follows the same recursive definition of a B^+-tree where each tree is made up of a smaller sub tree. The current implementation of K-tree uses Euclidean distance for all measures of similarity. Future versions will have the ability to specify any distance measure.

5.1 Building a K-tree

The K-tree is constructed dynamically as data vectors arrive. Initially the tree contains a single empty root node at the leaf level. Vectors are inserted via a nearest neighbour search, terminating at the leaf level. The root of an empty tree is a leaf so the nearest neighbour search terminates immediately, placing the vector in the root. When $m+1$ vectors arrive the root node can not contain any more keys. It is split using k-means where $k = 2$ using all $m+1$ vectors.

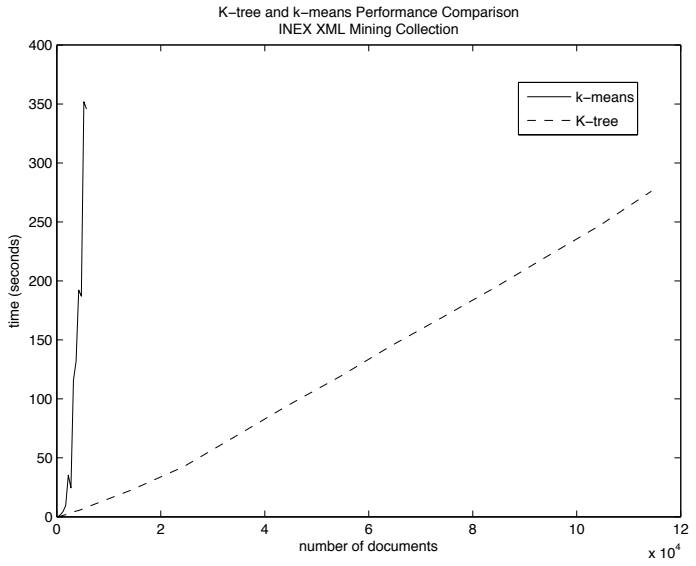
The two centroids that result from k-means become the keys in a new parent. New root and child nodes are constructed and each centroid is associated with a child. The vectors associated with each centroid from k-means are placed into the associated child. This process has created a new root for the tree. It is now two levels deep. The root has two keys and two children, making a total of three nodes in the tree. Now that the tree is two levels deep, the nearest neighbour search finds the closest centroid in the root and inserts it in the associated child. When a new vector is inserted the centroids are updated along the nearest neighbour search path. They are weighted by the number of data vectors contained beneath them. This process continues splitting leaves until the root node becomes full. K-means is run on the root node containing centroids. The keys in the new root node become centroids of centroids. As the tree grows internal and leaf nodes are split in the same manner. The process can potentially propagate to a full root node and cause construction of a new root. Figure 4 shows this construction process for a K-tree of order three ($m = 3$).

Fig. 4. K-tree Construction



The time complexity of building a K-tree for n vectors is $O(n \log n)$. An insertion of a single vector has the time complexity of $O(\log n)$. These properties comes from the tree structure of the algorithm. This allows the algorithm to scale efficiently with the number of input vectors. When a node is split, k-means is always restricted to $m+1$ vectors and two centroids ($k = 2$). Figure 5 compares k-means performance with K-tree where k for k-means is determined by the number of codebook vectors. The order, m, for K-tree was 50. Each algorithm was run on the 8000 dimension BM25 vectors from the XML mining track.

Fig. 5. K-tree Performance



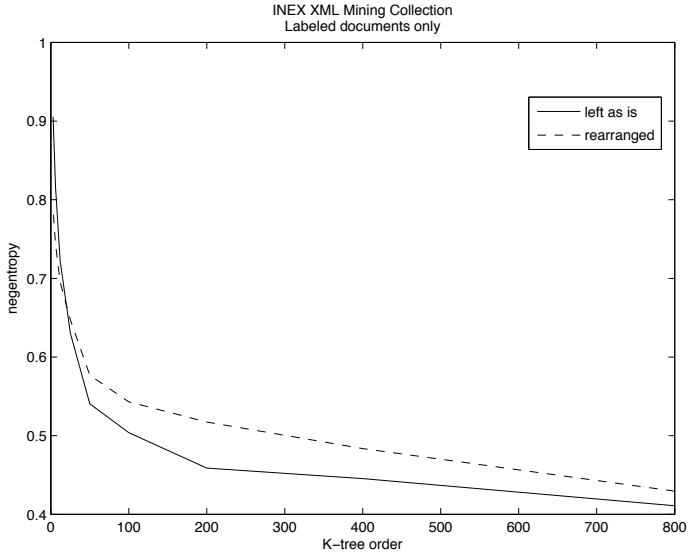
5.2 K-tree Submissions

K-tree was used to create clusters using the Wikipedia corpus. Documents were represented as 8000 dimension BM25 weighted vectors. This representation was used because it was most effective in the classification task. The K-tree was constructed using the entire collection. Cluster membership was determined by comparing each document to all centroids using cosine similarity. The track required a submission with 15 clusters but K-tree does not produce a fixed number of clusters. Therefore, the codebook vectors were clustered using k-means++ where $k = 15$. As k-means++ uses a randomised seeding process, it was run 20 times to find the solution with the lowest distortion. The k-means++

algorithm [14] differs improves k-means by using the D^2 weighting for seeding. Two other submission were made representing different levels of a K-tree. A tree of order 100 had 42 clusters in the first level and a tree of order 20 had 147 clusters in the second level. This made for a total of three submissions for K-tree.

Negentropy was used to determine the optimal tree order. K-tree was built using the documents in the 10% training set from the classification task. A tree was constructed with an order of 800 and it was halved each time until the smallest possible order was achieved. Each time negentropy was measured in the clusters represented by the leaf nodes. As the order decreases the size of the nodes shrinks and the purity increases. If all clusters became pure at a certain size then decreasing the tree order further would not improve negentropy. However, this was not the case and negentropy continued to increase as the tree order decreased. This suggests that the representation chosen does not match perfectly with the labels for documents. This can be seen in Fig. 6. The "left as is" line represents the K-tree as it is built initially. The "rearranged" line represents the K-tree when all the leaf nodes have been reinserted to their nearest neighbours without modify the internal nodes.

Fig. 6. K-tree Negentropy



Negentropy was calculated using the 10% training set labels provided on clusters for the whole collection. This was used to determine which order of 10,

20 or 35 fed into k-means++ with $k = 15$ was best. A tree of order 20 provided the best negentropy.

6 Non-negative Matrix Factorization

NMF factorizes a matrix into two matrices where all the elements are ≥ 0. If V is a $n \times m$ matrix and r is a positive integer where $rmin(n, m)$, NMF finds two non-negative matrices $W_{n \times r}$ and $H_{r \times m}$ such that $V \approx WH$. When applying this process to document clustering V is a term document matrix. Each column in H represents a document and the largest value represents its cluster. Each row in H is a cluster and each column is a document.

The projected gradient method was used to solve the NMF problem [15]. V was a 8000×114366 term document matrix of BM25 weighted terms. The algorithm ran for a maximum of 70 iterations. It produced the W and H matrices. Clusters membership was determined by the maximum value in the columns of H. NMF was run with r at 15, 42 and 147 to match the submissions made with K-tree.

7 Clustering Task

Every team submitted at least one solution with 15 clusters. This allows for a direct comparison between different approaches. It only makes sense to compare results where the number of clusters are the same. The K-tree performed well according to the macro and micro purity measures in comparison to the rest of the field. The difference in macro and micro purity for the K-tree submissions can be explained by the uneven distribution of cluster sizes. Figure 8 shows that many of the higher purity clusters are small. The same graphs are available for all submissions with 15 clusters in the appendix in Fig. 9. Macro purity is simply the average purity for all clusters. It does not take cluster size into account where micro purity does by weighting each purity in the average by the cluster size. Splitting the x-axis into three in Fig. 8 results in three categories of clusters. There are very high purity clusters that are easy to find. In the middle there are some smaller clusters that have varying purity. The larger, lower purity clusters in the last third are hard to distinguish. Figure 7 shows clusters sorted by purity and size. K-tree consistently found higher purity clusters than other submissions. Even with many small high purity clusters, K-tree achieved a high micro purity score.

The K-tree submissions were determined by the cosine similarity with the centroids produced by K-tree. The tree has an internal ordering of cluster as well. Future work will need to analyze the clusters that form in the tree.

8 Future Work

The work in this area falls into two categories, XML mining and K-tree. Further work in the XML mining area involves better representation of structure. For

Table 4. Clustering Results Sorted by Macro Purity

Name	Size	Macro	Micro
K-tree	15	0.59	0.49
QUT LSK 1	15	0.56	0.45
NMF	15	0.54	0.47
QUT LSK 3	15	0.53	0.49
QUT LSK 2	15	0.52	0.44
QUT Entire collection 15	15	0.51	0.49
QUT LSK 4	15	0.49	0.45
Hagenbuchner	15	0.26	0.38

Table 5. Clustering Results Sorted by Micro Purity

Name	Size	Micro	Macro
K-tree	15	0.49	0.59
QUT LSK 3	15	0.49	0.53
QUT Entire collection 15	15	0.49	0.51
NMF	15	0.47	0.54
QUT LSK 1	15	0.45	0.56
QUT LSK 4	15	0.45	0.49
QUT LSK 2	15	0.44	0.52
Hagenbuchner	15	0.38	0.26

Fig. 7. All Submissions with 15 Clusters

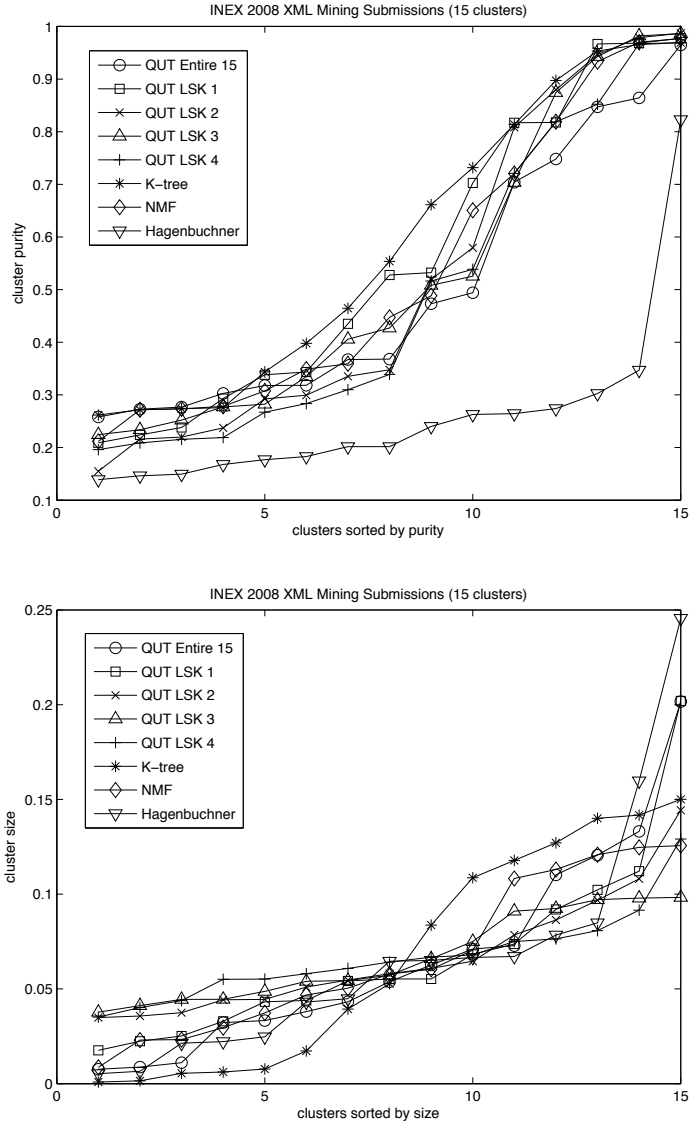
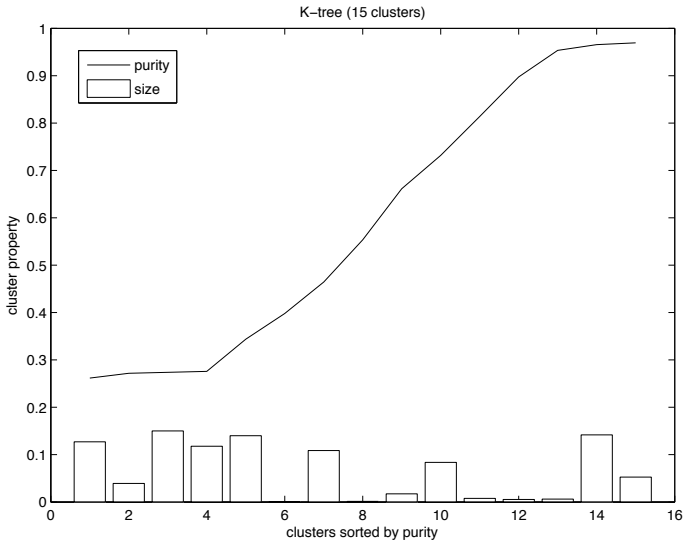


Fig. 8. K-tree Breakdown



example, link information can be included into clustering via a modified similarity measure for documents. Extra structure could be extracted from the content. For example, all people appearing in the text could be mined and added as XML markup. Future work with the K-tree algorithm will involve different splitting strategies, dynamic restructuring of the tree and other methods to try improve quality of clustering results by only a linear increase in complexity.

9 Conclusion

In this paper an approach to the XML mining track was presented, discussed and analyzed. The K-tree algorithm was applied to document clustering for the first time. The results show that it is well suited for the task.

References

1. Geva, S.: K-tree: a height balanced tree structured vector quantizer. Proceedings of the 2000 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing X, 2000. **1** (2000) 271–280 vol.1
2. Xu, W., Liu, X., Gong, Y.: Document clustering based on non-negative matrix factorization. In: SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, New York, NY, USA, ACM (2003) 267–273
3. Surdeanu, M., Turmo, J., Ageno, A.: A hybrid unsupervised approach for document clustering. In: KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, New York, NY, USA, ACM (2005) 685–690
4. Hotho, A., Staab, S., Stumme, G.: Ontologies improve text document clustering. Third IEEE International Conference on Data Mining, 2003. ICDM 2003. (Nov. 2003) 541–544
5. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. **34** (2000) 35
6. Joachims, T.: Text categorization with Support Vector Machines: Learning with many relevant features. In: Text categorization with Support Vector Machines: Learning with many relevant features. (1998) 137–142
7. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of Machine Learning Research **2** (2002) 45–66
8. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. SIGIR Forum (2006)
9. Salton, G., Fox, E.A., Wu, H.: Extended boolean information retrieval. Communications of the ACM **26**(11) (1983) 1022–1036
10. Robertson, S., Jones, K.: Simple, proven approaches to text retrieval. Update (1997)
11. Porter, M.: An algorithm for suffix stripping. Program: Electronic Library and Information Systems **40**(3) (2006) 211–218
12. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large Margin Methods for Structured and Interdependent Output Variables. Journal of Machine Learning Research **6** (2005) 1453–1484
13. Shannon, C., Weaver, W.: The mathematical theory of communication. University of Illinois Press (1949)
14. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2007) 1027–1035
15. Lin, C.: Projected Gradient Methods for Nonnegative Matrix Factorization. Neural Computation **19**(10) (2007) 2756–2779

A Appendix: All Results

Table 6. Clustering Results

Name	Size	Micro	Macro
K-tree	147	0.6	0.67
NMF	147	0.59	0.59
QUT Content from Freq struct 30	30	0.54	0.59
QUT Entire collection 30	30	0.54	0.58
NMF	42	0.54	0.57
QUT Content from Freq struct 30 using links	30	0.54	0.55
K-tree	42	0.53	0.6
QUT LSK 8	30	0.53	0.57
QUT LSK 6	30	0.53	0.57
QUT Content from Freq struct 15 using links	16	0.52	0.52
QUT LSK 7	30	0.5	0.53
QUT LSK 5	30	0.5	0.52
K-tree	15	0.49	0.59
QUT LSK 3	15	0.49	0.53
QUT Entire collection 15	15	0.49	0.51
QUT Content from Freq struct 15	16	0.48	0.49
NMF	15	0.47	0.54
Hagenbuchner	512	0.47	0.52
Hagenbuchner	512	0.46	0.54
QUT LSK 1	15	0.45	0.56
QUT LSK 4	15	0.45	0.49
QUT LSK 2	15	0.44	0.52
Hagenbuchner	15	0.38	0.26

Fig. 9. Breakdown of Submissions with 15 Clusters

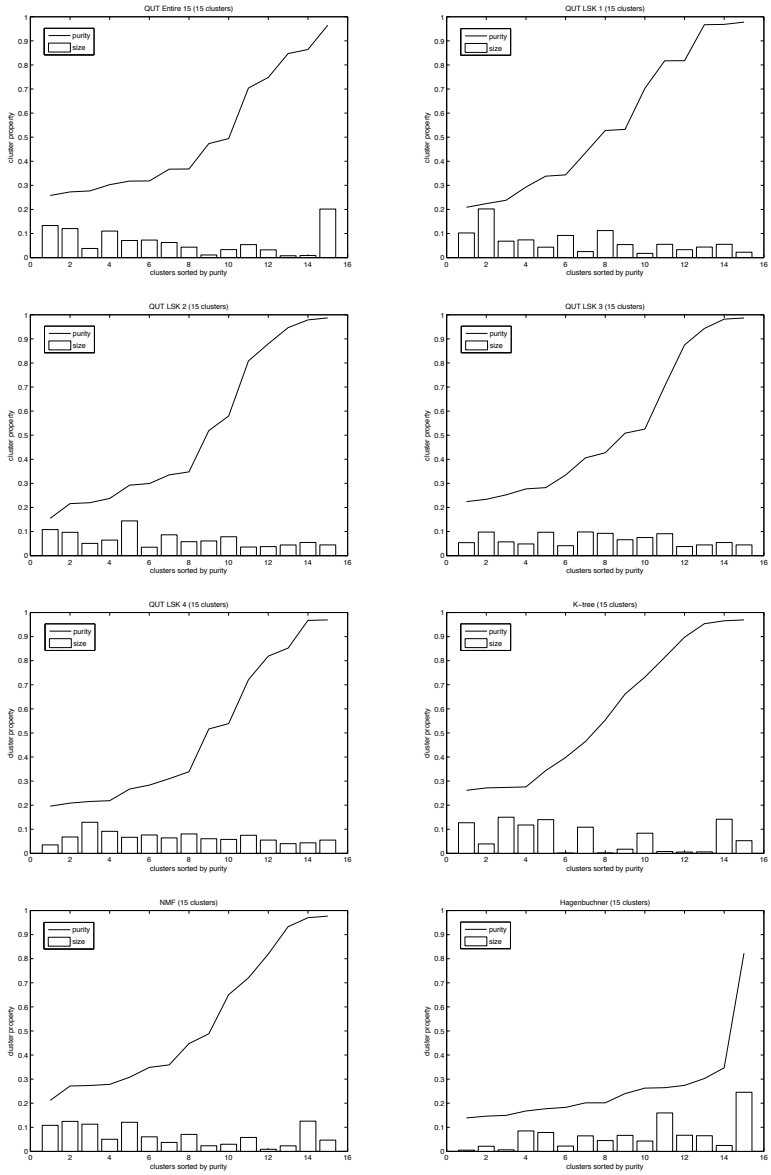


Fig. 10. All Submissions with 42 Clusters

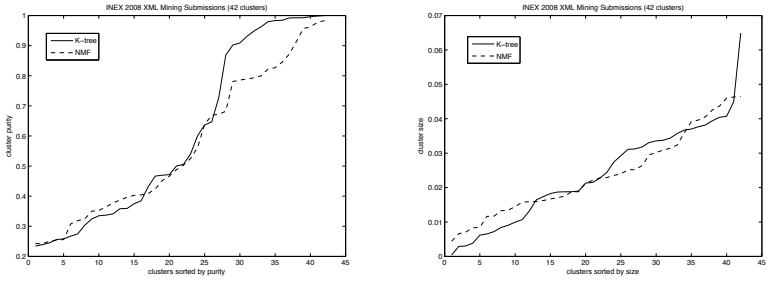


Fig. 11. Breakdown of Submissions with 42 Clusters

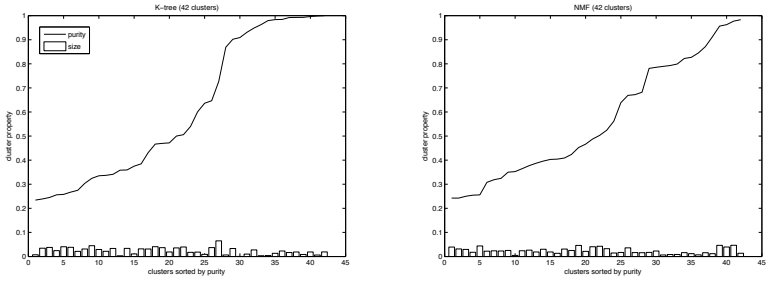


Fig. 12. All Submissions with 147 Clusters

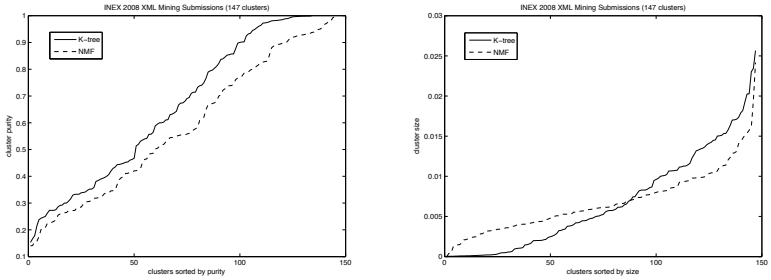
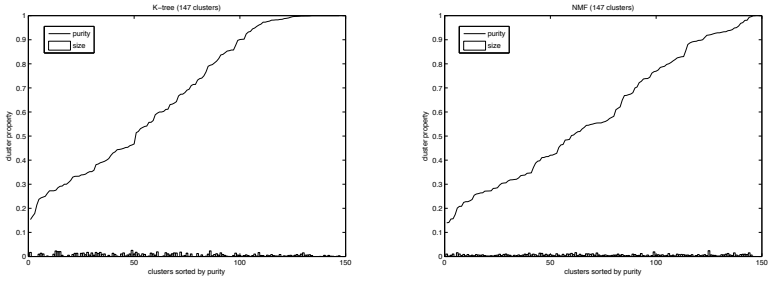


Fig. 13. Breakdown of Submissions with 147 Clusters



UJM at INEX 2008 XML mining track

Mathias Géry, Christine Largeron and Christophe Moulin

Université de Lyon, F-69003, Lyon, France
Université de Saint-Étienne, F-42000, Saint-Étienne, France
CNRS UMR5516, Laboratoire Hubert Curien
{mathias.gery, christine.largeron, christophe.moulin}@univ-st-etienne.fr

Abstract. This paper[1] reports our experiments carried out for the INEX XML Mining track, consisting in developing categorization (or classification) and clustering methods for XML documents. We represent XML documents as vectors of index terms. For our first participation, the purpose of our experiments is twofold: Firstly, our overall aim is to set up a categorization text only approach that can be used as a baseline for further work which will take into account the structure of the XML documents. Secondly, our goal is to define two criteria based on terms distribution for reducing the size of the index. Results of our baseline are good and using our two criteria, we improve these results while we slightly reduce the index term. The results are slightly worse when we reduce sharply the index of terms.

1 Introduction

The INEX XML Mining Track is organized in order to identify and design machine learning algorithms suited for XML documents mining. Two tasks are proposed: clustering and categorization. Clustering is an unsupervised process through which all the documents must be classified into clusters. The problem is to find meaningful clusters without any prior information. Categorization (or classification) is a supervised task for which, 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 one of the categories.

This second task is considered in this article. Moreover, even if the content information (the text of the documents), the structural information (the XML structure of the documents) and the links between the documents can be used for this task, we have exploited only the textual information. Indeed, this is our first participation to this track and our aim was to design a framework that could be used as a baseline for further works dealing with structured documents.

More precisely, we focus on the preprocessing step, particularly the feature selection, which is an usual step of the knowledge discovery process. On textual

[1] This work is supported by the project "Web Intelligence", Rhône-Alpes region, cf. http://www.web-intelligence-rhone-alpes.org

data, this step can be essential for improving the performance of the categorization algorithm. It exists a lot of words in the natural language, including stop words, synonymous, *etc.*. These words are not equally useful for categorization. Moreover, their distribution must also be considered. For example, words that appear in a single document are not useful for the categorization task.

So, we need to extract from the text a subset of terms that can be used to efficiently represent the documents in view of their categorization. In this paper, the documents are represented according to the vector space model (VSM). Our aim is to adapt some VSM principles, for example the measure of the discriminatory power of a term, to the categorization task. We propose two criteria based on terms distribution aiming at extracting the indexing terms from the training set corpora. After a brief presentation of the vector space model given to introduce our notations in section 2, these criteria are defined in the following section. Our categorization approach is described in section 3 while the experiments and the obtained results are detailed in sections 4 and 5.

2 Document model for categorization

2.1 Vector space model (VSM)

Vector space model, introduced by Salton and al. [2], has been widely used for representing text documents as vectors which contain term weights. Given a collection D of documents, an index $T = \{t_1, t_2, ..., t_{|T|}\}$, 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 $\boldsymbol{d_i} = (w_{i,1}, w_{i,2}, ..., w_{i,|T|})$ where $w_{i,j}$ represents the weight of the term t_j in the document d_i. In order to calculate this weight, formula TF.IDF can be used.

2.2 TF: term representativity

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 occurrences of all 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

This VSM is widely used for text mining and information retrieval, as well for free format document like scientific articles as for semi structured document written in markup languages like HTML or XML.

But, in the context of categorization, even for limited collections, the dimensionality of the index can be exceedingly large. For example, in INEX collection, 652'876 nontrivial words have been identified. This is a real problem for categorization since the terms belonging to this bag of words are not necessarily discriminatory features of the categories. So, we introduced two criteria (CC and CCE) in order to select a subset of T providing a description of the documents belonging to the same category. We consider that these terms must be very frequent in the documents of the category and, on the contrary, that they must be infrequent in the other categories.
Let df_j^k, the number of documents in the category C_k where term t_j appears, and f_j^k, the frequency of documents belonging to C_k and including t_j:

$$df_j^k = |\{d_i \in C_k : t_j \in d_i\}|, k \in \{1, ...r\} \quad (1)$$

$$f_j^k = \frac{df_j^k}{|C_k|} \quad (2)$$

The higher the number of documents of C_k containing t_j, the higher is f_j^k.

On the other hand, we consider that the number of documents where the term t_j appears in all the categories except C_k, can be an indicator of the discriminatory power of t_j for the category C_k. Thus, a first criteria, noted CC (Category Coverage), is computed as follows:

$$CC_j^k = \frac{(f_j^k)^2}{df_j}$$

If the value of CC_j^k is high, then t_j is a characteristic feature of the category C_k.

The frequency f_j^k considers the number of documents containing t_j but it does not take into account the number of occurrences of t_j in the category. It is

the reason why we consider also p_j^k the frequency of t_j in the category C_k and a measure commonly used in information theory, called entropy, which evaluates the purity of the categories for the term t_j. In the context of text categorization, it measures the discriminatory power of t_j. Let n_j^k be the number of occurrences of t_j in the documents of C_k and p_j^k the corresponding frequency:

$$n_j^k = \sum_{d_i \in C_k} n_{i,j} \qquad\qquad p_j^k = \frac{n_j^k}{\sum_{k=1,r} n_j^k}$$

The Shannon entropy E_j of the term t_j is given by [3]):

$$E_j = -\sum_{k=1,r} (p_j^k) * log_2(p_j^k)$$

The entropy is minimal, equal to 0, if the term t_j appears only in one category. We consider that this term might have a good discriminatory power for the categorization task. Conversely, the entropy is maximal if t_j is not a good feature for representing the documents *i.e.* if t_j appears in all the categories with the same frequency.

We propose a second criteria, denoted CCE (Category Coverage Entropy), combining f_j^k (from CC) and entropy. CCE is defined by:

$$CCE_j^k = (alpha * f_j^k) + (1 - alpha) * (1 - \frac{E_j}{MaxE})$$

where $alpha$ is a parameter and $MaxE$ is the maximal value of E. When the term t_j is characteristic of the category C_k, the value of the criteria is high.

For each category, a subset of the terms of T corresponding to the highest values of the criterion is build. Then, the index is defined as the union of these subsets.

4 Experiments

4.1 Collection INEX XML Mining

The collection is composed of 114,336 XML documents of the Wikipedia XML Corpus. This subset of Wikipedia represents 21 categories, each corresponding to one subject or topic. Each document of the collection belongs to one category. In the XML Mining Track, the training set is composed of 10% of the collection.

4.2 Preprocessing

The first step of the categorization approach that we propose, consists in a preprocessing 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

652'876 terms is build using the LEMUR software[2]. The Porter Algorithm [1] has also been applied in order to reduce different forms of a word to a common form. This operation reduces the index to 560'209 terms. However, it still remains a large number of irrelevant elements 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. The index obtained at this stage is denoted I. In our experiments, its size is reduced to 161'609 terms on all the documents of the collection and to 77'697 on the training set.

4.3 Features selection

However, as explained in the previous section, the terms of I are not necessarily appropriated for the categorization task inasmuch they are not discriminatory for the categories. This is the reason why our criteria based on entropy and on frequency are used to select more suited features. The terms were sorted according to CC and CCE and only those corresponding to the highest values are retained. In our experiments, the top 100 terms by class and the top 10'000 terms by class were considered for each criteria to build four index, denoted respectively CC_{100} and CC_{10000} using CC and CCE_{100} and CCE_{10000} using CCE. Table 1 indicates the size of these indexes.

Table 1. Indexes sizes

Index	number of words
I	77697
CC_{100}	1051
CC_{10000}	75181
CCE_{100}	909
CCE_{10000}	77580

Using one of these indexes, the content of a document is then represented by the $tf.idf$ vector model described in the first section.

The second step is the categorization step itself. Two usual methods of classification are used: Support Vector Machines (SVM) and k-nearest neighbors. Only the most promising results obtained with the SVM were submitted. SVM was introduced by Vapnik for solving two class pattern recognition problems using Structural Risk Minimization principal[4]. In our experiments, the SVM algorithm available in the Liblinear library[3] has been used. The results provided by this approach are presented in the next section.

[2] Lemur is available at the URL http://www.lemurproject.org
[3] http://www.csie.ntu.edu.tw/ cjlin/liblinear/ - L2 loss suport vector machine primal

5 Experimental results

This work has been done with a dual purpose: firstly develop a categorization text approach usable as a baseline for further work on XML categorization taking into account the structure, and secondly evaluate performances of this method using our selection features approach.

5.1 Global results

We have submitted 5 experiments using our 5 indexes presented in table 1. The global results of XML Mining 2008 are synthesized in table 2 (participant: LaHC).

Rank	Participant	Run	Recall	Documents
1	LaHC	submission.expe_5.tf_idf_T5_10000.txt	0.7876	102929
2	LaHC	submission.expe_3.tf_idf_T4_10000.txt	0.7874	102929
3	LaHC	submission.expe_1.tf_idf_TA.txt	0.7873	102929
4	Vries	Vries_classification_text_and_links.txt	0.7849	102929
5	boris	boris_inex.tfidf.sim.037.it3.txt	0.7379	102929
6	boris	boris_inex.tfidf1.sim.0.38.3.txt	0.7347	102929
7	boris	boris_inex.tfidf.sim.034.it2.txt	0.7309	102929
8	LaHC	submission.expe_4.tf_idf_T5_100.txt	0.7230	102929
9	kaptein	kaptein_2008NBscoresv02.txt	0.6980	102929
10	kaptein	kaptein_2008run.txt	0.6978	102929
11	romero	romero_naive_bayes_links.txt	0.6813	102929
12	LaHC	submission.expe_2.tf_idf_T4_100.txt	0.6770	102929
13	romero	romero_naive_bayes.txt	0.6767	102929
14	Vries	Vries_classification_links_only.txt	0.6232	102929
15	Vries	Vries_classification_text_only.txt	0.2444	92647

Table 2. Summary of all XML Mining results.

5.2 Baseline results

Our baseline corresponds to the first experiment (*expe_1*), which was ranked 3th with a quite good recall: 0.7873.

5.3 Selection features improves results

When we select 10000 terms for each class using CCE (*expe_3*) and CC (*expe_5*) criteria, we reduce the size of the index to respectively $77'580$ and $75'181$. This reduction is small compared to the size of the baseline index ($77'697$). However, it lets us to slightly improve our baseline to 0.7874 with CCE and 0.7876 with CC. These three runs obtained the three best results of the XML Mining challenge.

5.4 Selection features reduces indexes

The last two submitted runs correspond to the selection of the first 100 terms for each class using CCE ($expe\_2$) and CC ($expe\_4$). As presented in table 1, the size of the index is sharply reduced to 909 terms for CCE and $1'051$ for CC. This reduction respectively correspond to 85% and 74% of the size of the baseline index. Even if the obtained results are lower than the results obtained with larger indexes, they are still relatively good. Indeed, the obtained recall is 0.6770 with CCE and 0.7230 with CC.

6 Conclusion

We proposed a categorization text approach for XML documents that let us obtain a good baseline for further work. For now we just used CC and CCE criteria as a threshold to select terms in order to build the index. For future work, we aim to exploit the computed value of CC and CCE to improve the categorization. Moreover, we could use the structure information of XML documents represented by the links between document to improve even more the results.

References

1. M. F. Porter. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316, 1997.
2. G. Salton and M.J. McGill. *Introduction to modern information retrieval*, volume 27. McGraw-Hill, 1983.
3. C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948.
4. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

Combining the structure and content of XML documents for Clustering using frequent subtrees

Sangeetha Kutty Tien Tran Richi Nayak Yuefeng Li
Faculty of Science and Technology
Queensland University of Technology
GPO Box 2434, Brisbane Qld 4001, Australia

{s.kutty, t4.tran, r.nayak,y2.li}@qut.edu.au

Abstract. This paper presents the experimental study conducted over the INEX 2008 Document Mining Challenge corpus using both the structure and content for clustering the documents. Using the structural information of the XML documents, the concise common substructures known as the closed frequent subtrees are generated. The closed frequent subtrees are then used to extract the content from the XML documents. A matrix which contains the term distribution of the documents in the dataset is developed using the extracted content for clustering of the dataset. In spite of the large number of documents in INEX 2008 Wikipedia dataset, the proposed frequent subtree-based clustering approach was successful in clustering the documents. Another advantage of this technique is that there is a significant reduction in the dimensionality of the terms used for clustering without much loss in accuracy.

Keywords: Clustering, XML document mining, Frequent Mining, Frequent subtrees, INEX, Wikipedia, Structure and content.

1 Introduction

Due to the inherent flexibility in structure representation, XML *(eXtensible Markup Language)* is fast becoming a ubiquitous standard for data representation and exchange on the Internet as well as in Intranets. The simplicity and the self describing nature of XML documents have promised its acceptance in a wide range of industries from education to entertainment and business to government sectors.

With the rapid growth of XML documents, there arise many issues concerning the management of these documents effectively. Clustering task has been perceived as an effective solution to organise these documents. It basically involves grouping XML documents based on their similarity without any prior knowledge on the taxonomy[2]. Clustering has been frequently

applied to group text documents based on the similarity of its content. However, XML document clustering presents a new challenge as the document contains structural information with text data (or content). The structure of the XML documents is hierarchical in nature and it represents the relationship between the elements at various levels.

Clustering of XML documents is a challenging task[2] as there are two important dimensions for XML documents namely structure and content. Most of the existing approaches do not focus on utilising these two dimensions due to increased complexity. However in order to achieve meaningful clustering results, it is essential to utilise both the dimensions of XML documents. This study not only combines the structure and content of XML documents effectively but also provides an approach that helps to reduce the dimensions for clustering without much loss in accuracy.

In this paper, we utilise the PCITMiner[3] (Prefix-based Closed Induced Tree Miner) to generate the closed frequent induced(CFI) subtrees. CFI is then utilised to extract the content information from the XML documents. The extracted content information which contains the document terms is represented in the Vector Space Model(VSM). Pair-wise clustering is then applied on the VSM to group the XML documents.

The assumption that we have made in this paper, based on the previous research [4], is that the structure of XML documents in the absence of their schema could be measured using frequent subtrees. Also, the content corresponding to the frequent subtrees are important, whereas, the content of the infrequent subtrees are redundant and hence can be removed.

The rest of this paper is organized as follows: Section 2 provides the overview of our approach. Section 3 covers the details about the pre-processing of structure and Section 4 on the structure mining to extract the common substructures among the XML documents. Section 5 discusses extraction of content using the common substructures and the clustering process is covered in Section 6. In Section 7, we present the experimental results and discussion and conclude in Section 8.

2 Combining the Structure and content non-linearly: An Overview

As illustrated in Fig.1 there are four major phases in our proposed method. The first phase of the proposed method is the pre-processing od the document structure. The structure of XML documents is modelled as *document trees*. Each *document tree* contains nodes which represent the tag names. PCITMiner[3] is then applied to generate the closed frequent induced (CFI)

subtrees from the *document trees* in the second phase for a given support threshold.

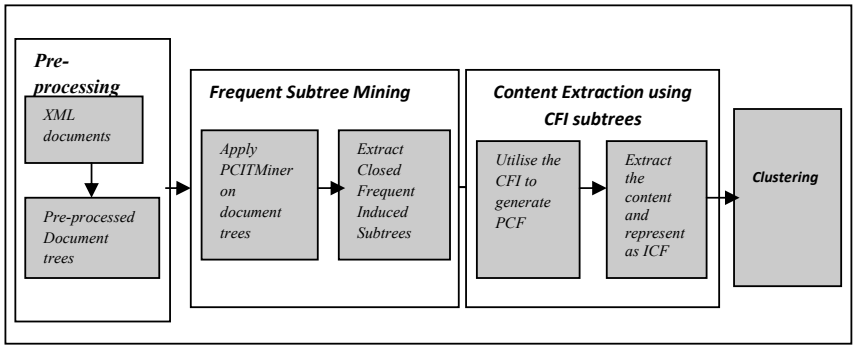


Fig. 1. Overview of the clustering method.

The CFI subtrees distribution is represented as a Pre-Cluster Form(PCF). Using the PCF in the third phase, the content is then extracted and pre-processed. The pre-processed content is represented as the Intermediate Cluster Form(ICF) Representation which is the distribution of the content or the terms corresponding to the CFI subtrees for documents in the corpus. It is basically a term-document matrix, $TDoc_{|Term|\times|DT|}$, where Term represents the terms corresponding to the CFI subtrees and *DT* represents the document trees in the given document tree collection. Each cell in the *TDoc* matrix represents the number of occurrences of the terms for the set of closed frequent subtrees in a given XML document. This matrix is used in the final phase to compute the similarity between the XML documents for the clustering of the dataset. The next section describes the phases of the proposed method in more detail.

3 Phase 1: Pre-processing of Structure

In the pre-processing phase, the XML document is decomposed into a tree structure with nodes representing only the tag names. The tag names are then mapped to unique integers for ease of computation. The semantic and syntactic meanings of tags are ignored since the Wikipedia documents conform to the same schema using the same tag set. Additionally, previous research has shown that the semantic variations of tags do not provide any significant contribution in the clustering process [2, 4]. Other node information such as data types, attributes and constraints is also ignored.

The structure of the XML document has many representations depending on its usability such as graph, tree, path, etc. We have chosen to use the tree

representation over path representation, which has been used by a number of researchers[2, 5], since it preserves the sibling information of the nodes. As shown in Fig. 2, the pre-processing of XML documents involves three sub-phases:

1. Parsing
2. Representation
3. Duplicate branches removal.

3.1 Parsing

Each XML document in INEX Wikipedia corpus is parsed and modelled as a rooted labeled ordered *document tree*. As the document tree has a root node and all the nodes are labeled using the tag names it is *rooted* and *labeled*. The left-to-right *ordering* is preserved among the child nodes of a given parent in the document tree and therefore they are ordered.

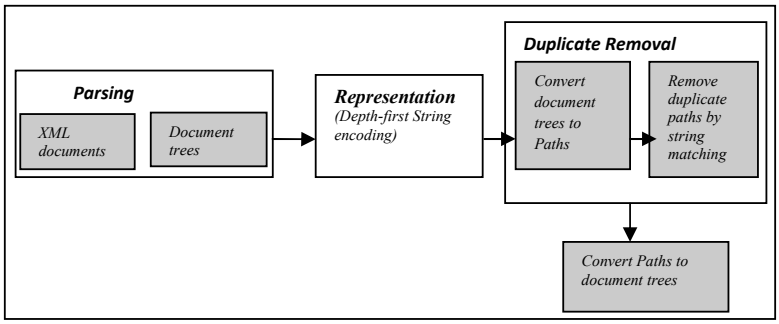


Fig. 2. The Pre-processing phase

3.2 Representation

The document trees need to be represented in a way that is suitable for mining in the next phase. A popular representation for tree is the depth-first string format[6] which is used to represent the document trees in this method. The *depth-first string encoding* traverses a tree in the *depth-first order*. It represents the *depth-first traversal* of a given document tree in a string like format where every node has a "–1" to represent backtracking and "#" to represent the end of the string encoding. For a document tree T with only one node r, the depth-first string of T is $S(T) = l_r\#$ where l is the label of the root node r. For a document tree T with multiple nodes, where r is the root node and the children nodes of r are $r_1,...,r_k$ preserving left to right ordering, the depth-first string of T is $S(T)= l_r l_{r_1}\text{-}1\ l_{r_2}\text{-}1 \ldots l_{r_k}\text{-}1\#$.

3.3 Duplicate branches removal

An analysis of the INEX Wikipedia dataset reveals that a large number of *document trees* contain duplicate branches. These duplicate branches are redundant information and hence they could cause additional overhead in the mining process. In order to remove the duplicate branches, the document tree is converted to a series of paths. The duplicate paths of the document trees are identified using string matching and removed. The remaining paths are combined together to create the document trees without any duplicate branches.

4 Phase 2: Structure Mining

The structure mining phase involves the mining of the frequent subtrees. Instead of utilising all the structures for the content extraction, we need to identify only the frequent or common subtrees as the content corresponding to the infrequent subtrees are redundant. In order to do so, frequent subtree mining is first applied on the document trees to identify frequent subtrees for a given user-specified support threshold. However, there could be a very large number of frequent subtrees generated at lower support threshold. In order to control the explosion we utilise closed frequent subtrees which are condensed representations of frequent subtrees without any information loss[7]. Frequent subtree mining on XML documents can be formally defined as follows:

Problem definition for the frequent subtree mining on XML documents

Given a collection of XML documents $D = \{D_1, D_2, D_3, \ldots, D_n\}$ modelled as document trees $DT = \{DT_1, DT_2, DT_3, \ldots, DT_n\}$ where n represents the number of XML documents or document trees. There exists a subtree $DT' \subseteq DT_k$ that preserves the parent-child relationship among the nodes as that of the document tree DT_k. This type of subtrees are called as induced subtrees.

Support(DT') (or frequency(DT')) is defined as the number of document trees in DT where DT' is an induced subtree. An induced subtree DT' is frequent if its support is not less than a user-defined minimum support threshold. In other words, DT' is a frequent induced subtree in the document trees in DT such that,

$$frequency\ (DT')/|DT| \geq min\_supp \qquad (1)$$

where *min_supp* is the user-given support threshold and $|DT|$ is the number of document trees in the document tree dataset *DT*.

Due to the large number of frequent subtrees generated at lower support thresholds, recent researchers have focused on using condensed representation without any information loss [3]. The popular condensed representation is the closed frequent subtrees which is defined as follows.

Problem definition for Closed Frequent subtree

For a given document tree dataset, $DT = \{DT_1, DT_2, DT_3, \ldots, DT_n\}$, if there exists two frequent induced subtrees DT' and DT'', the frequent induced subtree DT' is closed of DT'' iff for every $DT' \supseteq DT''$, $supp(DT') = supp(DT'')$ and there exists no superset for DT' having the same support as that of DT'. This property is called as *closure*.

In order to generate the closed frequent induced subtrees from the pre-processed document trees, the PCITMiner[3] is utilized. This algorithm adopts the partition-based approach to determine the CFI subtrees. Having generated the CFI subtrees, their distribution in the corpus is modelled as a Boolean subtree-document matrix, denoted by $CD_{|CFI| \times |DT|}$, where *CFI* subtrees represents the closed frequent induced subtrees and *DT* represents the document trees in the given document tree collection. Each cell in the *CD* matrix has a Boolean value to indicate the presence or absence of a given closed frequent induced subtree $\{cfi_1, cfi_2, \ldots, cfi_n\}$ in the document tree $\{DT_1, DT_2, DT_3, \ldots, DT_n\}$. Fig. 3 shows a $CD_{|CFI| \times |DT|}$ with closed frequent induced subtrees $\{cfi_1, cfi_2, cfi_3\}$ in the document trees $DT = \{DT_1, DT_2, DT_3, DT_4\}$.

	DT_1	DT_2	DT_3	DT_4
cfi_1	**1**	**0**	**1**	**1**
cfi_2	**0**	**1**	**0**	**1**
cfi_3	**1**	**1**	**1**	**0**

Fig. 2. *CD* matrix

This Pre-cluster Form (PCF) stores the structural information of the XML documents and is used to extract the content from each of the XML documents.

5 Phase 3: Content Extraction and pre-processing

Using the CD matrix, the content of the documents is extracted. The extracted content is then pre-processed using the following steps:

1. Stop-word removal
2. Stemming
3. Integer removal
4. Shorter length words removal

5.1 Stop-word removal

Stop words are words that are considered poor as index terms [8], for example words which occur very frequent such as 'the', 'of', 'and', etc., are needed to be removed prior to performing any natural language processing or data analysis. The most common stop list available for English text is from Christopher Fox which contains a list of 421 words [9]. Fox's stop list includes variants of the stop words, e.g. the word 'group' with its variants: 'groups', 'grouped' and 'grouping'.

However, there is always a pitfall in using the common stop list without any modification to suit to the domain under investigation. For example, the use of common stop list causes the removal of the word 'back' even though 'back' (a part of body) is a useful term in medical domain. It is therefore essential to customize the stop list considering the domain specific knowledge in order to avoid removing important words. In this research, the stop word list has been customised considering the tag names of the XML documents.

5.2 Stemming

Word stemming is a process to remove affixes (suffixes or prefixes) from the words and/or to replace them with the root word. For example, the word 'students' becomes 'student' and the word 'says' becomes 'say'. This research uses the most commonly used Porter stemming algorithm for affix removal [10].

5.3 Integer removal

Due to the huge size of the dataset and the very large number of unique terms in Wikipedia dataset it is essential to reduce the dimension of the dataset without any information loss. A careful analysis of the dataset revealed that there were a large number of integers and they did not contribute to the semantic of the documents and hence they were removed in the pre-processing step.

5.4 Shorter length words removal

Also based on the analysis of the dataset, words with length less than 4 are not meaningful, thus, they were removed. After the pre-processing of the extracted content, the content is represented as a term-document matrix which is called the Intermediate cluster form.

6 Phase 4: Clustering

The ICF matrix, generated from the previous phase, is fed to a partitional clustering algorithm such as the k-way clustering solution[11]. The k-way clustering algorithm groups the documents to the required number of clusters. The k-way clustering solution computes cluster by performing a sequence of *k-1* repeated bisections. In this approach, the matrix is first clustered into two groups, and then one of these groups is chosen and bisected further. This process of bisection continues until the desired number of bisections is reached. During each step of bisection, the cluster is bisected so that the resulting 2-way clustering solution locally optimizes a particular criterion function [11].

7 Experiments and Discussion

We implemented the algorithm using Microsoft Visual C++ 2005 and conducted experiments on the Wikipedia corpus from the INEX XML Mining Challenge 2008. The required numbers of clusters for INEX result submission was 15 clusters. The k-way clustering algorithm option in CLUTO[11] is applied on the ICF to cluster the documents.

The following table summarizes the results based on Micro F1 and Macro F1 measure evaluation metrics for 15 clusters. From the table 1 given below, it is very clear that our proposed method performs better than the structure-only approach by Hagenbuchner et al. Also there is not much loss in accuracy for our approach in comparison with other content only methods such as Tien et al and Chris De Vries. Hence, this method is suitable for combining the structure and content of XML documents without compromising on the accuracy.

Table 1. Submitted clustering results for INEX Wikipedia XML Mining Track 2008

Approaches	Number of clusters	Micro F1	Macro F1
Hagenbuchner et.al	15	0.26	0.38
Tien et. al	15	0.45	0.56
	30	0.53	0.57
Chris de Vries	15	0.49	0.59
Our approach	15	0.48	0.51
	30	0.53	0.58

In order to measure the reduction in the number of terms, a Vector Space Model was built on all the terms of the documents and clustering was then applied on it. From the experimental results summarised in table 2 it is evident that in spite of a drastic reduction in the number of unique terms and the total number of terms by about 50% and 40%, there is not any significant loss in accuracy and hence our method effectively combines the structure and content for clustering the documents and reduces the dimensionality.

Table 2. Dimensionality reduction

No. of Clus	Method	Micro-avg F1	Macro – avg F1	Num of Uniq. Terms	# Num Of Terms
	Our Approach	**0.53**	**0.58**	**442509**	**8528006**
30	On all the terms (without dimension reduction)	0.54	0.58	896050	13813559

8 Conclusion

In this paper, we have proposed and presented the results of our algorithm for mining both the structure and content of XML documents on INEX 2008 Wikipedia dataset. The main aim of this study is to explore and understand the importance of the content and structure of the XML documents for clustering task. In order to cluster the XML documents, we have used the content corresponding to the frequent subtrees in a given document and generated a term by document matrix. Using the matrix, we have computed the similarity between XML documents and clustered them based on their similarity values. We have demonstrated by including the structure we could not only reduce the dimensionality but also provide more meaningful clusters.

References

1. Nayak, R., R. Witt, and A. Tonev. *Data Mining and XML Documents*. in *International Conference on Internet Computing*. 2002.
2. Tran, T. and R. Nayak, *Evaluating the Performance of XML Document Clustering by Structure Only*, in *Comparative Evaluation of XML Information Retrieval Systems*. 2007. p. 473-484.
3. Kutty, S., R. Nayak, and Y. Li. *PCITMiner- Prefix-based Closed Induced Tree Miner for finding closed induced frequent subtrees*. in *Sixth Australasian Data Mining Conference (AusDM 2007)*. 2007. Gold Coast, Australia: ACS.
4. Nayak, R., *Investigating Semantic Measures in XML Clustering*, in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*. 2006, IEEE Computer Society. p. 1042-1045.
5. Aggarwal, C.C., et al., *Xproj: a framework for projected structural clustering of xml documents*, in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, ACM: San Jose, California, USA. p. 46-55.
6. Chi, Y., et al., *Frequent Subtree Mining- An Overview*, in *Fundamenta Informaticae*. 2005, IOS Press. p. 161-198.
7. Kutty, S., R. Nayak, and Y. Li, *XML Data Mining: Process and Applications*, in *Handbook of Research on Text and Web Mining Technologies*, M. Song and Y.-F. Wu, Editors. 2008, Idea Group Inc., USA.
8. Rijsbergen, C.J.v., *Butterworth.* Information Retrieval, 1979.
9. Fox, C., *A stop list for general text* ACM SIGIR Forum, 1989. **24**(1-2): p. 19-35.
10. Porter, M.F., *An algorithm for suffix stripping.* Program, 1980. **14**(3): p. 130-137.
11. Karypis, G. *CLUTO - Software for Clustering High-Dimensional Datasets | Karypis Lab*. 25 May 2007 .http://glaros.dtc.umn.edu/gkhome/views/cluto.

12. Hagenbuchner, M., et al. *Efficient clustering of structured documents using Graph Self-Organizing Maps*. in *Pre-proceedings of the Sixth Workshop of Initiative for the Evaluation of XML Retrieval*. 2007. Dagstuhl, Germany.

Utilizing the Structure and Data Information for XML Document Clustering

Tien Tran, Sangeetha Kutty, and Richi Nayak

Faculty of Science and Technology
Queensland University of Technology
GPO Box 2434, Brisbane QLD 4001, Australia
{t4.tran,s.kutty,r.nayak}@qut.edu.au

Abstract. This paper reports on the experiments and results of a clustering approach used in the INEX 2008 Document Mining Challenge. The clustering approach utilizes both the structure and the content information of the XML documents in the Wikipedia collection. The content of the XML documents is measured using the latent semantic kernel (LSK). A well-known problem with the construction of latent semantic kernel is the use of singular vector decomposition (SVD) method on a large feature space which is extremely expensive, in terms of computational as well as of memory requirements. Therefore a dimensional reduction method based on the common structural information of the XML documents is applied to reduce the dimension of the document space for building the latent semantic kernel. After the kernel is constructed, the proposed clustering approach uses the kernel to measure the similarity between each pair of document contents in the dataset. The proposed clustering approach has shown to be effective on the Wikipedia dataset.

Key words: XML document, clustering, content, structure, LSK, feature reduction

1 Introduction

Most electronic data on the web, nowadays, is presented in the format of semi-structured data including XML and HTML. Semi-structured data does not need to follow a specific structure and its data is nested and heterogeneous. As the continuous growth of the semi-structured data on the web, the need is unavoidable to efficiently manage these data. Along the line, data mining technique such as clustering has been widely used to group documents based on their common features such as their content without prior knowledge [1]. XML document clustering has become important in applications such as database indexing, data-warehouse, data integration and document engineering.

A number of clustering approaches have been proposed based on measuring the similarity according to content-only information [2, 3] and based on the structural similarity [4, 5]. An efficient XML document clustering method is yet to be developed that takes the structural and content information within documents

into consideration. Previous approaches [6, 7] utilize techniques such as vector space model [8] (VSM) to represent the content of the documents for clustering. However, the vector space model using tf-idf weighting [8] is associated with a number of limitations [9]. Firstly, it assigns weights to terms according to term frequency and ignores the semantic association of terms. Secondly, it does not perform well when both the structural and content features are used to infer the similarity of the XML documents [6].

To address the problems above, this paper proposes to use the latent semantic kernel (LSK) [10] to determine the similarity between the content of the documents for clustering. A shortcoming of LSK is its incapability to perform with large size datasets due to the requirement of large-size matrix computation. We propose using a dimensional document reduction method to reduce the dimensionality of the document space for LSK construction. Firstly, the XML dataset is grouped based on the document structural information. Using the structural groupings, the dimensional document reduction method is applied to reduce the document dimensional space of a term-document matrix for building the semantic kernel. The kernel is later used to group the Wikipedia dataset based on the content of XML documents.

This paper is structured as follows. The next section explains the clustering approach which has been used for INEX 2008 Document Mining Challenge in more detail. Section 3 evaluates the clustering approach with experiments and data analysis. The paper is then concluded in section 4.

2 The Clustering Approach

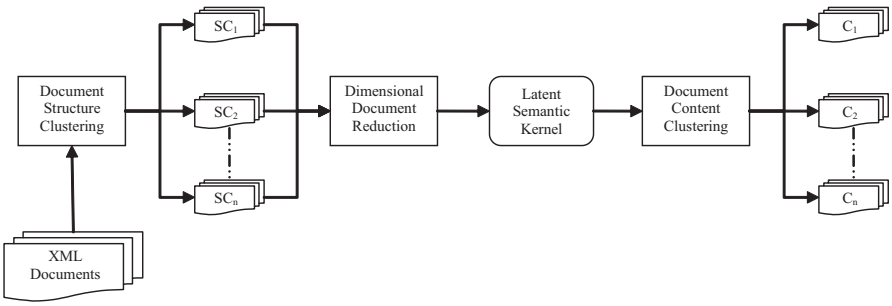


Fig. 1. Overview of the XML document clustering approach.

Fig. 1 outlines the overview of the XML document clustering approach which has been used for INEX 2008 Document Mining Challenge. The XML document dataset is first clustered based on the structural information present in the documents. The output of the clustering process is the groupings of the XML

documents according their common structural information. Documents within each group are used to reduce the dimensionality of the term-document matrix without having loss of terms using a dimensional document reduction method. The latent semantic kernel is constructed from this reduced size term-document matrix.

Given a collection of XML documents $\{d_1, d_2, ..., d_n\}$, an original term-document matrix, X, of $m \times n$ can be derived, where m stands for the number of unique terms and n stands for the number of documents in the collection. It is very computational expensive (and sometimes infeasible) to construct the LSK on this matrix. Based on the dimensional document reduction method, a modified term-document matrix, X_1, of $m \times n_1$ can be derived, where m stands for the number of unique terms and n_1 stands for the modified number of documents. When applying Singular Value Decomposition(SVD) on X_1, the matrix is decomposed into 3 matrices (equation 1), where U and V have orthonormal columns of left and right singular vectors respectively and S is a diagonal matrix of singular values ordered in decreasing magnitude.

$$X_1 = USV^T. \tag{1}$$

The SVD model can optimally approximate matrix X_1 with a smaller sample of matrices by selecting k largest singular values and setting the rest of the values to zero. Matrix U_k of size $m \times k$ and matrix V_k of size $n_1 \times k$ may be redefined along with $k \times k$ singular value matrix S_k (equation 2). This can approximate the matrix X_1 in a $k-$dimensional document space.

$$\hat{X_1}_{m \times n_1} = U_k S_k V_k^T. \tag{2}$$

Matrix $\hat{X_1}$ is known to be the matrix of rank k, which is closest in the least squares sense to X_1 [11]. Since U_k is the matrix containing the weighting of terms in a reduced dimensional space, it can be used as a kernel for learning the semantic between concepts. The U_k matrix becomes the semantic kernel that is used to group the Wikipedia dataset using the content of the documents.

Example. Let us take a collection D that contains 4 XML documents $\{d_1, d_2, d_3, d_4\}$, as shown in fig. 2; element names in the documents are shown as embraced within brackets, where $\langle R \rangle$ is the root element and $\langle E_i \rangle$ is the internal element or leaf element. The content of a document is denoted by T. Table 1 shows an example of term-document matrix X. Assume that these terms are extracted after the standard text preprocessing of stopword removal and stemming.

2.1 Structure Matching and Clustering

The first step in the clustering approach is to determine the structural commonality among the input XML documents. Given a collection of XML documents $\{d_1, d_2, ..., d_n\}$, denoted by D, a set of distinct paths $\{p_1, p_2, ..., p_y\}$, denoted by

Table 1. Example of an X matrix.

	d_1	d_2	d_3	d_4
t_1	2	1	2	2
t_2	2	2	2	0
t_3	2	2	2	0
t_4	2	2	1	4
t_5	2	0	2	0
t_6	1	0	0	0
t_7	2	2	2	1
t_8	0	1	0	1
t_9	1	1	1	0
t_{10}	0	1	0	0

d_1

$\langle R \rangle$
- $\langle E_1 \rangle\, t_1, t_2, t_3$
- $\langle E_2 \rangle\, t_4, t_3, t_6$
- $\langle E_3 \rangle\, t_5, t_4, t_7$
  - $\langle E_{3.1} \rangle\, t_5, t_2, t_1$
  - $\langle E_{3.2} \rangle\, t_7, t_9$

d_2

$\langle R \rangle$
- $\langle E_1 \rangle\, t_1, t_4$
- $\langle E_2 \rangle\, t_3, t_3$
- $\langle E_3 \rangle\, t_4, t_7$
  - $\langle E_{3.1} \rangle\, t_2, t_9$
  - $\langle E_{3.1} \rangle\, t_2, t_7, t_8, t_{10}$

d_3

$\langle R \rangle$
- $\langle E_1 \rangle\, t_1, t_2$
- $\langle E_2 \rangle\, t_3, t_3$
- $\langle E_3 \rangle\, t_5, t_4, t_7$
  - $\langle E_{3.1} \rangle\, t_5, t_2, t_1$
  - $\langle E_{3.2} \rangle\, t_7, t_9$

d_4

$\langle R \rangle$
- $\langle E_1 \rangle\, t_1, t_4$
- $\langle E_3 \rangle\, t_4, t_7$
- $\langle E_3 \rangle\, t_4, t_8$
- $\langle E_1 \rangle\, t_1, t_4$

Fig. 2. An Example of a collection D that contains 4 XML documents

P, are extracted from D.

Definition 1 (Paths). *A path, p_i, contains element names from the root element to the leaf element. The leaf element is an element that contains the textual content*

Definition 2 (Structure). *The structure of an XML document is a collection of paths $\{p_1, p_2, ..., p_f\}$.*

Definition 3 (Structure Modeling). *The structure of a document d_i is modelled as a vector $\{p_{i,1}, p_{i,2}, ..., p_{i,y}\}$ which contains the occurrences of paths, the distinct paths extracted from dataset D, in the document. The structure of all documents in a collection is modelled as a path-document matrix, $PD_{y \times n}$, where y is the number of paths in P and n is the number of documents in D.*

Example. Revisiting the example collection D in fig. 2, an example of path-document matrix, PD, is shown in table 2.

Table 2. Example of a PD matrix.

	d_1	d_2	d_3	d_4
R/E_1	1	1	1	2
R/E_2	1	1	1	0
$R/E_3/E_{3.1}$	1	2	1	0
$R/E_3/E_{3.2}$	1	0	1	0
R/E_3	1	1	1	2

A structural clustering solution, SC, is obtained by applying a k-partitioning clustering method [12] on the PD matrix. The number of clusters for the document structure is equalled to the number of clusters in the final clustering solution.

2.2 Dimensional Document Reduction

As discussed earlier that applying SVD on a large matrix X is expensive, thus, a dimensional document reduction method is used to reduce the dimension of the document space in X. The dimensional document reduction method uses the structural clustering solution, SC, obtained in the previous phase. Let the structural clustering solution be a collection of clusters $SC = \{SC_1, SC_2, ..., SC_i\}$, where (1) $SC_i = \{d_1, d_2, d_j\}$, (2) clusters are disjoint, i.e., a d_j can only exist in one SC_i and (3) $\#SC_i > \#SC_i + 1$, where $\#SC_i$ stands for the number of documents in cluster SC_i, i.e., clusters in SC are sorted in ascending order according to the number of documents that they contain. Clusters containing the smaller number of documents are processed before the larger clusters to reduce the document dimensional space. For each SC_i in SC, if the number of

documents $\#SC_i$ in the cluster SC_i is equalled to user-defined number of documents selected for each cluster, Ψ, then all the documents in cluster SC_i are selected for matrix X' construction. For each SC_i in SC, if $\#SC_i < \Psi$ then all the documents in the cluster SC_i are selected for matrix X' construction and Ψ is adjusted for the left over clusters so that the document dimensional space of matrix X' is equalled to user defined document dimensional space, n'. For all the clusters where $\#SC_i > \Psi$, document importance of each document in the cluster is calculated. The document importance (DI) of a document in a cluster is measured as:

$$DI(d_j) = \frac{\sum_{i=1}^{l} w_i(d_j)}{\sqrt{\sum_{i=1}^{l}(w_i(d_j))^2}}. \tag{3}$$

where l is the number of distinct terms extracted from d_j and w_i is the weight of a term t_i in d_j calculated as:

$$w(t_i)(d_j) = tf_i(d_j) \times idf_i \tag{4}$$

where tf_i is the ratio of the number of occurrences of a term t_i in document d_j to the number of occurrences of all terms in document d_j; and, idf_i is obtained by dividing the number of all documents in SC_i by the number of documents containing t_i, and then taking the logarithm of that quotient. A high weight is yielded by a high term frequency in a given document and a low document frequency of the term in the collection in SC_i.

Documents with higher DI value are selected for kernel construction. Content of the documents with lower DI values is merged into one document, thus, the content of the merged document becomes $d' = (d_1 \cup d_2 \cup d_{y_i - n - 1})$. The number of documents to be merged is equalled to the number of documents $\#SC_i$ in cluster SC_i minus the number of documents Ψ to be selected from each cluster lesser than 1. After performing dimensional document reduction on the structural clustering solution SC, a matrix X' of $m \times n'$ is generated. A kernel is then constructed using matrix X'. Since the kernel is for learning the semantic association between the terms of the documents, reducing the document dimension but keeping the original term dimension intact allows the kernel to retain the terms of the input dataset for better semantic association.

2.3 Content Extraction and Matching

The kernel constructed in the previous phase is now utilized in determining the similarity between the content of each pair of documents. Given a collection of XML documents D, a set of distinct terms $\{t_1, t_2, ..., t_m\}$, denoted by T, are extracted from D after the pre-processing of the Wikipedia document contents. The pre-processing of the document contents involves the removal of unimportant terms and word stemming. The removal of unimportant terms includes stop words which are terms considered not to be important such as 'the', 'of', 'and', etc. With extensive experiments, integers and terms with length lesser than 4

are not important, thus, they are also removed from the term set.

Definition 4 (Terms). *A term, t_i, is a keyword that appears in the text content of the leaf element in an XML document.*

Definition 5 (Content Modelling). *The content of a document is modelled as a vector $\{t_1, t_2, ..., t_m\}$ where m has the same term dimensionality feature as the semantic kernel P. The presence of a term in the document is denoted by the frequency of the term in the document.*

Definition 6 (Content Matching). *Given two vectors, d_x and d_y, the semantic similarity of the two document contents is measured as the cosine similarity using the kernel P, $\frac{d_x^T P P^T d_y}{|P^T d_x||P^T d_y|}$*

2.4 Document Clustering

Using definition 6 described in the previous section, a pair-wise document similarity matrix can be generated by computing the similarity between each pair of document contents. However due to the large number of the Wikipedia dataset, the generation of the pair-wise document similarity matrix was not possible with the given memory space. Thus in this paper, we propose using only the first 1000 best document similarities associated with each document for the grouping of the Wikipedia dataset. With a number experiments and analysis, it has shown that using only the first 1000 best document similarities associated with each document for the clustering of the Wikipedia dataset improves the accuracy of the clustering solution as all the outlier similarities are discarded from the similarity matrix. So, instead of having a pair-wise document similarity matrix of 114366 by 114366 (the number of documents in the Wikipedia dataset), we have a similarity matrix of 114366 by 1000. Using the similarity matrix of 114366 by 1000, a k-partitioning clustering method [12] is used to cluster the dataset. The clustering method performs by first divides the input dataset, using the similarity matrix of 114366 by 1000, into two groups, and then one of these two groups is chosen to be bisected further. The process is repeated until the number of bisections in the process equals the number of user-defined clusters.

3 Experiments and Discussion

The experiments were performed using the Wikipedia collection containing 114366 documents from INEX 2008 document mining challenge for the clustering task. After the pre-processing of the document contents which involved the removal of unimportant word and word stemming, 446294 terms were extracted from the Wikipedia collection and used for the construction of the latent semantic kernel. The document dimensional space was reduced down to 257 using the dimensional document reduction method then only $200k$ was selected for the dimension of the kernel.

The required number of clusters on the Wikipedia collection was 15 clusters. Two evaluation methods were used to evaluate the performance of the clustering solution in the INEX 2008 challenge: Micro F1 and Macro F1 measures. There were 4 participants in the clustering task including our approach. Many submissions were submitted by the

participants, table 3 only shows the best result from each participant on 15 clusters. Hagenbuchner et al. approach, assuming to be using the structure-only information for the grouping of the dataset, has the worse clustering solution. Chris de Vries approach slightly outperforms Kutty et al. and our approach but not that significantly. Overall our clustering approach performed really well on the Macro F1 measure.

Table 3. Comparing Clustering Results on Wikipedia Dataset with 15 Clusters

Approaches	**Micro F1**	**Macro F1**
Hagenbuchner et al.	0.26	0.38
Kutty et al.	0.48	0.51
Chris de Vries	0.49	0.59
Our Approach	**0.45**	**0.56**

If using the structure-only clustering solution, as shown in table 4, our result and Hagenbuchner et al. result do not vary that much in accuracy. However the structure-only clustering solutions are much lower than the content (or with structure) clustering solutions in table 3. This emphasizes the importance of the content when grouping the Wikipedia dataset.

Table 4. Comparing Structure Clustering Results on Wikipedia Dataset with 15 Clusters

Approaches	**Micro F1**	**Macro F1**
Hagenbuchner et al.	0.26	0.38
Our Approach	**0.29**	**0.33**

Based on a number of experiments and analysis, the following can be ascertained from our clustering approach:

- Our approach performs efficiently well even though the semantic kernel is constructed using the dimensional document reduction method where the term semantic associations of the document contents are not fully captured with a small document dimensional space.
- The similarity matrix of 114366 by 1000 for the grouping of the Wikipedia dataset produced a better clustering solution than using the pair-wise similarity matrix of 114366 by 1143366 since the outlier document similarities are discarded.
- Even though the structure-only solution produced a poor clustering solution, however it has been utilized in our approach for the dimensional document reduction method and has improved the accuracy of the clustering solution with the use of content.

4 Conclusion

In this paper, we have proposed using a clustering approach that utilizes both the structural and the content information for XML document clustering. First the structure is used to group the XML documents, then, a semantic kernel is built. The kernel

is then used to cluster the content of XML documents. The results obtained from the experiments shows that the clustering approach performs effectively on the Wikipedia dataset.

References

1. Han, J., Kamber, M.: Data Mining: Concepts and Techiques., San Diego, USA: Morgan Kaufmann (2001)
2. Kurgan, L., Swiercz, W., Cios, K.J.: Semantic mapping of xml tags using inductive machine learning. In: CIKM'2002, Virginia, USA (2002)
3. Shen, Y., Wang, B.: Clustering schemaless xml document. In: CoopIS'2003. (2003)
4. Nayak, R., Tran, T.: A progressive clustering algorithm to group the xml data by structural and semantic similarity. IJPRAI'2007 **21**(3) (2007) 1–23
5. Nayak, R., Xu, S.: Xcls: A fast and effective clustering algorithm for heterogenous xml documents. In: PAKDD'2006, Singapore (2006)
6. Doucet, A., Lehtonen, M.: Unsupervised classification of text-centric xml document collections. In: INEX'2006. (2006) 497–509
7. Yao, J., Zerida, N.: Rare patterns to improve path-based clustering. In: INEX'2007, Dagstuhl Castle, Germany (Dec 17-19, 2007)
8. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill Book Co., New York (1989)
9. Garcia, E.: The classic vector space model (2006)
10. Cristianini, N., Shawe-Taylor, J., Lodhi, H.: Latent semantic kernels. JJIS'2002 **18**(2) (2002)
11. Landauer, T.K., Foltz, P.W., Laham, D.: An introduction to latent semantic analysis. Discourse Processes (25) (1998) 259–284
12. Karypis, G.: Cluto - software for clustering high-dimensional datasets — karypis lab

Self Organizing Maps for the clustering of large sets of labeled graphs.

S. Zhang[1], M. Hagenbuchner[1], A.C. Tsoi[2], A. Sperduti[3]

[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 Padova, Padova, Italy. Email:sperduti@math.unipd.it

Abstract. Data mining on Web documents is one of the most challenging tasks in machine learning due to the size of the Web domain, the underlying (link) structures, and due to the fact that the data is commonly not labeled (the meaning of data is not known a-priori). This paper considers Self-Organizing Maps (SOM) as an approach to such application domain. The SOM approach is applied to a data mining tasks which was made available as part of a competition on mining XML formatted Web documents from the Wikipedia domain.
The SOM is a popular unsupervised machine learning method which allows the projection of high dimensional data onto a low dimensional display space. Relative recent works defined and studied models of Self-Organizing Maps for the treatment of graphs. The most recent of such works, called GraphSOM, allowed the encoding of undirected and cyclic graphs. Problems with stability and computational demand are the main disadvantage of this approach. This paper applies the GraphSOM model to a data mining exercise involving XML formatted Web documents. This is made possible through an extension to the GraphSOM model which substantially improves the stability of the model, and allows for a much accelerated training. This extension is based on a soft encoding of the information needed to represent the vertices of an input graph. Experimental results versus the original GraphSOM model demonstrate the advantages of the proposed extension in data mining applications which require the clustering of data.

1 Introduction

Self-Organizing Maps [1] are a well established machine learning method, and are popularly applied to tasks requiring the clustering of high dimensional vectorial data into groups, or the projection of high dimensional data vectors onto lower dimension display space. The popularity of the SOM is mainly due to the following reasons: (1) The computational complexity of the underlying algorithms is linear, and hence, SOMs can be applied to data mining tasks, (2) high dimensional data can be projected onto low-dimensional (e.g. 2-dimensional) space, and hence, data can be readily displayed on paper, a display, etc, and (3) SOMs perform a topology preserving projection, and hence, the topology of data in the data space is largely preserved on the display space.

This paper investigates the possibility of engaging SOM to data mining tasks involving graph structured data. More specifically, this paper will give an overview of extensions to the SOM algorithm which allow the processing of graphs. We also propose a modification to an existing method which improves stability and scalability of

the model so as to be able to apply the SOM to a clustering task involving $114,366$ XML formatted Wikipedia documents from the Web. This dataset has been made available as part of a competition on XML mining and XML clustering [2]. It is represented as a graph containing $114,366$ nodes (each document is represented as a node). The topology of the graph is defined by the hyperlink structure between documents. Textual content is available for each of the documents and can be used to label the nodes.

Most approaches to machine learning are capable of processing vectorial information, it is observed that many real world problems are more naturally presented as sequences or graphs. For example, in speech recognition, the data is generally made available as a temporal sequence. Often, the meaning of a given sequence is not known, and hence, unsupervised machine learning methods are needed to deal with this type of data. Kohonen suggests an extension of the SOM to allow the clustering of phonemes (sub-strings of audio signals) [1].

It can be stated that all but the most exotic learning problems are are more suitable represented by richer structures such as trees and graphs. To give an example, data in molecular chemistry are more appropriately represented as a graph where the nodes represent the atoms of a molecule, and the links between nodes represent atomic bindings. The nodes and links in such graphs may be labelled. For example, to add a description of the type of atom, or the strength of an atomic binding. Note that vectors, and sequences are special cases of graphs, and hence, any model capable of processing graphs can be expected to also be able to process vectors and sequences.

The traditional approach to the processing of graphs apply a pre-processing step in order to represent a graph structure in vectorial form such that the data can then be processed in a conventional way. However, such pre-processing may result in the removal of important structural relations between the atomic entities of a graph. Hence, it is advisable to design algorithms which can deal with graph structured data directly without requiring pre-processing.

Relatively recent developments allowed the processing of graph structured data by a SOM such that the projection can be made from a domain of graphs to a fixed-dimensional display space [3–6]. The pioneering work presented in [3] introduced a SOM for Structured Data (SOMSD) which allowed the processing of labeled directed ordered acyclic graph (more commonly regarded to as labeled trees) structured data by individually processing the atomic components of a graph structure (the nodes, the labels, and the links). The basic idea is to present to a SOM an input vector which is a representation of a node from a given graph, a numeric data label which may be attached to the node, and information about the mappings of the offsping of the node. A main difference to the traditional SOM is that the input vectors are dynamic (i.e. can change during training). This is because the mapping of the node's offsprings can change when the codebook vectors of the SOM are updated, and hence, the input vector to a corresponding node changes. In practice, it has been shown that such dynamics in the input space do not impose a stability problem to the training algorithm [3].

When processing a node, the SOM-SD requires knowledge of the mapping of the node's offsprings. This imposes a strict causal processing order by starting from leaf nodes (which feature no further offsprings), and ending at the root node (which has no further parent nodes or incoming links). In other words, the SOM-SD cannot encode cyclic graphs, and can not differentiate identical sub-graphs which occur in different contextual setting.

The shortcoming of the SOM-SD was addressed through the introduction of a Contextual SOMSD (CSOMSD) [5] which allows the inclusion of information about both, parent and children of a given node, and hence, allows the encoding of *contextual* information about nodes in a directed graph. A problem with [5] is that the improved ability to discriminate between identical substructures can create a substantially increased demand on mapping space, and hence, the computational demand can be prohibitive for large scale learning problems. A second problem is that the method can not process cyclic graphs as would be required for the given XML clustering task.

A subsequent expansion proposed in [6] allows the processing of cyclic or undirected graphs. The method in [6] is called the GraphSOM and was shown to be computationally more efficient than a CSOMSD in large scale learning problems. In this paper we will show that the training times of a GraphSOM remain too large for learning tasks such as the given one due to stability issues of the approach which must be countered using small learning rates. Hence, we improved the GraphSOM algorithm so as to improve the stability of the approach and to allow much faster training of the method.

This paper is structured as follows: Section 2 introduces to concepts of SOM training and its extensions to the graph domain. The experimental setting and experimental findings are presented in Section 3. Conclusions are drawn in Section 4.

2 Self-Organizing Maps for graphs

In general, Self-Organizing Maps perform a topology preserving mapping of high dimensional data through a projection to a low dimensional display space. For simplicity, this paper assumes the display space to be $2-$dimensional. The display space is formed by a set of prototype units which are arranged on a regular grid. There is one prototype unit associated with each element on the lattice. An input to a SOM is expected to be k-dimensional vectors, the prototype units must be of the same dimension. The elements of the prototype units are adjusted during training by (1) obtaining the prototype unit which matches a given input best, and (2) adjusting the elements of the winner units and all its neighbors. The two steps are referred to as the "competitive step" and "cooperative step" respectively. An algorithmic description can be given as follows:

Competitive step: One sample input vector $\mathbf{u}$ is randomly drawn from the input data set and its similarity to the codebook vectors is computed. When using the Euclidean distance measure, the winning neuron is obtained through:

$$r = \arg\min_i \|(\mathbf{u} - \mathbf{m}_i)^T \mathbf{\Lambda}\|, \tag{1}$$

where $\mathbf{m}_i$ refers to the $i-$th prototype unit, the superscript T denotes the transpose of a vector, and $\mathbf{\Lambda}$ is a $k \times k$ dimensional diagonal matrix. For the standard SOM, all diagonal elements of $\mathbf{\Lambda}$ are equal to 1.

Cooperative step: $\mathbf{m}_r$ itself as well as its topological neighbours are moved closer to the input vector in the input space. The magnitude of the attraction is governed by the learning rate α and by a neighborhood function $f(\Delta_{ir})$, where Δ_{ir} is the topological distance between $\mathbf{m}_r$ and $\mathbf{m}_i$. Here topological distance is used to described the distance between the neurons topologically. It is common to use the Euclidean

distance to measure the distance topologically. The updating algorithm is given by:

$$\Delta \mathbf{m}_i = \alpha(t) f(\Delta_{ir})(\mathbf{m}_i - \mathbf{u}), \tag{2}$$

where α is the learning rate decreasing to 0 with time t, $f(.)$ is a neighborhood function which controls the amount by which the codebooks are updated. Most commonly used neighborhood function is the Gaussian function:

$$f(\Delta_{ir}) = \exp\left(-\frac{\|\mathbf{l}_i - \mathbf{l}_r\|^2}{2\sigma(t)^2}\right), \tag{3}$$

where the spread σ is called neighborhood radius which decreases with time t, $\mathbf{l}_r$ and $\mathbf{l}_i$ are the coordinates of the winning neuron and the i-th neuron in the lattice respectively.

These two steps together constitute a single training step and they are repeated a given number of iterations. The number of iterations must be fixed prior to the commencement of the training process so that the rate of convergence in the neighborhood function, and the learning rate, can be calculated accordingly.

Note that this training procedure does not utilize any ground truth (target) information. In other words, the algorithm does not require the availability of targets for a given set of training data. This renders the algorithm useful to applications for which target information is not available. Note also that the computational complexity of this algorithm scales linear with the size of the training set, and hence, explains the suitability of the method to data mining tasks.

When processing graphs, an input vector $\mathbf{x}$ is formed for each node in a set of graphs through concatenation of a numerical data label $\mathbf{u}$ which may be associated with the node, and the *state* information about the node's offsprings or neighbors. The literature describes two possibilities of computing the state:

1. SOM-SD approach: The state of an offspring or neighbor can be the mapping of the offspring or the neigbor [3, 5]. In this case, the input vector for the j-th node is $\mathbf{x}_j = (\mathbf{u}_j, \mathbf{y}_{ch[j]})$, where $\mathbf{u}_j$ is a numerical data vector associated with the j-th node, $\mathbf{y}_{ch[i]}$ is the concatenated list of coordinates of the winning neuron of all the children of the j-th node. Since the size of vector $\mathbf{y}_{ch[i]}$ depends on the number of offsprings, and since the SOM training algorithm requires constant sized input vectors, padding with a default value is used for nodes with less than the maximum outdegree of any graph in the training set.

2. GraphSOM approach: The state of an offspring or neighbor can be the activation of the SOM when mapping all the node's neighbors or offsprings [6]. In this case, the input vector is formed through $\mathbf{x}_j = (\mathbf{u}_j, \mathbf{M}_{ne[j]})$, where $\mathbf{u}_j$ is defined as before, and, $\mathbf{M}_{ne[j]}$ is a m-dimensional vector containing the activation of the map $\mathbf{M}$ when presented with the neighbors of node j. An element M_i of the map is zero if none of the neighbors are mapped at the i-th neuron location, otherwise it is the number of neighbors that were mapped at that location. This latter approach produced fixed sized input vectors which do not require padding. Note also that the latter approach requires knowledge of the mappings of all of a node's neighbors. The availability of these mappings cannot be assured when dealing with undirected or cyclic graphs. This is overcome in [6] by utilizing the mappings from a previous time step. The approximation is valid since convergence is guaranteed. Hence, the GraphSOM can process undirected or cyclic graphs.

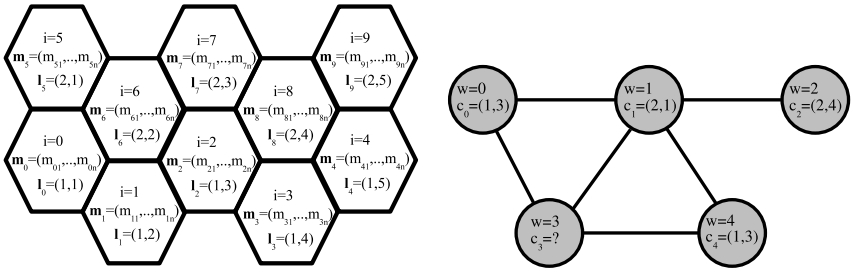


Fig. 1. A 2-dimensional map of size 5×2 (left), and an undirected graph (right). Each hexagon is a neuron. ID, codebook, and coordinate value for each neuron is shown. For each node, the node number, and coordinate of best matching codebook is shown.

It can be observed that the inclusion of state information in the input vector provides a local *view* of the graph structure. The iterative nature of the training algorithm ensures that local views are propagated through the nodes in a graph, and hence, structural information about the graph is passed on to all reachable nodes.

It can be seen that the concatenation of data label and state produces hybrid input vectors. The diagonal matrix $\mathbf{\Lambda}$ is used to control the influence of these two components on the mapping. The diagonal elements $\lambda_{11} \cdots \lambda_{pp}$ are set to $\mu \in (0; 1)$, all remaining diagonal elements are set to $1 - \mu$, where $p = |\mathbf{u}|$. Thus, the constant μ influences the contribution of the data label, and the state component to the Euclidean distance. Note that if $|\mathbf{u}| = |\mathbf{x}|$ and $\mu = 1$ then the algorithm reduces to Kohonen's basic SOM training algorithm.

After training a SOM on a set of training data it becomes then possible to produce a mapping for input data from the same problem domain but which may not necessarily be contained in the training dataset. The level of ability of a trained SOM to properly map unseen data (data which are not part of the training set) is commonly referred to as the *generalization* performance. The generalization performance is one of the most important performance measures. However, in this paper, rather than computing the generalization performance of the SOM, we will evaluate the performance on the basis of micro purity and macro purity. This is done in order to comply with guidelines set out by the INEX-XML mining competition.

The GraphSOM provides a mechanism for processing the given XML mining tasks. However, we discovered a stability problem with GraphSOM which we will describe by using an example. Consider the example shown in Figure 1. This figure shows a SOM of size 5×2, and an undirected graph containing 5 nodes. For simplicity, we assume that no data label is associated with any node in the graph. When processing node $w = 3$ with a GraphSOM, then the network input is the k-dimensional vector $\mathbf{x}_3 = (0, 0, 2, 0, 0, 1, 0, 0, 0, 0)$. This is because two of the neighbors of node 3 are mapped at the coordinate $(1, 3)$ which refers to the 2-nd neuron, and the third neighbour of node 3 is mapped at $(2, 1)$ which refers to the 5-th neuron. The algorithm proceeds with the execution of Eq. 1 and Eq. 2. Due to the weight changes in Eq. 2 it is possible that the mapping of neighbors $w = 0$, $w = 1$, and $w = 4$ change. Assume that there is a minor change in the mapping of node $w = 0$, for example, to the nearby location

$(1, 4)$. However, the Euclidean distance measure in Eq. 1 does not make a distinction as whether a mapping changed to a nearby location or to a far away location; the contribution to the Euclidean distance remains the same. This defects the very purpose to achieve topology preserving properties, and can cause alternating states to occur. To counter this behaviour it is necessary to either reduce the learning rate to a very small value (causing long training times due to an increased demand on the iterations), or to use a large value for μ (causing a neglect of structural information).

To counter this problem, and to account for the fact that changes in the mapping of nodes are most likely to a location near a previous winning location, this paper suggest to *soft code* the mappings of neighbors. The GraphSOM *hard codes* 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. Instead, we propose to code the likelihood of a mapping in a subsequent iteration with a probability value. We note that due to the effects of Eq. 2 it is most likely that the mapping of a node will be unchanged at the next iteration. But since all neurons are updated, and since neurons which are close to a winner neuron (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 than to a far away location. These likelihoods are directly influenced by the neighborhood function and its spread. Hence, we propose to incorporate the likelihood of a mapping in subsequent iterations as follows:

$$M_i = e^{-\frac{\|1_i - 1_r\|^2}{2*\sigma(t)^2}} \cdot \frac{1}{\sigma(t) * \sqrt{2 * \pi}},$$

where $\sigma(t)$ decreases with time t towards zero, all other quantities are as defined before. The computation is accumulative for all of the node's neighbors. Note that the term $\frac{1}{\sigma(t)*\sqrt{2*\pi}}$ 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. It will be observed in Section 3 that 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. For ease of reference, we will refer to the proposed approach as the probability mapping GraphSOM (PM-GraphSOM).

This approach produces state vector elements which are non-zero, as compared with the GraphSOM where the state vector can be sparse; this creates the opportunity for an optimization of the competitive step. Since the Euclidean distance is computed through element-wise computations on two vectors, and since we are only interested in finding the best matching codebook, hence, the computation of the Euclidean distance can be interrupted as soon as the partial sum exceeds a previously found minimum distance. This was found to be very effective in practice.

3 Experiments

The experiments were conducted on a set of XML formatted documents with the task to cluster the data. The dataset consists of $114,366$ Web documents from the Wikipedia

Table 1. Document Counts for different classes in training dataset

Class-ID	471	49	339	252	1530	1542	10049	380	897	4347	9430	1310	5266	323	1131
Size	2945	1474	915	866	789	696	679	639	637	592	405	294	264	128	114

domain, and contains hyperlink information. The dataset produces one graph consisting of $114,366$ nodes, each of which representing one document. These nodes are connected based on the hyperlinks between documents. There are a total of $636,187$ directed links between the documents in the dataset. The maximum number of out links (outdegree) from a document is 1527, and on average each document has 11 neighboring documents (either linked by or link to). We conducted two sets of experiments, one is by using link structure only, another is by combining both link structure and some document features.

10% of the documents are labeled to indicate the desired clustering result to be one of 15 classes. The labels are exclusively used for testing purposes, and are not used during training. The goal of the task is to associate each document with a cluster so that all documents within the same cluster are labeled alike. A closer look at the distribution of patterns amongst the pattern classes revealed that the dataset is heavily unbalanced. This is shown in Table1. For example, the table shows that the class with ID "471" is approximately 21 times the size of class with ID "1131". Since we engage an unsupervised machine learning scheme, we must not use class membership information in the pre-processing phase (i.e. to balance the dataset). We expect that this unbalance will affect the quality of the clustering by the GraphSOM and PMGraphSOM.

XML structure, and textual content was available for each of the documents. This allowed us to add a numerical datalabel to each of the nodes in a graph to represent some of the features of the corresponding document. We considered four types of document features:

XML tags:

- For each document, we extract the XML tags only.
- Compute N to be the number of unique XML tags contained in the dataset.
- Initialize an N-dimensional tag vector for each document and each element in the vector associates to a particular tag.
- For each document we update the tag vector by counting the occurrences of the tag within the document and assign the value of counts to the corresponding element.
- We divided documents into 16 groups, the first 15 groups are corresponding to 15 given classes, and all unlabeled documents are covered into the last group. For each unique tag, we count how many documents within each group contained it and build a 16xN matrix. For each row (each tag), we also compute the standard deviation of percentages among different groups.

  In order to reduce the dimension, we filter some tags according to following rules:

  1. Remove tags which are not contained in any labeled documents.
  2. Remove tags which are not contained in any unlabeled documents.
  3. Remove tags where the standard deviation is less than a threshold.
- Use PCA (Principal Component Analysis) [7] to reduce the dimension N to n by using first n principal components.
- Attach n-dimension tag-vector as the labels to each node.

There are totally 626 unique tags in this dataset, 83 of them exist in the labeled set. After filtering, there are 14 tags remaining in the tag vector. After the application of the PCA, we used the first three principal components for the label vector.

Document text: By following the same steps for XML tag analysis, we extract text contents of the documents and build text vector for each document. There are 567,304 unique words, and remaining 432,904 unique words after stemming. Since this dimension of text vector is too large to be analyzed by PCA software, we filter out some words by following similar rules defined for reducing dimension of tag vector. After PCA, the five most important principal components are included to the label vector.

Template names: Documents in the dataset are using different templates which are defined as parameters of the tag $< template >$. We extract names of templates used by the documents and conduct statistic analysis. The results show that template information are associated with the document classes. Some documents within same class share similar sets of templates. We keep first four dimension of template vector after PCA.

Others: Other than tags, there are some tag-related features such as outdegree and depth of tag-tree. Besides links between documents, there are also unknown links between document and other web resources. In order to analyze the importance of these features for clustering task, we conduct statistic analysis, the results show that there is no obvious indication on the relationship between these features and document classes.

Thus, in the final label vector, we combine 5-dimensional text information, 4-dimensional template information and 3-dimensional tag information (12 dimensions totally). In some experiments we used template information only by using PCA to reduce the vector to 10 dimensions.

The performance of the system will be measured in terms of Classification performance, clustering performance, macro purity, and micro purity. Macro purity, and micro purity are defines in the usual way. Classification and clustering performance are computed as follows:

Classification: After training, we filter out the neurons which are activated at least by one labeled document. For each neuron we find the largest number of documents with same class label and associate this class label to this neuron. For all nodes in the graph we re-compute the Euclidean distance only on these activated neurons. Re-map the node to the winning neuron and assign the label of winning neuron to this node. For each of the labeled documents, if its original attached label matches the new label given by the network we count it as a positive result. Then we could measure the classification performance by computing the percentage of the number of positive results out of the number of all labeled documents.

Clustering: This measure is to evaluate pure clustering performance. For all nodes in the graph, we compute the Euclidean distance on all neurons on the map and get the coordinates of the winning neurons. Then we applied K-means on these coordinates to perform clustering. By using different values for K, we could get different numbers of clusters. Within each cluster, we find which class of documents are majority and associate the class label to this cluster. For all labeled documents, if its original attached label matches the label of the cluster which contains this document we count it as a positive result. The cluster performance can be indicated by the percentage of the number of positive results out of the number of all labeled documents.

3.1 Results

The training of the PMGraphSOM allows the adjustment of a variety of training parameters such as the learning rate, network size, number of training iterations, the weight μ, and neighborhood radius $\sigma(0)$. The optimal value of these parameters are problem dependent, and are not known a-priory. Hence, trial and error was used to identify a suitable set of training parameters. The following presents the best result obtained at due time for the INEX-2008 mining challenge.

A general observation was that the training of PM-GraphSOMs of a useful size required anywhere between 13 hours and 27 hours whereas the training of a GraphSOM of similar size and with similar parameters required about 40 days (approx. one iteration per day). The time required to train the GraphSOM is unreasonable, and hence, training was interrupted, and the experiments focused solely on PMGraphSOM.

Here we present the maps which performed best in clustering or classification. Figure 2 shows the mapping of documents in the training set of trained map which performed best in classifying the data, whereas Figure 3 shows the mapping of nodes for a map that performed best in clustering the same data. It can be seen, that the map shown in Figure 2 utilizes the mapping space considerably better when compared to the map shown in Figure 3. However, there are no clear clusters formed in Figure 2. In contrast, the map shown in Figure 3 visibly produced clusters of data belonging to the same pattern class.

To extract the clusters from a trained SOM, we applying K-means clustering to the mapped data of a trained PMGraphSOM. By setting K to a constant value, we can extract exactly K clusters from a SOM. The overall clustering and classification performance of these two maps when setting K to be either 15 or 512 are shown in Table 4. In table Table 4, the result named "hagenbuchner-01" refers to the SOM shown in Figure 2, whereas the results named "hagenbuchner-02" and "hagenbuchner-03" refer to the SOM shown in Figure 3. All other results refer to competitor's approaches. There is no published material available about the competitor's approaches at the time of writing, and hence, we are unable to compare our approach with the approach taken by others. Nevertheless, it can be seen that the best of our models performs reasonably well in comparison.

We found that the main reason which held us back from producing better results were due to the unbalanced nature of the dataset. This is illustrated by Table 2 which presents the confusion matrix of the training set produced by the SOM in Figure 2 (best classification performance). The values on the diagonal are the number of documents correctly classified. The total number of documents on the diagonal is 8902 and confusion rate is 28%. In comparison, Table 3 shows the confusion matrix for the SOM shown in Figure 3 (best clustering performance). Here are totally 7140 documents on the diagonal and the confusion rate is 60%. It can be seen that in both confusion matrices the worst performing classes are the smallest classes, and hence, are a main contributor to the observed overall performance levels. In the former experiment, we used 12 dimension labels which combined text, template name and tags information of the documents while in the later we only attached 10 dimension labels of template information. The result shows that the combined features can better differential documents under different classes than using less feature. Since we can not use information about the class labels, and hence, the best approach to counter the issue may be to increase the size of the SOM. However, due to time requirements we omitted these experiments here.

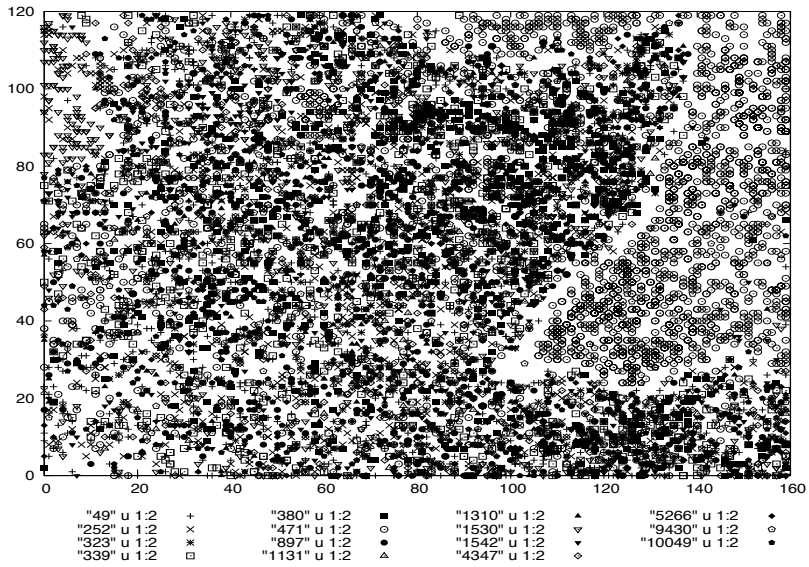


Fig. 2. Mapping of all labeled nodes on a PMGraphSOM performing best in classifying the nodes.

Table 2. Confusion Matrix generated by using parameters where: mapsize=160x120, iteration=50, grouping=20x20, $\sigma(0)$=10, μ=0.95, $\alpha(0)$=0.9, label: text=5, template=4, tag=3

9430	4347	1542	897	10049	49	252	323	471	1530	339	5266	380	1131	1310	%
253	5	5	20	20	34	12	1	15	8	8	5	13	0	6	62.4691
0	**400**	3	31	28	68	2	0	0	17	9	9	25	0	0	67.5676
0	16	**408**	38	32	73	23	0	19	15	25	8	30	1	8	58.6207
0	1	3	**554**	30	22	6	0	3	3	3	0	10	1	1	86.9702
0	0	1	5	**623**	26	4	0	2	5	8	0	5	0	0	91.7526
1	5	7	80	51	**1275**	5	0	5	10	15	0	17	0	3	86.4993
1	20	1	47	32	57	**622**	0	6	34	6	8	32	0	0	71.8245
0	5	2	7	13	11	9	**61**	5	1	5	4	5	0	0	47.6563
4	19	8	31	38	73	20	0	**2629**	33	34	12	36	0	8	89.2700
0	0	1	38	30	59	5	0	1	**610**	6	10	27	0	2	77.3131
0	17	4	46	36	98	35	0	3	30	**570**	14	50	1	11	62.2951
0	0	2	17	14	30	2	0	1	1	5	**175**	16	0	1	66.2879
0	3	1	29	26	65	6	0	4	5	10	1	**487**	0	2	76.2128
2	3	6	10	6	16	7	0	4	4	6	0	5	**45**	0	39.4737
0	4	0	22	17	33	4	0	0	8	4	0	12	0	**190**	64.6259

4 Conclusions

This paper presented an unsupervised machine learning approach to the clustering of a relatively large scale data mining tasks requiring the clustering of structured Web documents from the Wikipedia domain. It was shown that this can be achieved through

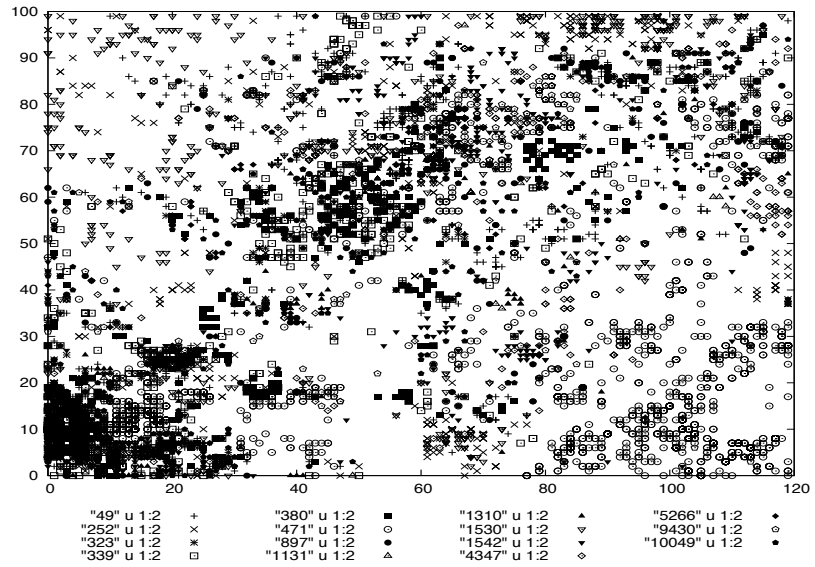


Fig. 3. Mapping of all labeled nodes on a PMGraphSOM performing best in clustering the nodes.

Table 3. Confusion Matrix generated by using parameters where: mapsize=120x100, iteration=50, grouping=10x10, $\sigma(0)$=10, μ=0.99994737, $\alpha(0)$=0.9, label: template=10.

9430	4347	1542	897	10049	49	252	323	471	1530	339	5266	380	1131	1310	%
144	29	20	15	11	87	18	1	50	7	9	0	8	0	6	35.5556
1	**312**	37	25	26	124	12	0	13	13	17	0	7	0	5	52.7027
3	29	**381**	25	25	157	13	0	19	6	19	0	10	1	8	54.7414
0	9	7	**245**	39	266	21	0	13	5	13	2	10	2	5	38.4615
0	5	3	11	**396**	168	29	0	10	7	24	0	23	0	3	58.3211
1	9	11	35	66	**1177**	37	1	28	26	42	2	30	0	9	79.8507
0	11	10	25	38	211	**445**	1	9	63	20	1	26	0	6	51.3857
1	1	1	5	18	63	0	**11**	4	3	13	1	7	0	0	8.5938
8	18	13	33	27	225	24	1	**2522**	19	23	2	16	0	14	85.6367
0	5	7	26	12	206	25	0	13	**475**	10	0	8	0	2	60.2028
3	22	10	35	48	228	40	1	16	10	**448**	1	32	0	21	48.9617
0	3	3	14	18	129	17	0	2	0	6	**60**	12	0	0	22.7273
0	2	3	22	36	171	8	0	10	2	10	1	**373**	0	1	58.3725
0	3	2	12	8	44	2	0	1	1	4	0	3	**34**	0	29.8246
1	14	13	11	14	84	8	0	9	1	11	1	10	0	**117**	39.7960

a suitable extension of an existing machine learning method based on the GraphSOM model. Experimental results produced a reasonably satisfying performance of of the approach, and revealed that the unbalanced nature of the training set are the main inhibitve factors to this task. A challenge has also been the encoding of textual information em-

Table 4. Results of the clustering task.

	Clusters	**MacroF1**	**MicroF1**	**Classification**	**Clustering**
hagenbuchner-01	15	0.2586290	0.377381	77.8351	40.1
hagenbuchner-02	512	0.5369338	0.464696	50.52024	47.6
hagenbuchner-03	512	0.5154928	0.470362	50.52024	48.8
QUT Freq_struct_30+links	30	0.551382	0.5440555	?	?
QUT collection_15	15	0.5051346	0.4879729	?	?
QUT Freq_struct_30	30	0.5854711	0.5388792	?	?
QUT LSK_7	30	0.5291029	0.5025662	?	?
QUT LSK_3	15	0.5307000	0.4927645	?	?
QUT LSK_1	15	0.5593612	0.4517649	?	?
QUT LSK_2	15	0.5201315	0.4441753	?	?
QUT LSK_8	30	0.5747859	0.5299867	?	?
QUT LSK_6	30	0.5690985	0.5261482	?	?
QUT LSK_4	15	0.4947789	0.4476466	?	?
QUT LSK_5	30	0.5158845	0.5008087	?	?
QUT Freq_struct_15	15	0.4938312	0.4833562	?	?
QUT collection_30	30	0.5766933	0.5369905	?	?
QUT Freq_struct_15+links	15	0.5158641	0.515699	?	?

bedded within the Wikipedia documents. This was resolved by using a Bag-of-Words approach in combination with Principal Component Analysis to extract and compress such type of information. Work on the proposed approach is ongoing with investigations into the effects of network size, and feature extraction on the clustering performance.

Acknowledgment: The authors wish to acknowledge financial support by the Australian Research Council through its Discovery Project grant DP0774168 (2007 - 2009) to support the research reported in this paper.

References

[1] Kohonen, T.: Self-Organisation and Associative Memory. 3rd edn. Springer (1990)
[2] Denoyer, L., Gallinari, P.: Initiative for the evaluation of xml retrieval, xml-mining track. http://www.inex.otago.ac.nz/ (2008)
[3] Hagenbuchner, M., Tsoi, A., Sperduti, A.: A supervised self-organising map for structured data. In Allison, N., Yin, H., Allison, L., Slack, J., eds.: WSOM 2001 - Advances in Self-Organising Maps. Springer (June 2001) 21–28
[4] Günter, S., Bunke, H.: Self-organizing map for clustering in the graph domain. Pattern Recognition Letters **23**(4) (2002) 405–417
[5] Hagenbuchner, M., Sperduti, A., Tsoi, A.: Contextual processing of graphs using self-organizing maps. In: European symposium on Artificial Neural Networks. Poster track, Bruges, Belgium (27 - 29 April 2005)
[6] Hagenbuchner, M., Tsoi, A.C., Sperduti, A., Kc, M.: Efficient clustering of structured documents using graph self-organizing maps. In: INEX. (2007) 207–221
[7] K., P.: On lines and planes of closest fit to systems of points in space. Philosophical Magazine **series 6**(2) (1901) 559–572

Author Index