INEX 2010 Workshop Pre-proceedings

Shlomo Geva, Jaap Kamps,
Ralf Schenkel, Andrew Trotman
(editors)

December 13–15, 2010
Huize Bergen, Vught, the Netherlands
http://www.inex.otago.ac.nz/

Published by: IR Publications, Amsterdam. ISBN 978-90-814485-4-3.

Preface

Welcome to the 9th workshop of the Initiative for the Evaluation of XML Retrieval (INEX)!

Now, in its ninth 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 2010 was an exciting year for INEX in which a number of new tracks started. In total nine research tracks were included, which studied different aspects of focused information access:

Ad hoc Track The main track of INEX 2010 will be investigating the effectiveness of XML-IR and Passage Retrieval for highly focused retrieval by restricting result length to "snippets" or discounting for reading effort, using Wikipedia as a corpus.

Book Track Investigating techniques to support users in reading, searching, and navigating full texts of digitized books.

Data Centric Track Investigating Focused Retrieval over a strongly structured collection of IMDb documents.

Interactive Track (iTrack) Investigating the behavior of users when interacting with XML documents, as well as developing retrieval approaches which are effective in user-based environments, working on large set of Amazon data (including formal descriptions and user-generated data).

Link-the-Wiki Track Investigating link discovery in the Te Ara encyclopedia.

Question Answering (QA@INEX) Track Investigating technology for accessing semi-structured data can be used to address real-world focused information needs formulated as natural language questions.

Relevance Feedback Track Investigate the utility of XML markup and passage retrieval in Relevance Feedback evaluation, with submission in the form of a executable computer program rather than a list of search result.

Web Service Discovery Investigate techniques for discovery of Web services based on searching service descriptions provided in WSDL.

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

The aim of the INEX 2010 workshop is to bring together researchers in the field of XML IR who participated in the INEX 2010 campaign. During the past year participating organizations contributed to the building of a large-scale XML test collection by creating topics, performing retrieval runs and providing relevance assessments. The workshop concludes the results of this large-scale effort, summarizes and addresses encountered issues and devises a work plan for the future evaluation of XML retrieval systems.

All INEX tracks start from having available suitable text collections. We gratefully acknowledge the data made available by: Amazon and LibraryThing (Interactive Track), New Zealand Ministry for Culture and Heritage (*Te Ara*, Link-the-Wiki Track), Microsoft Research (Book Track), the Internet Movie Database (Data Centric Track), and the Wikimedia Foundation (Adhoc, Relevance Feedback, and XML-Mining Track).

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 built by participants, and relevance judgments provided by participants. So the main thank you goes each of these individuals!

December 2010

Shlomo Geva
Jaap Kamps
Ralf Schenkel
Andrew Trotman

Organization

Steering Committee

Charlie Clarke (University of Waterloo)
Norbert Fuhr (University of Duisburg-Essen)
Shlomo Geva (Queensland University of Technology)
Jaap Kamps (University of Amsterdam)
Mounia Lalmas (University of Glasgow)
Stephen Robertson (Microsoft Research Cambridge)
Andrew Trotman (University of Otago)
Arjen P. de Vries (CWI)
Ellen Voorhees (NIST)

Chairs

Shlomo Geva (Queensland University of Technology)
Jaap Kamps (University of Amsterdam)
Ralf Schenkel (Max-Planck-Institut für Informatik)
Andrew Trotman (University of Otago)

Track Organizers

Ad Hoc

Paavo Arvola (University of Tampere)
Shlomo Geva (Queensland University of Technology)
Jaap Kamps (University of Amsterdam)
Ralf Schenkel (Max-Planck-Institut für Informatik)
Andrew Trotman (University of Otago)

Book

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

Data Centric

Qiuyue Wang (Renmin University of China)
Andrew Trotman (University of Otago)

Interactive (iTrack)

Thomas Beckers (University of Duisburg-Essen)
Norbert Fuhr (University of Duisburg-Essen)
Ragnar Nordlie (Oslo University College)
Nils Pharo (Oslo University College)

Link-the-Wiki

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

Question Answering (QA)

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

Relevance Feedback

Timothy Chappell (Queensland University of Technology)
Shlomo Geva (Queensland University of Technology)

Web Service Discovery

James Thom (RMIT University)
Chen Wu (Curtin University of Technology)

XML-Mining

Chris De Vries (Queensland University of Technology)
Sangeetha Kutty (Queensland University of Technology)
Richi Nayak (Queensland University of Technology)
Andrea Tagarelli (University of Calabria)

Table of Contents

Overview of the INEX 2010 Ad Hoc Track

Paavo Arvola[1] Shlomo Geva[2], Jaap Kamps[3],
Ralf Schenkel[4], Andrew Trotman[5], and Johanna Vainio[1]

[1] University of Tampere, Tampere, Finland
paavo.arvola@uta.fi, s.johanna.vainio@uta.fi
[2] Queensland University of Technology, Brisbane, Australia
s.geva@qut.edu.au
[3] University of Amsterdam, Amsterdam, The Netherlands
kamps@uva.nl
[4] Max-Planck-Institut für Informatik, Saarbrücken, Germany
schenkel@mpi-sb.mpg.de
[5] University of Otago, Dunedin, New Zealand
andrew@cs.otago.ac.nz

Abstract. This paper gives an overview of the INEX 2010 Ad Hoc Track. The main goals of the Ad Hoc Track were three-fold. The first goal was to study focused retrieval under resource restricted conditions such as a small screen mobile device or a document summary on a hit-list. The leads to variants of the focused retrieval tasks that address the impact of result length/reading effort, thinking of focused retrieval as a form of "snippet" retrieval. The second goal was to extend the ad hoc retrieval test collection on the INEX 2009 Wikipedia Collection with additional topics and judgments. For this reason the Ad Hoc track topics and assessments stayed unchanged. The third goal was to examine the trade-off between effectiveness and efficiency by continuing the Efficiency Track as a task in the Ad Hoc Track. The INEX 2010 Ad Hoc Track featured four tasks: the *Relevant in Context* Task, the *Restricted Relevant in Context* Task, the *Restrict Focused* Task, and the *Efficiency* Task. We discuss the setup of the track, and the results for the four tasks.

1 Introduction

The main novelty of the Ad Hoc Track at INEX 2010 is its focus on retrieval under resource restricted conditions such as a small screen mobile device or a document summary on a hit-list. Here, retrieval full articles is no option, and we need to find the best elements/passages that convey the relevant information in the Wikipedia pages. So one can view the retrieved elements/passages as extensive result snippets, or as an on-the-fly document summary, that allow searchers to directly jump to the relevant document parts.

There are three main research questions underlying the Ad Hoc Track. The first goal is to study focused retrieval under resource restricted conditions, thinking of focused retrieval as a form of "snippet" retrieval. The leads to variants of the focused retrieval tasks that address the impact of result length/reading effort, either by measures that factor in reading effort or by tasks that have restrictions on the length of results. The second goal is to extend the ad hoc retrieval test collection on the INEX 2009 Wikipedia Collection—four times the size, with longer articles, and additional semantic markup than the collection used at INEX 2006–2008—with additional topics and judgments. For this reason the Ad Hoc track topics and assessments stayed unchanged, and the test collections of INEX 2009 and 2010 can be combined to form a valuable resource for future research. The third goal is to examine the trade-off between effectiveness and efficiency by continuing the Efficiency Track as a task in the Ad Hoc Track. After running as a separate track for two years, the Efficiency Track was merged into the Ad Hoc Track for 2010. For this new Efficiency Task, participants were asked to report efficiency-oriented statistics for their Ad Hoc-style runs on the 2010 Ad Hoc topics, enabling a systematic study of efficiency-effectiveness trade-offs with the different systems.

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 since INEX 2007. Every XML element is, of course, also a passage of text. At INEX 2008, a simple passage retrieval format was introduced using file-offset-length (FOL) triplets, that allow for standard passage retrieval systems to work on content-only versions of the collection. That is, the offset and length are calculated over the text of the article, ignoring all mark-up. The evaluation measures are based directly on the highlighted passages, or arbitrary best-entry points, as identified by the assessors. As a result it is possible to fairly compare systems retrieving elements, ranges of elements, or arbitrary passages. These changes address earlier requests to liberalize the retrieval format to ranges of elements [3] and to arbitrary passages of text [10].

The INEX 2010 Ad Hoc Track featured four tasks:

1. The *Relevant in Context* Task asks for non-overlapping results (elements or passages) grouped by the article from which they came, but is now evaluated with an effort-based measure.
2. The *Restricted Relevant in Context* Task is a variant in which we restrict results to maximally 500 characters per article, directly simulating the requirements of resource bounded conditions such as small screen mobile devices or summaries in a hitlist.
3. The *Restrict Focused* Task asks for a ranked-list of non-overlapping results (elements or passages) when restricted to maximally 1,000 chars per topic, simulating the summarization of all information available in the Wikipedia.
4. The *Efficiency* Task asks for a ranked-list of results (elements or passages) by estimated relevance and varying length (top 15, 150, or 1,500 results per topic), enabling a systematic study of efficiency-effectiveness trade-offs with the different systems.

Note that the resulting test collection also supports the INEX Ad Hoc tasks from earlier years: *Thorough*, *Focused*, and *Best in Context*. We discuss the results for the four tasks, giving results for the top 10 participating groups and discussing their best scoring approaches in detail.

The rest of the paper is organized as follows. First, Section 2 describes the INEX 2010 ad hoc retrieval tasks and measures. Section 3 details the collection, topics, and assessments of the INEX 2010 Ad Hoc Track. In Section 4, we report the results for the Relevant in Context Task (Section 4.2); the Restricted in Context Task (Section 4.3); the Restricted Focused Task (Section 4.4); and the Efficiency Task (Section 4.5). Section 5 discusses the differences between the measures that factor in result length and reading effort, and the old measures that were based on precision and recall of highlighted text retrieval. Section 6 looks at the article retrieval aspects of the submissions, treating any article with highlighted text as relevant. Finally, in Section 7, we discuss our findings and draw some conclusions.

2 Ad Hoc Retrieval Track

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

2.1 Tasks

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.

At INEX 2010, the task is interpreted as a form of "snippet" retrieval, and the evaluation will factor in result length/reading effort.

Restricted Relevant in Context Task The scenario underlying *Restricted Relevant in Context* addresses the requirements of resource bounded conditions, such as small screen mobile devices or summaries in a hitlist, directly by imposing a limit of maximally 500 characters per article.

Restricted 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.

At INEX 2010, we interpret the task as a form of summarization of all information available in the Wikipedia, and restrict results to exactly 1,000 chars per topic.

Efficiency Task The efficiency task is different in its focus on the trade-off between effectiveness and efficiency. Specifically, participants should create runs with the top-15, top-150, and top-1500 results for the Thorough task, a system-oriented task that has been used for many years in the Ad Hoc Track. Additionally, participants reported runtimes and I/O costs for evaluating each query as well as general statistics about the hard- and software environment used for generating the runs.

The core system's task underlying most XML retrieval strategies is the ability to estimate the relevance of potentially retrievable elements or passages in the collection. Hence, the Thorough Task simply asks systems to return elements or passages ranked by their relevance to the topic of request. Since the retrieved results are meant for further processing (either by a dedicated interface, or by other tools) there are no display-related assumptions nor user-related assumptions underlying the task.

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 2010, we allowed the submission of three types of results: XML elements, file-offset-length (FOL) text passages, and ranges of XML elements. The submission format for all tasks is a variant of the familiar TREC format extended with two additional fields.

```
topic Q0 file rank rsv run_id column_7 column_8
```

Here:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.3 Evaluation Measures

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

The INEX 2010 measures are solely based on the retrieval of highlighted text. We simplify all INEX tasks to highlighted text retrieval and assume that systems will try to return all, and only, highlighted text. We then compare the characters of text retrieved by a search engine to the number and location of characters of text identified as relevant by the assessor. For the earlier Best in Context Task we used the distance between the best entry point in the run to that identified by an assessor.

Relevant in Context Task (INEX 2009) The evaluation of the Relevant in Context Task is based on the measures of generalized precision and recall [7] over articles, where the per document score reflects how well the retrieved text matches the relevant text in the document. Specifically, the per document score is the harmonic mean of precision and recall in terms of the fractions of retrieved and highlighted text in the document. We use an F_β score with $\beta = 1/4$ making precision four times as important as recall:

$$F_\beta = \frac{(1+\beta^2) \cdot \mathit{Precision} \cdot \mathit{Recall}}{(\beta^2 \cdot \mathit{Precision}) + \mathit{Recall}}.$$

We are most interested in overall performances, so the main measure is mean average generalized precision (MAgP). We also present the generalized precision scores at early ranks (5, 10, 25, 50).

Relevant in Context Task (INEX 2010) The INEX 2010 version of the Relevant in Context Task is as before, but viewed as a form of snippet retrieval, and uses a different per-document score that takes reading effort into account. Specifically, the per document score is the character precision at a tolerance to irrelevance (T2I) point. In this measure, the user is expected to read the returned passages in document order. When result passages are read, the user is expected to continue reading from the beginning of the document and read the remaining parts in document order. The reading stops when the user's tolerance to irrelevance (i.e. the amount of irrelevant characters) is met, or all characters of a document are read. In other words, the reading/browsing is expected to end when the user has bypassed 300 (default) irrelevant characters. The T2I(300) score per document is again used in the measure based on generalized precision and recall. We are most interested in overall performances so the main measure is mean average generalized precision (MAgP). We also present the generalized precision scores at early ranks (5, 10, 25, 50).

Restricted Relevant in Context Task The evaluation of the Restricted Relevant in Context Task is the same as of the (unrestricted) Relevant in Context Task using T2I(300). So the main performance measure is mean average generalized precision (MAgP) based on T2I(300). We also present the generalized precision scores at early ranks (5, 10, 25, 50).

Restricted Focused Task We are interested in giving a quick overview of the relevant information in the whole Wikipedia. This is a variant of the Focused Task where we restrict the results to exactly 1,000 characters per topic. Evaluation will be in terms of set-based precision over the retrieved characters (char_prec). In addition, we will report on the earlier Focused measures such as mean average interpolated precision (MAiP), calculated over over 101 standard recall points (0.00, 0.01, 0.02, ..., 1.00). We also present interpolated precision at early recall points (iP[0.00], iP[0.01], iP[0.05], and iP[0.10]),

Efficiency Task Precision is measured as the fraction of retrieved text that was highlighted. Recall is measured as the fraction of all highlighted text that has been retrieved. The Efficiency Task is evaluated as the INEX 2009 Thorough Task, which is basically identical to the Focused task. Since the Thorough Tasks allows for "overlapping" results, the evaluation will automatically discount text seen before in the ranked list. The notion of rank is relatively fluid for passages so we use an interpolated precision measure which calculates interpolated precision scores at selected recall levels. Since we are most interested in overall performance, the main measure is mean average interpolated precision (MAiP), calculated over over 101 standard recall points (0.00, 0.01, 0.02, ..., 1.00). We also present interpolated precision at early recall points (iP[0.00], iP[0.01], iP[0.05], and iP[0.10]),

For further details on the INEX measures, we refer to [1, 6].

3 Ad Hoc Test Collection

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

3.1 Corpus

Starting in 2009, INEX uses a new document collection based on the Wikipedia. The original Wiki syntax has been converted into XML, using both general tags of the layout structure (like *article*, *section*, *paragraph*, *title*, *list* and *item*), typographical tags (like *bold*, *emphatic*), and frequently occurring link-tags. The annotation is enhanced with semantic markup of articles and outgoing links, based on the semantic knowledge base YAGO, explicitly labeling more than 5,800 classes of entities like persons, movies, cities, and many more. For a more technical description of a preliminary version of this collection, see [9].

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

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

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

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

we can specifically ask about a group about Freddie Mercury:

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

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

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

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

3.2 Topics

The ad hoc topics were created by participants following precise instructions. Candidate topics contained a short CO (keyword) query, an optional structured CAS query, a phrase title, a one line description of the search request, and narrative with a details of the topic of request and the task context in which the information need arose. For candidate topics without a ⟨`castitle`⟩ field, a default

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

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

CAS-query was added based on the CO-query: `//*[about(., "`*CO-query*`")]`. Figure 2 presents an example of an ad hoc topic. Based on the submitted candidate topics, 107 topics were selected for use in the INEX 2010 Ad Hoc Track as topic numbers 2010001–2010107.

Each topic contains

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

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

```
<topic id="2010048" ct_no="371">
  <title>Pacific navigators Australia explorers</title>
  <castitle>
    //explorer[about(., Pacific navigators Australia explorers)]
  </castitle>
  <phrasetitle>"Pacific navigators" "Australia explorers"</phrasetitle>
  <description>
    Find the navigators and explorers in the Pacific sea in search of
    Australia
  </description>
  <narrative>
    I am doing an essay on the explorers who discovered or charted
    Australia. I am already aware of Tasman, Cook and La Prouse and
    would like to get the full list of navigators who contributed to
    the discovery of Australia. Those for who there are disputes about
    their actual discovery of (parts of) Australia are still
    acceptable. I am mainly interested by the captains of the ships
    but other people who were on board with those navigators still
    relevant (naturalists or others). I am not interested in those
    who came later to settle in Australia.
</narrative>
</topic>
```

Fig. 2. INEX 2010 Ad Hoc Track topic 2010048.

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

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

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

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

3.3 Judgments

Topics were assessed by participants following precise instructions. The assessors used the GPXrai assessment system that assists assessors in highlight relevant text. Topic assessors were asked to mark all, and only, relevant text in a pool of documents. After assessing an article with relevance, a separate best entry point decision was made by the assessor. All INEX 2010 tasks were evaluated against the text highlighted by the assessors, but the test collection does support the tasks of earlier years, such as the Thorough, Focused and Relevant in Context Tasks evaluated in terms of precision/recall, as well as the Best in Context Task evaluated against the best-entry-points.

The relevance judgments were frozen on November 3, 2010. At this time 52 topics had been fully assessed. Moreover, for 7 topics were is a second set of judgments by another assessor. All results in this paper refer to the 52 topics with the judgments of the first assigned assessor, which is typically the topic author.

– The 52 assessed topics were numbered 2010n with n: 003, 004, 006, 007, 010, 014, 016–021, 023, 025–027, 030–041, 043, 045–050, 054, 056, 057, 061, 068–070, 072, 075, 079, 095–097, 100, and 105–107.

In total 39,031 articles were judged. Relevant passages were found in 5,471 articles. The mean number of relevant articles per topic is 66, and the mean number of passages per topic is was 112.

Assessors where requested to provide a separate best entry point (BEP) judgment, for every article where they highlighted relevant text.

3.4 Questionnaires

At INEX 2010, as in earlier years, 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 1) featured 20 questions capturing contextual data on the search request. The post-assessment questionnaire (shown in Table 2) 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 2010 Ad Hoc Track.

4 Ad Hoc Retrieval Results

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

Table 1. Candidate Topic Questionnaire.

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

Table 2. Post Assessment Questionnaire.

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

Table 3. Participants in the Ad Hoc Track.

Id	Participant	Relevant in Context	Restricted Relevant in Context	Restricted Focused	Efficiency	CO query	CAS query	Phrase query	Reference run	Element results	Range of elements results	FOL results	# valid runs	# submitted runs
4	University of Otago	8	1	1	58	68	0	0	0	68	0	0	68	68
5	Queensland University of Technology	4	5	6	0	15	0	0	7	8	2	5	15	15
6	University of Amsterdam	2	2	2	0	6	0	0	0	0	0	6	6	6
9	University of Helsinki	0	0	4	0	4	0	0	0	0	0	4	4	8
22	ENSM-SE	4	0	0	0	4	0	4	2	4	0	0	4	4
25	Renmin University of China	2	0	0	0	2	0	0	0	2	0	0	2	2
29	INDIAN STATISTICAL INSTITUTE	2	2	3	3	10	0	0	1	3	0	7	10	12
55	Doshisha University	3	3	3	0	0	9	0	3	9	0	0	9	9
60	Saint Etienne University	1	0	0	0	1	0	0	1	1	0	0	1	2
62	RMIT University	2	0	0	0	2	0	0	0	2	0	0	2	2
65	Radboud University Nijmegen	1	1	3	0	4	1	0	3	0	0	5	5	9
68	University Pierre et Marie Curie - LIP6	0	0	3	3	6	0	0	2	6	0	0	6	6
72	University of Minnesota Duluth	1	1	1	0	0	3	0	0	3	0	0	3	0
78	University of Waterloo	1	1	1	0	3	0	0	0	0	0	3	3	3
98	LIA - University of Avignon	4	2	2	3	11	0	11	0	3	0	8	11	10
138	Kasetsart University	0	0	0	0	0	0	0	0	0	0	0	0	3
167	Peking University	12	9	2	17	40	0	0	0	40	0	0	40	45
557	Universitat Pompeu Fabra	0	0	3	0	3	0	0	1	0	0	3	3	9
	Total runs	47	27	34	84	179	13	15	20	149	2	41	192	213

4.1 Participation

A total of 213 runs were submitted by 18 participating groups. Table 3 lists the participants and the number of runs they submitted, also broken down over the tasks (Relevant in Context, Restricted Relevant in Context, Restricted Focused, or Efficiency); the used query (Content-Only or Content-And-Structure); whether it used the Phrase query or Reference run; and the used result type (Element, Range of elements, or FOL passage). Unfortunately, no less than 21 runs turned out to be invalid.

Participants were allowed to submit up to two element result-type runs per task and up to two passage result-type runs per task (for all four tasks). In addition, we allowed for an extra submission per task based on a reference run containing an article-level ranking using the BM25 model. For the efficiency task, we allowed sets of runs with 15, 150, 1,500 results per topic. The submissions

Table 4. Top 10 Participants in the Ad Hoc Track Relevant in Context Task (INEX 2010 T2I-score).

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p22-Emse303R	0.3752	0.3273	0.2343	0.1902	0.1977
p167-36p167	0.2974	0.2536	0.1921	0.1636	0.1615
p98-I10LIA1FTri	0.2734	0.2607	0.2067	0.1692	0.1588
p5-Reference	0.2736	0.2372	0.1800	0.1535	0.1521
p4-Reference	0.2684	0.2322	0.1714	0.1442	0.1436
p65-runRiCORef	0.2642	0.2310	0.1694	0.1431	0.1377
p25-ruc-2010-base2	0.2447	0.2198	0.1744	0.1359	0.1372
p62-RMIT10titleO	0.2743	0.2487	0.1880	0.1495	0.1335
p55-DUR10atcl	0.1917	0.1484	0.1163	0.0982	0.1014
p6-0	0.1798	0.1614	0.1314	0.1183	0.0695

are spread well over the ad hoc retrieval tasks with 47 submissions for Relevant in Context, 27 submissions for Restricted Relevant in Context, 34 for Restricted Focused, and 84 submissions for Efficiency.

4.2 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, factoring reading effort with T2I(300). The official measure for the task was mean average generalized precision (MAgP).

Table 4 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 3 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 three groups (based on MAgP).

ENSM-SE An element run, using the keyword (CO) query, the phrase title and the reference run.
Description: The method for scoring one document/element is based on the proximity of query terms in the document [2]. In this basic method, the influence of query terms is modelized by triangular functions. For the Run Emse303R, the height of the triangle was enlarged proportionnally to a weight learnt with the 2009 queries and assessments [4]. In the final run the elements and the documents are sorted with many keys. The first documents returned are those that appear both in our list and in the reference run, then documents from our list. For each document, elements are returned according to their score.

Peking University An element run, using the keyword (CO) query.

Description: Starting from a BM25 article retrieval run, then according to the semantic query model MAXimal Lowest Common Ancestor (MAXLCA), candidate element results are extracted. These elements are further ranked by BM25 and Distribution Measurements.

LIA – University of Avignon A FOL run, using the keyword (CO) query, and the phrase query.

Description: Based on advanced query expansion. We first retrieve the 10 top documents with a baseline query. The queries of this baseline are generated by combining the words from the ⟨`title`⟩ and ⟨`phrasetitle`⟩ fields of the topics. The documents are ranked with a language modeling approach and the probabilities are estimated using Dirichlet smoothing. We select the 50 most frequent unigrams, 20 most frequent 2-grams and 10 most frequent 3-grams from these 10 top-ranked documents, and we use them to expand the baseline query, allowing term insertions within the 2-grams and 3-grams. Finally, we retrieve the 1000 top documents with this expanded query and we get the file offset lengths corresponding to the first ¡section¿ field of each document.

Based on the information from these and other participants:

- The runs ranked ninth (*p55-DUR10atcl*) is using the CAS query. All other runs use only the CO query in the topic's title field.
- The first (*p22-Emse303R*), second (*p167-36p167*) and fourth (*p5-Reference*) run retrieve elements; the second (*p167-36p167*) and tenth (*p6-0*) run use FOL passages.
- Solid article ranking seems a prerequisite for good overall performance, with fifth (*p4-Reference*) through ninth (*p55-DUR10atcl*) runs retrieving only full articles.

4.3 Restricted Relevant in Context Task

We now discuss the results of the Restricted Relevant in Context Task in which we allow for only 500 characters per article to be retrieved. The Restricted Relevant in Context Task was also evaluated using generalized precision with the generalized score per article based on T2I(300). The official measure for the task was mean average generalized precision (MAgP).

Table 5 shows the top 10 participating groups (only the best run per group is shown) in the Restricted Relevant in Context Task. The first column lists the participant, see Table 3 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 three groups (based on MAgP).

Peking University Element retrieval run using the CO query.

Description: This is a variant of the run for the Relevant in Context task. That is, starting from a BM25 article retrieval run, then according to the

Table 5. Top 10 Participants in the Ad Hoc Track Restricted Relevant in Context Task (INEX 2010 T2I-score).

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p167-32p167	0.2910	0.2474	0.1872	0.1595	0.1580
p98-I10LIA2FTri	0.2631	0.2503	0.1972	0.1621	0.1541
p5-Reference	0.2722	0.2362	0.1785	0.1520	0.1508
p4-Reference	0.2684	0.2322	0.1714	0.1442	0.1436
p65-runReRiCORef	0.2641	0.2313	0.1686	0.1428	0.1375
p78-UWBOOKRRIC2010	0.1111	0.1001	0.0874	0.0671	0.0650
p55-DURR10atcl	0.1555	0.1300	0.1003	0.0822	0.0600
p6-categoryscore	0.1439	0.1191	0.1053	0.0980	0.0576
p29-ISI2010_rric_ro	0.1979	0.1673	0.1183	0.1008	0.0485
p72-1	0.0000	0.0000	0.0000	0.0000	0.0000

semantic query model MAXimal Lowest Common Ancestor (MAXLCA), candidate element results are extracted. These elements are further ranked by BM25 and Distribution Measurements. Here, the first 500 characters are returned for each element.

LIA – University of Avignon FOL passage retrieval using the CO query and phrases.
Description: Based on advanced query expansion. We first retrieve the 10 top documents with a baseline query. The queries of this baseline are generated by combining the words from the ⟨`title`⟩ and ⟨`phrasetitle`⟩ fields of the topics. The documents are ranked with a language modeling approach and the probabilities are estimated using Dirichlet smoothing. We select the 50 most frequent unigrams, 20 most frequent 2-grams and 10 most frequent 3-grams from these 10 top-ranked documents, and we use them to expand the baseline query, allowing term insertions within the 2-grams and 3-grams. Finally, we only select the 500 first characters of the first ⟨`section`⟩ field of each document (or less if the field contains less than 500 characters).

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

Based on the information from these and other participants:

- The best run (*p167-32p167*), the third run (*p5-Reference*), and the tenth run (*p72-1*) retrieve elements. The fourth run (*p4-Reference*), seventh run (*p55-DURR10atcl*), eighth run (*p6-categoryscore*) retrieve full articles, and the remaining four runs retrieve FOL passages.
- With the exception of the runs ranked seventh (*p55-DURR10atcl*) and tenth (*p72-1*), which used the CAS query, all the other best runs per group use the CO query.

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

Participant	char_prec	iP[.01]	iP[.05]	iP[.10]	MAiP
p68-LIP6-OWPCparentFo	0.4125	0.1012	0.0385	0.0000	0.0076
p55-DURF10SIXF*	0.3884	0.1822	0.0382	0.0000	0.0088
p9-yahRFT	0.3435	0.1186	0.0273	0.0000	0.0069
p98-LIAenertexTopic	0.3434	0.1500	0.0000	0.0000	0.0077
p167-40p167	0.3370	0.1105	0.0384	0.0000	0.0067
p65-runFocCORef	0.3361	0.0964	0.0435	0.0000	0.0067
p5-Reference	0.3199	0.1170	0.0431	0.0000	0.0070
p557-UPFpLM45co	0.3066	0.1129	0.0264	0.0000	0.0070
p4-Reference	0.3036	0.0951	0.0429	0.0000	0.0063
p29-ISI2010_rfcs_ref	0.2451	0.1528	0.0192	0.0000	0.0072

4.4 Restricted Focused Task

We now discuss the results of the Restricted Focused Task in which a ranked-list of non-overlapping results (elements or passages) was required, totalling maximally 1,000 characters per topic.

The official measure for the task was the set-based character precision over the 1,000 characters retrieved (runs were restricted or padded to retrieve exactly 1,000 characters if needed). Table 6 shows the best run of the top 10 participating groups. The first column gives the participant, see Table 3 for the full name of group. The second column gives the character-based precision over 1,000 characters retrieved, the third to fifth column give the interpolated precision at 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 three groups (based on official measure for the task, char_prec).

LIP6 An element retrieval run using the CO query.
Description: A learning to rank run that is retrieving elements for the CO queries (negated words are removed and words are not stemmed). We limit the domain of elements to the tag-types: {sec, ss, ss1, ss2, ss3, ss4, p}.

Doshisha University A manual element retrieval run, using the CAS query.
Description: We used the result reconstruction method from earlier years. In this method, we aim to extract more relevant fragments without irrelevant parts to return appropriate granular fragments as search results. We considered: 1) which granular fragments are more appropriate in overlapped fragments, and 2) what size is more suitable for search results. Our method combines neighbor relevant fragments to satisfy these views, by using the initial fragments obtained by a well-known scoring technique: BM25E as a basic scoring method for scoring each fragment, and ITF (inverse tag frequency) instead of IPF (inverse path frequency) because there are a number of tags in the test collection.

University of Helsinki A passage retrieval run using the CO query.

Table 7. Participants in the Ad Hoc Track Efficiency Task.

Participant	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
p167-18P167	0.4561	0.4432	0.4215	0.3936	0.2354
p4-OTAGO-2010-10topk-18	0.4425	0.4272	0.4033	0.3697	0.2304
p68-LIP6-OWPCRefRunTh	0.4790	0.4651	0.4343	0.3985	0.2196
p29-ISI2010_thorough.1500	0.2931	0.2930	0.2480	0.2145	0.0846
p98-I10LIA4FBas	0.5234	0.4215	0.2500	0.1677	0.0417

Description: The result list for each topic consists of a total of 1,000 characters from the beginning of the top two articles as ranked by the Yahoo! search-engine. Retrieving the passages from the beginning of the article is based on the assumption that the best entry point is in the beginning of the article. Because Yahoo! does not suggest any other entry point to the article, retrieving the beginning of the article is also what Yahoo! provides to users. Only the title field of the topic was used in the query.

Based on the information from these and other participants:

– Nine runs use the CO query. Only the second run (*p55-DURF10SIXF*) is a manual run using the CAS query.
– Only the ninth ranked system, (*p4-Reference*), retrieves full articles. The three runs ranked first (*p68-LIP6-OWPCparentFo*), second (*p55-DURF10SIXF**), and fifth (*p167-40p167*), and seventh (*p5-Reference*), retrieve elements. The remaining five runs retrieve FOL passages.

4.5 Efficiency Task

We now discuss the results of the Efficiency Task focusing on efficiency rather than effectiveness, and especially the trade-off between efficiency and effectiveness. Participants were asked to submit ranked-lists of 15 results, or 150 results, or 1,500 results per topic. The official measure for the task was mean average interpolated precision (MAiP). Table 7 shows the best run of the participating groups. The first column gives the participant, see Table 3 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 three groups (based on official measure for the task, MAiP).

Peking University An element retrieval run using the CO query.
Description: This is again a variant of the runs for (Restricted) Relevant in Context. That is, starting from a BM25 article retrieval run, then according to the semantic query model MAXimal Lowest Common Ancestor (MAXLCA), candidate element results are extracted. These elements are further ranked by BM25 and Distribution Measurements. Here, the parameters in ranking functions are tuned by a learning method.

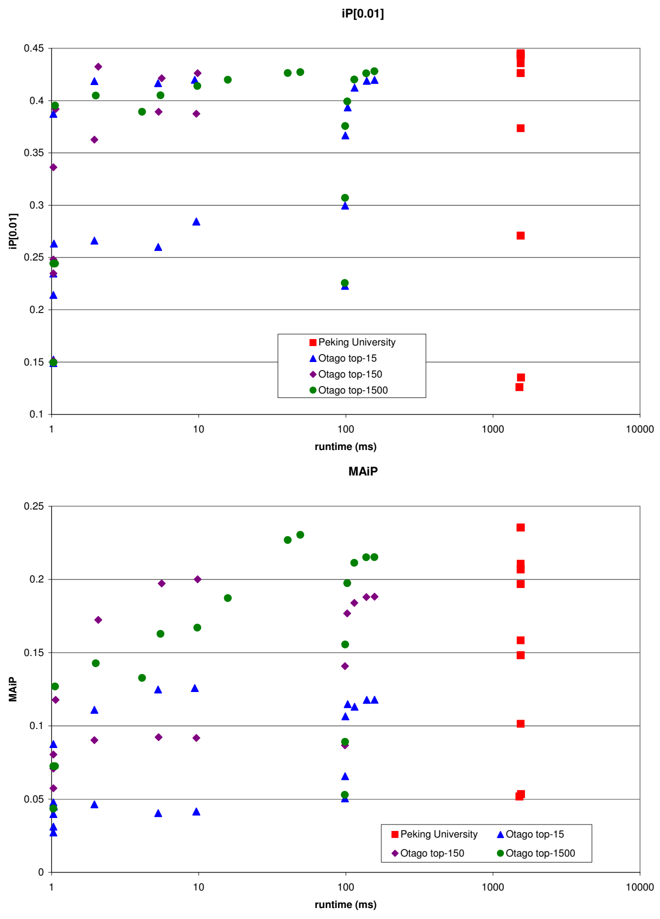


Fig. 3. Trade-off between Effectiveness and Efficiency: iP[0.01] (top) and MAiP (bottom).

University of Otago An article retrieval run using the CO query.

Description: The goal of the Otago runs was sub-millisecond per query. This was achieved using three techniques: impact ordered indexes, static pruning, and the use of a top-k ranking algorithm. Run p4-OTAGO-2010-10topk-18 scored the best in precision because it did the least pruning and least top-k restriction. It used BM25 and index-time S-stripper stemming. The fastest runs were, indeed, sub-millisecond, but at a reduced precision.

LIP6 An article retrieval run using the CO query.

Description: A learning to rank run that is retrieving top 1,500 documents for the CO queries (negated words are removed and words are not stemmed). For each document, the `/article[1]` element is retrieved.

Figure 3 shows the effectiveness, in terms of either iP[0.01] or MAiP, against the run-time efficiency. There is a vague diagonal trend—the best scoring runs

Table 8. Statistical significance (t-test, one-tailed, 95%).

(a) Relevant in Context Task

	1	2	3	4	5	6	7	8	9	10
p22		⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
p167			-	⋆	⋆	⋆	⋆	⋆	⋆	⋆
p98				-	-	⋆	⋆	⋆	⋆	⋆
p5					⋆	⋆	⋆	⋆	⋆	⋆
p4						⋆	-	-	⋆	⋆
p65							-	-	⋆	⋆
p25								-	⋆	⋆
p62									⋆	⋆
p55										-
p6										

(b) Restricted Relevant in Context Task

	1	2	3	4	5	6	7	8	9	10
p167		-	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
p98			-	-	-	⋆	⋆	⋆	⋆	⋆
p5				⋆	⋆	⋆	⋆	⋆	⋆	⋆
p4					⋆	⋆	⋆	⋆	⋆	⋆
p65						⋆	⋆	⋆	⋆	⋆
p78							-	-	⋆	⋆
p55								-	-	⋆
p6									-	⋆
p29										⋆
p72										

(c) Restricted Focused Task

	1	2	3	4	5	6	7	8	9	10
p68		-	-	-	-	-	⋆	⋆	⋆	⋆
p55			-	-	-	-	-	-	-	⋆
p9				-	-	-	-	-	-	⋆
p98					-	-	-	-	-	⋆
p167						-	-	-	-	⋆
p65							-	-	-	-
p5								-	-	-
p557									-	-
p4										-
p29										

(d) Efficiency Task

	1	2	3	4	5
p167		-	-	⋆	⋆
p4			-	⋆	⋆
p68				⋆	⋆
p29					⋆
p98					

tend to be the least efficient—but the trend is weak at best. Only the *University of Otago* submitted provided a large set of runs with all details. The MAiP scores tend to improve with longer runs, other things being equal this is no surprise. For the iP[0.01] scores, this is hardly the case.

Based on the information from these and other participants:

– The top scoring run (*p167-18P167*) uses elements, the fifth run (*p98-I10LIA4FBas*) uses FOL passages, and the other three runs retrieve articles.
– All runs use the CO query.

4.6 Significance Tests

We tested whether higher ranked systems were significantly better than lower ranked system, using a t-test (one-tailed) at 95%. Table 8 shows, for each task, whether it is significantly better (indicated by "⋆") than lower ranked runs. For the Relevant in Context Task, we see that the top run is significantly better than ranks 2 through 10. The second best run is significantly better than ranks 4 through 10. The third run better than ranks 6–10, the fourth run better than ranks 5-10, the fifth run better than runs 6 and 9–10, the sixth through eighth run better than runs 9–10. Of the 45 possible pairs of runs, there are 36 (or

Table 9. Top 10 Participants in the Ad Hoc Track Relevant in Context Task (INEX 2009 F-score).

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p22-Emse301R	0.3467	0.3034	0.2396	0.1928	0.1970
p167-21p167	0.3231	0.2729	0.2107	0.1767	0.1726
p4-Reference	0.3217	0.2715	0.2095	0.1751	0.1710
p25-ruc-2010-base2	0.2761	0.2627	0.2128	0.1686	0.1671
p65-runRiCORef	0.3190	0.2700	0.2078	0.1735	0.1623
p62-RMIT10title	0.2869	0.2585	0.1958	0.1573	0.1541
p98-I10LIA1FTri	0.2230	0.2048	0.1725	0.1421	0.1298
p55-DUR10atcl	0.2031	0.1663	0.1339	0.1096	0.1122
p29-ISI2010_ric_ro	0.2082	0.1874	0.1429	0.1250	0.0693
p5-Reference	0.0978	0.0879	0.0698	0.0640	0.0634

80%) significant differences, making MAgP a very discriminative measure. For the Restricted Relevant in Context Task, we see that the top run is significantly better than ranks 2 through 10. The second best run is significantly better than ranks 6 through 10. The third run better than ranks 4–10, the fourth run better than ranks 5–10, the fifth run better than runs 6–10, the sixth run better than 9–10, and the seventh through ninth run better than runs 10. Of the 45 possible pairs of runs, there are again 36 (or 80%) significant differences, confirming that MAgP is a very discriminative measure. For the Restricted Focused Task, we see that character precision at 1,000 characters is a rather unstable measure. The best run is significantly better than runs 7–10, and the runs ranked 2–5 and significantly better than the run ranked 10. Of the 45 possible pairs of runs, there are only 8 (or 18%) significant differences. Hence we should be careful when drawing conclusions based on the Focused Task results. For the Efficiency Task, we see that the performance (measured by MAiP) of the top scoring run is significantly better than the runs at rank 4 and 5. The same holds for the second and third best run. The fourth best run is significantly better than the run at rank 5. Of the 10 possible pairs of runs, there are 7 (or 70%) significant differences.

5 Analysis of Reading Effort

In this section, we will look in detail at the impact of the reading effort measures on the effectiveness of Ad Hoc Track submissions, by comparing them to the INEX 2009 measures based on precision and recall.

5.1 Relevant in Context

Table 9 shows the top 10 participating groups (only the best run per group is shown) in the Relevant in Context Task evaluated using the INEX 2009 measures based on a per article F-score. The first column lists the participant, see Table 3 for the full name of group. The second to fifth column list generalized precision at

Table 10. Top 10 Participants in the Ad Hoc Track Restricted Relevant in Context Task (INEX 2009 F-score).

Participant	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
p5-Reference	0.1815	0.1717	0.1368	0.1206	0.1064
p98-I10LIA2FTri	0.1639	0.1571	0.1340	0.1130	0.1053
p167-27p167	0.1622	0.1570	0.1217	0.1061	0.1030
p4-Reference	0.1521	0.1469	0.1119	0.0968	0.0953
p65-runReRiCORef	0.1610	0.1508	0.1138	0.0986	0.0945
p55-DURR10atcl	0.1369	0.1102	0.0870	0.0727	0.0537
p78-UWBOOKRRIC2010	0.0760	0.0777	0.0711	0.0544	0.0497
p6-0	0.0996	0.0880	0.0816	0.0782	0.0462
p29-ISI2010_rric_ro	0.1276	0.1189	0.0820	0.0759	0.0327
p72-1	0.0000	0.0000	0.0000	0.0000	0.0000

5, 10, 25, 50 retrieved articles. The sixth column lists mean average generalized precision.

Comparing Table 9 using the F-score and Table 4 using the T2I-score, we see some agreement. There are six runs in both tables, and some variant of the runs. There are however, notable upsets in the system rankings:

- Over all 47 Relevant in Context submissions, the system rank correlation is 0.488 between the F-score based and the T2I-score based evaluation.
- Taking the top 10 systems based on the T2I-score, their system ranks on the F-score have a correlation of 0.467.
- Taking the top 10 systems based on the F-score, their system ranks on the T2I-scores have a correlation of 0.956.

The overall system rank correlation is fairly low: the reading effort measure significantly affects the ranking. There is an interesting unbalance between the top 10 rankings. On the one hand, systems scoring well on the F-score tend to get very similar rankings based on the T2I-score. This makes sense since systems with a high F-score will tend to retrieve a lot of relevant text, and hence are to some degree immune to the T2I conditions. On the other hand, systems that score well on the T2I-score tend to have fairly different rankings based on the F-score. This can be explained by the high emphasis on precision of the T2I measures, and the relative importance of recall for the F-score.

Restricted Relevant in Context Table 10 shows the top 10 participating groups (only the best run per group is shown) in the Restricted Relevant in Context Task evaluated using the INEX 2009 measures based on a per article F-score. The first column lists the participant, see Table 3 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.

Comparing Table 10 using the F-score and Table 5 using the T2I-score, we see some agreement. There are eight runs in both tables, and some variant of the runs. There are however, notable upsets in the system rankings:

Table 11. Top 10 Participants in the Ad Hoc Track: Article retrieval.

Participant	P5	P10	1/rank	map	bpref
p22-Emse301R	0.6962	0.6423	0.8506	0.4294	0.4257
p167-38P167	0.7115	0.6173	0.8371	0.3909	0.3863
p25-ruc-2010-base2	0.6077	0.5846	0.7970	0.3885	0.3985
p98-I10LIA2FTri	0.6192	0.5827	0.7469	0.3845	0.3866
p4-Reference	0.6423	0.5750	0.7774	0.3805	0.3765
p5-Reference	0.6423	0.5750	0.7774	0.3805	0.3765
p62-RMIT10title	0.6346	0.5712	0.8087	0.3653	0.3683
p68-LIP6-OWPCRefRunTh	0.6115	0.5673	0.7765	0.3310	0.3480
p78-UWBOOKRRIC2010	0.5615	0.5115	0.7281	0.3237	0.3395
p65-runRiCORef	0.5808	0.5346	0.7529	0.3177	0.3382

– Over all 27 Restricted Relevant in Context submissions, the system rank correlation is 0.761 between the F-score based and the T2I-score based evaluation.
– Taking the top 10 systems based on the T2I-score, their system ranks on the F-score have a correlation of 0.022.
– Taking the top 10 systems based on the F-score, their system ranks on the T2I-scores have a correlation of 0.156.

The overall system rank correlation is higher than for the Relevant in Context task above, but the system rank correlations between the top 10's however are substantially lower.

6 Analysis of Article Retrieval

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

6.1 Article retrieval: Relevance Judgments

We will first look at the topics judged during INEX 2010, 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. 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 task requirements are now also considered, and we work with all 213 runs submitted to the Ad Hoc Track.

Table 11 shows the best run of the top 10 participating groups. The first column gives the participant, see Table 3 for the full name of group. The second and third column give the precision at ranks 5 and 10, respectively. The fourth column gives the mean reciprocal rank. The fifth column gives mean average precision. The sixth column gives binary preference measures (using the top R judged non-relevant documents). No less than five of the top 10 runs retrieved exclusively full articles: the three runs at rank one (*p22-Emse301R*), rank two (*p167-38P167*), and rank six (*p5-Reference*) retrieved elements proper, and the two runs at rank four (*p98-I10LIA2FTri*) and rank nine (*p78-UWBOOKRRIC2010*) retrieved FOL passages. The relative effectiveness of these article retrieval runs in terms of their article ranking is no surprise. Furthermore, we see submissions from all four ad hoc tasks. Runs from the Relevant in Context task at ranks 1, 3, 7; runs from the Restricted Relevant in Context task at ranks 4, 5, 9, 10; runs from the Restricted Focused task at ranks 6; and runs from the Efficiency task at rank 2, 8

If we break-down all runs over the original tasks, shown in Table 12, we can compare the ranking to Section 4 above. We see some runs that are familiar from the earlier tables: five Relevant in Context runs correspond to Table 4, seven Restricted in Context runs correspond to Table 5, seven Restricted Focused runs correspond to Table 6, and five Efficiency runs correspond to Table 7. More formally, we looked at how the two system rankings correlate using kendall's tau.

– Over all 47 Relevant in Context submissions the system rank correlation between MAgP and map is 0.674.
– Over all 27 Restricted Relevant in Context submissions the system rank correlation between MAgP and map is 0.647.
– Over all 34 Restricted Focused task submissions the system rank correlation is 0.134 between char_prec and map, and 0.194 between MAiP and map.
– Over all 84 Efficiency Task submissions the system rank correlation is 0.697 between MAiP and map.

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 only exception is the correlation between article retrieval and the Restricted Focused task. This is a likely effect of the evaluation over the bag of all retrieved text, regardless of the internal ranking.

7 Discussion and Conclusions

The Ad Hoc Track at INEX 2010 studied focused retrieval under resource restricted conditions such as a small screen mobile device or a document summary on a hit-list. Here, retrieval full articles is no option, and we need to find the best elements/passages that convey the relevant information in the Wikipedia pages. So one can view the retrieved elements/passages as extensive result snippets, or as an on-the-fly document summary, that allow searchers to directly jump to the relevant document parts.

Table 12. Top 10 Participants in the Ad Hoc Track: Article retrieval per task.

(a) Relevant in Context Task

Participant	P5	P10	1/rank	map	bpref
p22-Emse301R	0.6962	0.6423	0.8506	0.4294	0.4257
p25-ruc-2010-base2	0.6077	0.5846	0.7970	0.3885	0.3985
p98-I10LIA1ElTri	0.6192	0.5827	0.7469	0.3845	0.3866
p167-21p167	0.6423	0.5750	0.7774	0.3805	0.3765
p4-Reference	0.6423	0.5750	0.7774	0.3805	0.3765
p5-Reference	0.6423	0.5750	0.7774	0.3805	0.3765
p62-RMIT10title	0.6346	0.5712	0.8087	0.3653	0.3683
p78-UWBOOKRIC2010	0.5615	0.5115	0.7281	0.3237	0.3395
p65-runRiCORef	0.5808	0.5346	0.7529	0.3177	0.3382
p557-UPFpLM45co	0.5885	0.5423	0.7623	0.3041	0.3210

(b) Restricted Relevant in Context Task

Participant	P5	P10	1/rank	map	bpref
p98-I10LIA2FTri	0.6192	0.5827	0.7469	0.3845	0.3866
p4-Reference	0.6423	0.5750	0.7774	0.3805	0.3765
p167-29p167	0.6423	0.5750	0.7774	0.3805	0.3765
p5-Reference	0.6423	0.5750	0.7774	0.3805	0.3765
p78-UWBOOKRRIC2010	0.5615	0.5115	0.7281	0.3237	0.3395
p65-runReRiCORef	0.5808	0.5346	0.7529	0.3177	0.3382
p557-UPFsecLM45co	0.5846	0.5212	0.7904	0.2684	0.2919
p9-goo100RRIC	0.6423	0.5712	0.8830	0.2180	0.2503
p6-categoryscore	0.3115	0.2981	0.4319	0.1395	0.2566
p55-DURR10atcl	0.3269	0.2769	0.4465	0.1243	0.1540

(c) Restricted Focused Task

Participant	P5	P10	1/rank	map	bpref
p4-Reference	0.6423	0.5750	0.7774	0.3805	0.3765
p5-Reference	0.6423	0.5750	0.7774	0.3805	0.3765
p65-runFocCORef	0.5808	0.5346	0.7529	0.3177	0.3382
p98-LIAenertexDoc	0.5654	0.3192	0.7388	0.0636	0.0759
p55-DURF10SIXF*	0.4000	0.2442	0.7186	0.0531	0.0603
p557-UPFpLM45co	0.3769	0.2038	0.7308	0.0492	0.0531
p167-40p167	0.3038	0.1519	0.8462	0.0474	0.0484
p6-0	0.3154	0.3096	0.4230	0.0384	0.0591
p9-goo100RFT	0.3038	0.1519	0.8654	0.0382	0.0399
p29-ISI2010_rfcs_ref	0.2577	0.1308	0.5689	0.0300	0.0346

(d) Thorough Task

Participant	P5	P10	1/rank	map	bpref
p167-38P167	0.7115	0.6173	0.8371	0.3909	0.3863
p4-OTAGO-2010-10topk-18	0.6115	0.5654	0.7632	0.3738	0.3752
p98-I10LIA4FBas	0.6115	0.5673	0.7984	0.3648	0.3671
p68-LIP6-OWPCRefRunTh	0.6115	0.5673	0.7765	0.3310	0.3480
p29-ISI2010_thorough.1500	0.3731	0.2865	0.7294	0.0886	0.1804

In this paper we provided an overview of the INEX 2010 Ad Hoc Track that contained four tasks: The *Relevant in Context* Task asked for non-overlapping results (elements or passages) grouped by the article from which they came, but evaluated with an effort-based measure. The *Restricted Relevant in Context* Task is a variant in which we restricted results to maximally 500 characters per article, directly simulating the requirements of resource bounded conditions such as small screen mobile devices or summaries in a hitlist. The *Restrict Focused* Task asked for a ranked-list of non-overlapping results (elements or passages) restricted to maximally 1,000 chars per topic, simulating the summarization of all information available in the Wikipedia. The *Efficiency* Task asked for a ranked-list of results (elements or passages) by estimated relevance and varying length (top 15, 150, or 1,500 results per topic), enabling a systematic study of efficiency-effectiveness trade-offs with the different systems. We discussed the results for the four tasks.

The Ad Hoc Track had three main research questions. The first goal was to study focused retrieval under resource restricted conditions such as a small screen mobile device or a document summary on a hit-list. That is, to think of focused retrieval as a form of "snippet" retrieval. The leads to variants of the focused retrieval tasks that address the impact of result length/reading effort, either by measures that factor in reading effort or by tasks that have restrictions on the length of results.

The results of the effort based measures are a welcome addition to the earlier recall/precision measures. It addresses the counter-intuitive effectiveness of article-level retrieval—given that ensuring good recall is much easier than ensuring good precision [5]. As a result there are significant shifts in the effectiveness of systems that attempt to pinpoint the exact relevant text, and are effective enough at it. Having said that, even here locating the right articles remains a prerequisite for obtaining good performance, and finding a set of measures that resonate closely with the perception of the searchers remains an ongoing quest in focused retrieval.

The second goal was to extend the ad hoc retrieval test collection on the INEX 2009 Wikipedia Collection—four times the size, with longer articles, and additional semantic markup—with additional topics and judgments. For this reason the Ad Hoc track topics and assessments stayed unchanged, and the test collections of INEX 2009 and 2010 combined form a valuable resource for future research.

INEX 2010 added 52 topics to the test collection on the INEX Wikipedia Corpus, making it a total of 110 topics. In addition there are seven double judged topics. This results in an impressive test collection, with a large topic set and highly complete judgments [8]. There are many ways of (re)using the resulting test collection for passage retrieval, XML element retrieval, or article retrieval, but also to piggy-back other retrieval tasks on top of the available topics and judgments.

The third goal was to examine the trade-off between effectiveness and efficiency by continuing the Efficiency Track as a task in the Ad Hoc Track. After running as a separate track for two years, the Efficiency Track was merged into the Ad Hoc Track for 2010. For this new Efficiency Task, participants were asked to report efficiency-oriented statistics for their Ad Hoc-style runs on the 2010 Ad Hoc topics, enabling a systematic study of efficiency-effectiveness trade-offs with the different systems.

The Efficiency task received more runs than at INEX 2009 but of a smaller number of participants. Regarding efficiency, average running times per topic varied from 1ms to 1.5 seconds, where the fastest runs where run on indexes kept in memory. This is again almost an order of magnitude faster than the fastest system from INEX 2009, and the low absolute response times clearly demonstrate that the current Wikipedia-based collection is not large enough to be a true challenge for current systems. Result quality was comparable to other runs submitted to other tasks in the AdHoc Track.

For all main research questions, we hope and expect that the resulting test collection will prove its value in future use. After all, the main aim of the INEX initiative is to create bench-mark test-collections for the evaluation of structured retrieval approaches.

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

Bibliography

[1] P. Arvola, J. Kekäläinen, and M. Junkkari. Expected reading effort in focused retrieval evaluation. *Information Retrieval*, 13:460–484, 2010.

[2] M. Beigbeder. Focused retrieval with proximity scoring. In *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC'10)*, pages 1755–1759. ACM Press, New York NY, USA, 2010.

[3] 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.

[4] M. Géry, C. Largeron, and F. Thollard. Integrating structure in the probabilistic model for information retrieval. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 763–769. IEEE Computer Society, 2008.

[5] J. Kamps, M. Koolen, and M. Lalmas. Locating relevant text within XML documents. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 847–849. ACM Press, New York NY, USA, 2008.

[6] J. Kamps, J. Pehcevski, G. Kazai, M. Lalmas, and S. Robertson. INEX 2007 evaluation measures. In *Focused access to XML documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2007)*, volume 4862 of *Lecture Notes in Computer Science*, pages 24–33. Springer Verlag, Heidelberg, 2008.

[7] J. Kekäläinen and K. Järvelin. Using graded relevance assessments in IR evaluation. *Journal of the American Society for Information Science and Technology*, 53:1120–1129, 2002.

[8] S. Pal, M. Mitra, and J. Kamps. Evaluation effort, reliability and reusability in XML retrieval. *Journal of the American Society for Information Science and Technology*, 2011.

[9] R. Schenkel, F. M. Suchanek, and G. Kasneci. YAWN: A semantically annotated Wikipedia XML corpus. In *12. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web (BTW 2007)*, pages 277–291, 2007.

[10] 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	1019	Reference	RiC	CO	Ele	Article-only
4	1020	Reference	RRiC	CO	Ele	Article-only
4	1021	Reference	RFoc	CO	Ele	Article-only
4	1138	OTAGO-2010-10topk-18	Eff	CO	Ele	Article-only
5	1205	Reference	RiC	CO	Ele	Reference run
5	1206	Reference	RRiC	CO	Ele	Reference run
5	1207	Reference	RFoc	CO	Ele	Reference run
5	1208	Reference	RiC	CO	Ran	Reference run Invalid
5	1212	Reference	RRiC	CO	Ele	Reference run
5	1213	Reference	RFoc	CO	Ele	Reference run
6	1261	0	RiC	CO	FOL	
6	1265	categoryscore	RRiC	CO	FOL	Article-only
6	1266	0	RRiC	CO	FOL	
6	1268	0	RFoc	CO	FOL	
9	1287	goo100RRIC	RRiC	CO	FOL	Invalid
9	1294	goo100RFT	RFoc	CO	FOL	
9	1295	yahRFT	RFoc	CO	FOL	
22	1249	Emse301R	RiC	CO	Ele	Phrases Reference run
22	1251	Emse303R	RiC	CO	Ele	Phrases Reference run
25	1282	ruc-2010-base2	RiC	CO	Ele	Article-only
29	1067	ISI2010_thorough.1500	Eff	CO	Ele	Article-only
29	1073	ISI2010_rric_ro	RRiC	CO	FOL	
29	1094	ISI2010_ric_ro	RiC	CO	FOL	
29	1096	ISI2010_ref_ric_aggr	RiC	CO	FOL	Reference run Invalid
29	1098	ISI2010_rfcs_ref	RFoc	CO	FOL	Reference run
55	1163	DUR10atcl	RiC	CAS	Ele	Reference run Article-only
55	1164	DURF10SIXF	RFoc	CAS	Ele	Manual
55	1169	DURR10atcl	RRiC	CAS	Ele	Reference run Article-only
60	1289	UJM_33456	RiC	CO	Ele	Reference run
62	1290	RMIT10title	RiC	CO	Ele	Article-only
62	1291	RMIT10titleO	RiC	CO	Ele	Article-only
65	1273	runRiCORef	RiC	CO	FOL	Reference run Article-only
65	1274	runReRiCORef	RRiC	CO	FOL	Reference run
65	1275	runFocCORef	RFoc	CO	FOL	Reference run
68	1170	LIP6-OWPCparentFo	RFoc	CO	Ele	
68	1181	LIP6-OWPCRefRunTh	Eff	CO	Ele	Reference run Article-only
72	1031	1	RRiC	CAS	Ele	
78	1024	UWBOOKRIC2010	RiC	CO	FOL	
78	1025	UWBOOKRRIC2010	RRiC	CO	FOL	
98	1255	I10LIA4FBas	Eff	CO	FOL	Phrases
98	1258	I10LIA1ElTri	RiC	CO	Ele	Phrases
98	1260	I10LIA1FTri	RiC	CO	FOL	Phrases
98	1270	I10LIA2FTri	RRiC	CO	FOL	Phrases
98	1284	LIAenertexTopic	RFoc	CO	FOL	Phrases
98	1285	LIAenertexDoc	RFoc	CO	FOL	Phrases

Continued on Next Page. . .

Group	Run	Label	Task	Query	Results	Notes
167	1049	21p167	RiC	CO	Ele	
167	1076	32p167	RRiC	CO	Ele	
167	1079	29p167	RRiC	CO	Ele	
167	1081	27p167	RRiC	CO	Ele	
167	1092	36p167	RiC	CO	Ele	
167	1219	40p167	RFoc	CO	Ele	
167	1241	18P167	Eff	CO	Ele	
167	1242	38P167	Eff	CO	Ele	
557	1313	UPFpLM45co	RiC	CO	FOL	Reference run Invalid
557	1316	UPFsecLM45co	RRiC	CO	FOL	Reference run Invalid
557	1319	UPFpLM45co	RFoc	CO	FOL	Reference run

When is in-Context Retrieval Beneficial?

Paavo Arvola, Johanna Vainio
University of Tampere, Dept. of Inf. Studies and Interactive Media
33014 University of Tampere, Finland
{paavo.arvola, s.johanna.vainio}@uta.fi

Abstract. In this study, the Relevant-in-Context retrieval task is explored in the light of traditional full document retrieval. Obviously, under some circumstances the full document retrieval is sufficient in finding relevant material effectively. Namely, the Relevant-in-Context retrieval does not bring any improvements in case the retrieved documents are thoroughly (i.e. densely) relevant, or they start with relevant material. By using the INEX data, we perform a topic-wise analysis focusing on these qualities of the retrieved relevant documents. In addition, we evaluate the submitted INEX runs with the localizing effort metric, in order to study what are the actual system performances in locating the relevant material within a document.

Keywords: Relevant-in-Context, evaluation, metrics, relevance density

1 Introduction

This study seeks justification for the in-Context retrieval tasks, especially the Relevant-in-Context (RiC) task. The RiC retrieval task can be considered as focused retrieval enhanced document retrieval. In it, the retrievable unit is a document having the best matching passages highlighted. This kind of grouping the result passages by their document is called *fetch and browse* retrieval [5] and the related RiC task is considered as the most credible task of all tasks in the ad hoc track [9]. An information retrieval system aims to comprise the following tasks within the fetch and browse retrieval:

1. In the fetch phase to rank the documents in decreasing order of relevance.
2. In the browse phase to identify the relevant passages in the retrieved sparsely relevant documents.

The fetch phase is a task for a document retrieval system, whereas the browse phase is a task for a focused retrieval system. The optimal document ranking is sometimes considered to be based on the exhaustivity and specificity dimensions of relevance [7]. Currently, in INEX, the exhaustivity dimension is binary and the specificity dimension is approximated as the density of relevance i.e. the amount of relevant text divided by total text within a retrievable unit [4, 8].

Aiming to rank the documents based on their relevance density is problematic in the perspective of focused retrieval. Namely, if the relevance density of a document is high, there is no need for a focused retrieval system to identify the relevant passages, because (nearly) everything within the document is relevant. Instead, with sparsely relevant documents, the focused retrieval may be beneficial. Another defeat for the focused retrieval approach is the case when the relevant content occurs at the beginning of a document. Typically the document start is offered for the user by default and there is often no need to guide the user elsewhere.

The aim of this paper is to study the potential and actual benefit of the RiC task by using the INEX data. We perform a topic-wise analysis based on relevance density and the location of the relevant text measured as the distance of the first relevant passage from the document start in the retrieved relevant documents. In addition, we evaluate the submitted INEX runs with the localizing effort metric, in order to study what are the actual system performances in locating the relevant material within a document.

2 Analysis Based on Three Document Retrieval Scenarios

Focused retrieval can be beneficial in locating relevant material from a document. This occurs especially, when the relevance density of the result document is low and the relevant material doesn't occur at the beginning of the document. In this section, we compare three different *document retrieval* scenarios using the INEX 2010 recall base and some of the top performing runs. The aim of this is to study, whether focused retrieval is a justified approach in reducing user's effort in general in terms of relevance density and the starting point of the relevant text within the document. We examine each topic with the following document retrieval scenarios:

1) optimal
2) average
3) realistic.

The optimal scenario refers to a case, where a document retrieval system is capable of delivering the documents in the best possible order. In terms of density this means that the documents are ranked in descending order by their density. It is worth noting that the optimal scenario is actually the worst scenario from the perspective of focused retrieval. The average scenario is the average of the relevant documents and the realistic scenario is based on the fetch phase of some best performing runs of INEX 2010 (Emse303R, 32p167, I10LIA1FTri, Reference/qtau).

In Figure 1 there is a topic-wise comparison between the scenarios based on relevance density. In order to emphasize precision in the optimal and realistic strategies, only the top five relevant documents are considered and the average is reported among them. That means we discard the possible non-relevant documents in between focusing on the top five *relevant* documents in the realistic scenario, because the non-relevant documents are not interesting. The average densities over all topics are 0.71, 0.34 and 0.42, for optimal, average and realistic scenarios respectively.

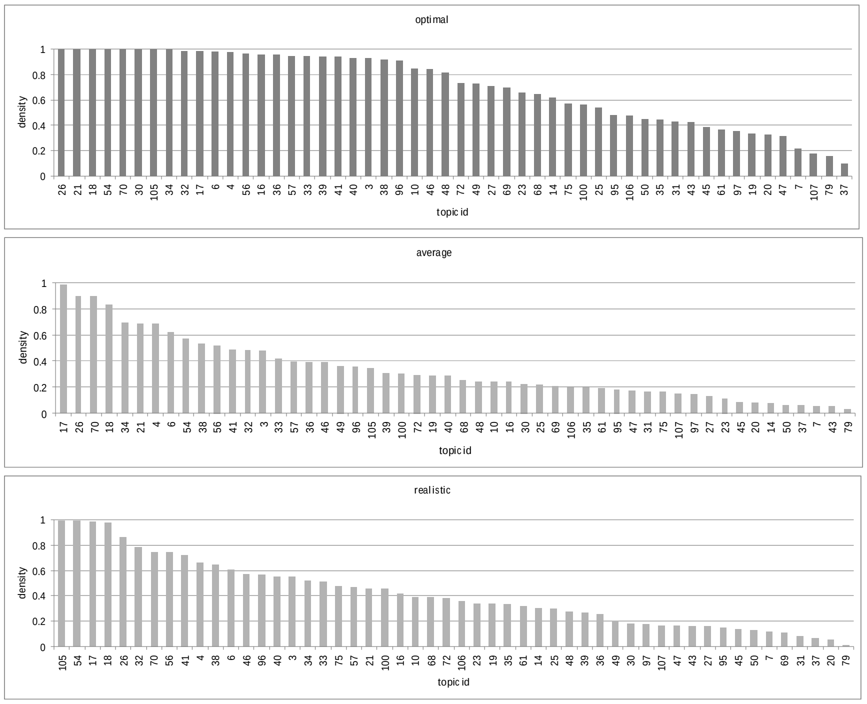


Figure 1. The average densities of retrieved documents by topic with optimal, average and realistic document retrieval scenarios. The topics are sorted according to the average density for each scenario.

Figure 2 illustrates the location of the first occurrence of relevant text, more precisely, how far the first relevant passage is from the document start on average per topic. The further the relevant content the better from the focused retrieval perspective. The document retrieval scenarios are equivalent to the analysis based on density in Figure 1, except that the criterion is based on the distance of the first relevant passage from the document start. That is, the optimal run delivers first documents having the shortest distance between document start and the first relevant passage. It is worth noting that the higher the bar, the better for focused retrieval. The average distances are approximately 230, 3049 and 1836 characters for optimal, average and realistic scenarios respectively.

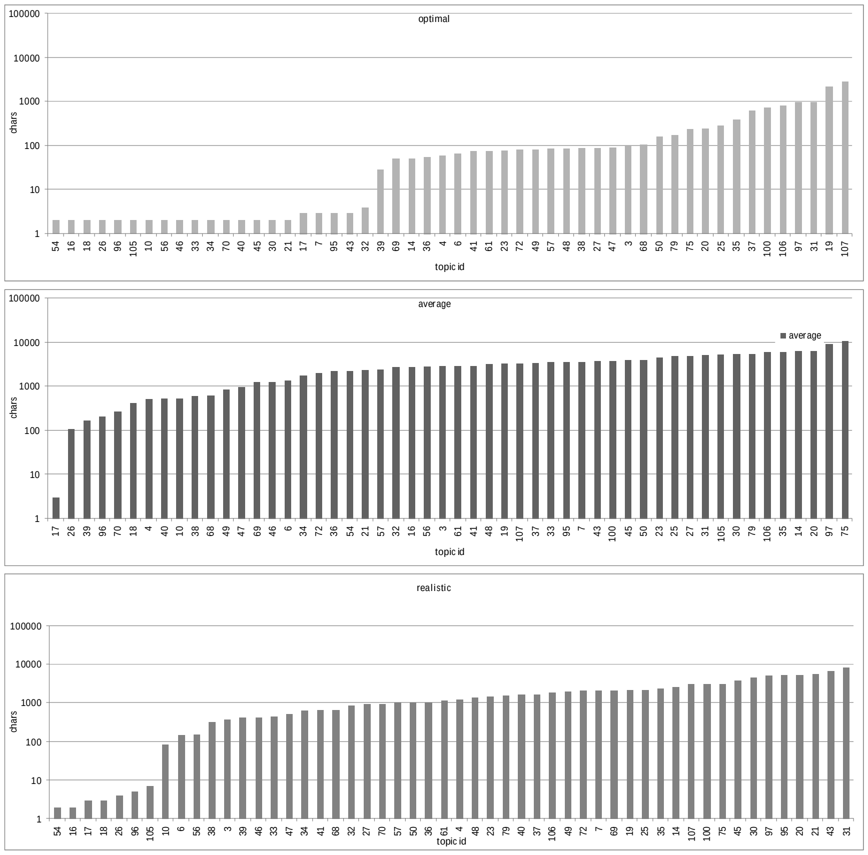


Figure 2. The average distances of the first relevant passage from the document start measured in characters by topic with optimal, average and realistic document retrieval scenarios. The topics are sorted according to the distance for each scenario.

In the next section, we briefly introduce the cumulating effort and localizing effort metric and related assumptions regarding the assumed reading order and report results of INEX 2010 runs based on how much text the user is expected to browse through before discovering the relevant material.

3 Results using Cumulated Effort and Localizing Effort

In this paper, we consider the sequential reading order of an individual document by default following the approach of [1] (see also [3]). This means that the user starts reading from the beginning of a document and continues to read sequentially the document's text until his or her information needs are fulfilled i.e. the relevant content of the document is read, or his or her tolerance-to-irrelevance is exceeded [10]. In the context of fetch and browse approach a focused retrieval system should return a set of the best matching passages i.e. a set of character-positions and their locations within each retrieved document. This aims to guide the user first to the best content of the document. When combining the two browsing, we are able to define a simple browsing model for a document with two consecutive phases:

1. The text passages retrieved by the passage retrieval system are read sequentially until possibly all relevant content has been reached.
2. The remaining passages are read until all relevant content has been reached starting from the first remaining passage of the document.

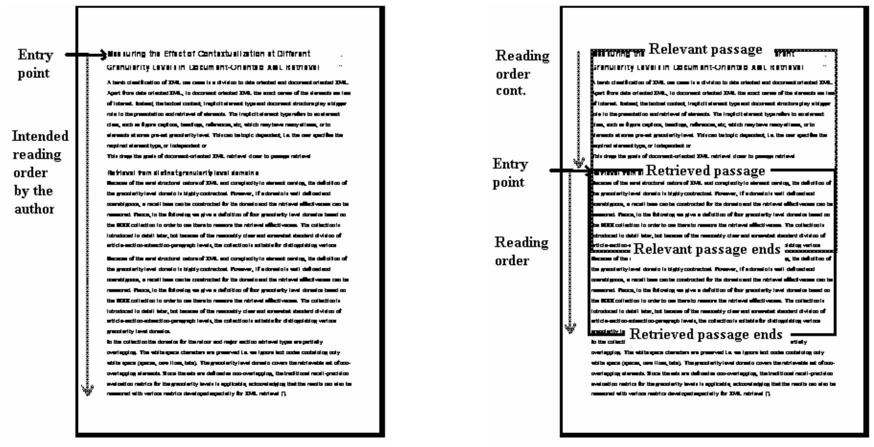


Figure 3. Conventional reading order (left) and focused retrieval driven reading order (right)

Figure 3 illustrates the difference between conventional browsing and focused retrieval driven browsing. The conventional browsing is assumed to be rather straightforward. In real life, the user might use skim reading in order to locate the relevant spots. However, when using a small screen device [e.g. 2] this option is limited. Nevertheless, focused retrieval is beneficial if the focused retrieval driven browsing methods overcome the conventional ones. In other words, it is beneficial only if the relevant content is yielded with less effort.

Next, we aim to measure the effort the user has to take in order to localize the relevant content. In other words, we measure the effectiveness in assessing the document to be relevant. This is done by assuming the reading order above and evaluating the INEX 2010 runs with cumulated effort metric, which is introduced next.

3.1 Metrics

Cumulated effort [3] is similar to the cumulated gain metric [6], except that instead of the gain the user receives by reading the documents in the result list, cumulated effort (*CE*) focuses on the effort the user has to spend while looking for relevant content. For calculating *CE*, an effort score for each ranked document *d*, *ES*(*d*), is needed. The values of *ES*(*d*) should increase with the effort; in other words the lower the score the better. Normalized cumulated effort (vector *NCE*) averages the scores over multiple topics. It is defined as follows:

$$NCE[i] = \sum_{j=1}^{i} \left(\frac{ES(d_j)}{IE[j]} - 1 \right) \qquad (1)$$

where *i* is the cut-off point in the result list and *IE* is the vector representing the ideal performance for the topic. A normalized optimal run produces a curve having zero values only. In this study, we report a value for the whole result list or a run. An average at a given cut-off point for normalized cumulated effort is calculated as follows:

$$ANCE[i] = \frac{\sum_{j=1}^{i} NCE[j]}{i} \qquad (2)$$

In this paper, we report a *MANCE*@300 value, which is calculated over a set of topics. This means the mean average cumulated effort at 300 top ranked documents. The function ES(*d*) can be defined in numerous ways, but here we assume that the system's task is just to point out that the retrieved document is relevant by guiding the user to relevant content. The document score represents how much expected effort it takes to find relevant text within the document. The scoring depends directly on (non-relevant) characters read before finding the first relevant passage or element.

The document effort score *ES*(*d*) is the score, that the *LE* function gives after the relevant text within the document is yielded. For non-relevant documents we assume a default effort score *NR*:

$$ES(d) = \begin{cases} LE(r_{d'}) & \text{, if d is relevant} \\ NR & \text{, otherwise} \end{cases} \qquad (3)$$

where *d'* is the expected reading order of document *d* and $r_{d'}$ is the position of the first relevant character with the reading order *d'* , i.e. number of characters to be read before the relevant text is yielded. The function $LE(r_{d'})$ gives the localizing effort score for an individual document, when $r_{d'}$ characters are read before the relevant text. Measuring the effort on finding relevant content is done with the Localizing Effort metric for the document score and Cumulated Effort for the list score. As scoring for an individual document, we set:

$$LE(i) = \begin{cases} 1 & ,if\ i \le 500 \\ 2 & ,if\ 500 < i \le 1000 \\ 3 & ,if\ 1000 < i \le 1500 \\ 4 & ,otherwise \end{cases} \qquad (4)$$

NR = 5

Next, we report results using the CE metrics for the list score together with localizing effort score for each individual result document.

3.2 Results

Figure 4 shows the NCE curves of four INEX 2010 runs by the top 5 participants (Emse301R, I10LIA1FTri, 31p167, Reference/qtau) plus the reference run (full document run: Reference_1). Table 1 presents the MANCE@300 value for all the runs of the participating organizations.

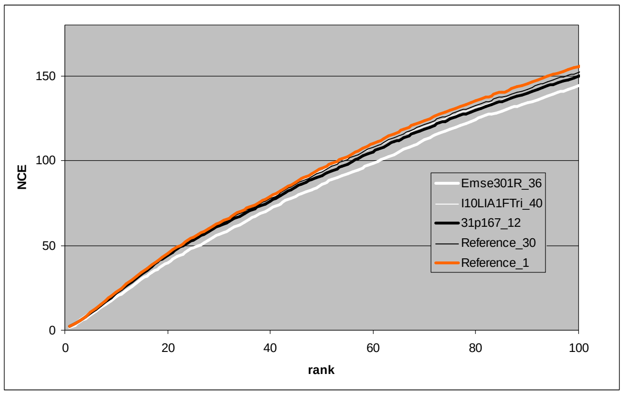


Figure 4. Normalized cumulated effort curves of the runs of some of the best participants.

Table 1. Mean Average Normalized Cumulated Effort (MANCE) at top 300 documents for each submitted run**.**

Institution	Run	MANCE
ENSM-SE	Emse301R	158.9
ENSM-SE	Emse303R	161.4
LIA - University of Avignon	I10LIA1FTri	164.9
Peking University	31p167	167.5
Peking University	37p167	167.5
LIA - University of Avignon	I10LIA1FUni	167.9
Peking University	32p167	169.0
Peking University	36p167	169.0
Queensland University of Technology	Reference	169.8
LIA - University of Avignon	I10LIA1EITri	170.1
University of Otago	Reference	171.0
University of Otago	pfrf-0.05	172.2
University of Otago	v_sstem	172.3
Peking University	23p167	172.7
Peking University	29p167	172.7
Peking University	22p167	172.8
Peking University	28p167	172.8
LIA - University of Avignon	I10LIA1EIUni	173.3
RMIT University	RMIT10titleO	173.4
Peking University	21p167	173.4
Peking University	27p167	173.4
Renmin University of China	ruc-2010-base2	173.7
Radboud University Nijmegen	runRiCORef	173.9
Full document	Reference	174.0
University of Otago	pfrf-0.10	174.5
University of Otago	v_porter	175.1
Peking University	24p167	175.4
Peking University	30p167	175.4
University of Otago	v_otago_w_pmi	176.0
University of Otago	v_ostem_w_jts	176.0
University of Otago	v_otago_stem_1	176.1
University of Otago	v_no_stem	177.0
Renmin University of China	ruc-2010-base1	177.8
RMIT University	RMIT10title	180.6
University of Otago	pfrf-0.25	186.2
ENSM-SE	Emse301	192.5
ENSM-SE	Emse303	194.0
University of Otago	v_sstem_w_jts	203.1
INDIAN STATISTICAL INSTITUTE	ISI2010_ric_ro	207.3
Doshisha University	DUR10atcl	211.1
University of Amsterdam	0	213.7
University of Amsterdam	0	217.7
University of Waterloo	UWBOOKRIC2010	223.0
Peking University	34p167	224.1
Peking University	35p167	228.9
INDIAN STATISTICAL INSTITUTE	ISI2010_ric_aggr	240.3
Saint Etienne University	UJM_33456	249.3
Doshisha University	DUR10BU_S	256.3
Doshisha University	DUR10BU_D	256.3

4 Discussion and Conclusions

This study aimed to motivate the RiC task by analyzing the relevance densities and the locations of the relevant material in the relevant result documents. The location was measured as a distance in characters between the document start and the start of the first relevant passage. Three document retrieval scenarios: optimal, average and realistic, were considered. With the optimal scenario, the average density of the top 5 documents was below 50% for only a minority of the topics. However, assuming the realistic scenario, a majority of the topics went below 0.5 density (using the top 5 relevant documents).

The same trend was present in the location analysis. In the optimal scenario in most of the topics, the relevant content was situated on average within 100 characters or less from the document start using top 5 documents for each topic. In the realistic scenario most of the first relevant material within a document was situated within 1000 characters or more in most of the topics. Accordingly, the results obtained with the localizing effort metric as document level metric and cumulated effort metric as list level metric showed the benefit of focused retrieval over plain full document retrieval.

Consequently, the Relevant-in-Context task of INEX seems to be beneficial in studying means to reduce user effort in locating relevant material within a document. This is the case even if the Wikipedia documents tend to be relatively short. Relevance sparsity and long not relevant document parts require scrolling when only conventional document retrieval is used. Scrolling thousands of characters for instance with a cumbersome small screen device requires effort, which can be aided using focused retrieval driven browsing methods within a document.

Acknowledgements

The study was supported by Academy of Finland under grant #115480

References

1. Arvola, P. Passage Retrieval Evaluation Based on Intended Reading Order, In *Workshop Information Retrieval*, LWA 2008, 91-94. 2008.
2. Arvola, P., Junkkari, M. and Kekäläinen, J. Applying XML Retrieval Methods for Result Document Navigation in Small Screen Devices. In *Proceedings of MUIA at MobileHCI 2006*, 6-10, 2006.
3. Arvola, P., Kekäläinen, J., and Junkkari, M. Expected reading effort in focused retrieval evaluation. *Information Retrieval*, 13(4), 460-484, 2010.
4. Arvola, P., Kekäläinen, J., and Junkkari, M. Focused access to sparsely and densely relevant documents. In *Proceedings of SIGIR 2010*, 781-782, 2010.
5. Chiaramella, Y. Information retrieval and structured documents. *Lectures on information retrieval*, 286–309, 2001.
6. Järvelin, K., and Kekäläinen, J. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems.* 20 (4). 422-446, 2002.
7. Kazai, G., and Lalmas, M. Notes on what to measure in INEX. In *Proceedings of the INEX, Workshop on Element Retrieval Methodology*, INEX 2005, 2005.
8. Piwowarski, B., and Lalmas, M. Providing consistent and exhaustive relevance assessments for XML retrieval evaluation. In *Proceedings of CIKM '04*, 361–370, 2004.
9. Trotman, A., Pharo, N., and Lehtonen, M. XML-IR Users and Use Cases. In *Proceedings of INEX 2006*, LNCS 4518, 400-412, 2007.
10. de Vries, A.P., Kazai, G., and Lalmas, M. 2004. Tolerance to irrelevance: A user-effort oriented evaluation of retrieval systems without predefined retrieval unit. In *Proceedings of RIAO 2004*, 463-473, 2004.

ENSM-SE and UJM at INEX 2010: Scoring with Proximity and Tags Weights

Extended abstract

Michel Beigbeder[1], Mathias Géry[2], Christine Largeron[2], and Howard Seck[3]

[1] École Nationale Supérieure des Mines de Saint-Étienne
michel.beigbeder@emse.fr
[2] Université de Lyon, F-42023, Saint-Étienne, France
{Mathias.Gery, Christine.Largeron}@univ-st-etienne.fr
[3] Université Paris-Dauphine
bseck@olaneo.fr

1 Introduction

The four runs labelled with “Emse” in the 2010 INEX campaign were done by a team both from the *École Nationale Supérieure des Mines de Saint-Étienne* and the *Université Jean Monnet de Saint-Étienne* and with a master student from the *Université Paris-Dauphine*. Both the approaches used in the previous years by these two organizations were merged for our 2010 participation. The first oneis based on the proximity/influence of the query terms in the documents [1] and the second one is based on learnt tags weights [2].

We will first present the notion of proximity between one (query) term and any position in a document. Then we will show how this notion can be extended to proximity between one boolean query and any position in a document. The following section 3 will be dedicated to the computation of tag weights. The integration of both methods were done by impacting the tag weights directly in the influence function of the occurrence of a query term. Finally in section 4 we present how elements are scored with these weighted proximity functions and how our runs were built with these scores and for two of them with the INEX Reference Run.

2 Influence functions

2.1 Structure, elements and logical elements

An XML document is composed of *elements*, each of them is delimited by an *opening tag* and a *closing tag*. Given a XML collection, we consider a partition of the set of tags, B, that appears in the collection with three subsets:

- B_l: the *logical tags* (or *section-like tags*);
- B_t: the *title-like tags*;
- $\overline{B_l \cap B_t}$: the other tags.

Given a position x in a document, $e(x)$ is the deepest element that surrounds the position x, and $e_l(x)$ is the deepest logical element that surrounds the position x; $b(x)$ is the tag of the element $e(x)$.

2.2 Influence function of a term to a position

The proximity between

- one occurrence of a term t at position i
- one position x in a document d

measures the influence of this occurrence of term t to the position x. Any function with the three following properties is acceptable and modelizes the proximity idea:

- symmetric around i,
- decreasing with the distance to i,
- maximum (value 1) reached at i.

The simplest one is a linearly decreasing function centered around i: $x \mapsto \max(\frac{k-|x-i|}{k}, 0)$ where k is a controlling parameter. When the distance between x and i is greater than k, the influence is zero – that's to say that the occurrence of t at position i is too far from position x to influence it. Moreover the influence is limited to the logical element $e_l(i)$ that surrounds the position i of the occurrence of the query term t. To do that we take the product of the triangle function by the characteristic function $\mathbf{1}_{e_l(i)}$ of the position range that belongs to the logical element $e_l(i)$. Lastly, the influence should be that of the nearest occurrence of the term t, which can be obtained with $\max_{i \in d^{-1}(t)}$ because the influence function are symmetric and decreasing with the distance.[4]

So the proximity $p_t^d(x)$ of term t to the position x in the document d is defined by:

$$p_t^d(x) = \max_{i \in d^{-1}(t)} \left(\mathbf{1}_{e_l(i)} \cdot \max \left(0, \frac{k - |x-i|}{k} \right) \right) \tag{1}$$

Though when $e(i)$ is a title-like element, the triangle function is replaced by the constant function 1. Thus one occurrence of a query term in a title spreads its influence over the whole surrounding logical element.

2.3 Influence function of a query to a position

As a boolean query, the query q is a tree with conjunctive and disjunctive nodes. To define the proximity on a conjunctive node the minimum is taken over the proximity functions of its children. Similarly, the proximity on a disjunctive node is defined as the maximum over the proximity functions of its children.

[4] The notation $d^{-1}(t)$ denotes the set of positions in the document d where one occurrence of term t does appear.

2.4 Score of of an element

Given the influence function of a document d to a query q that maps the positions in the document d to [0,1] with $p_q^d(x)$, the score of an element e is computed with the following formula:

$$s(q,e) = \frac{\sum_{x_1(e) \leq x \leq x_2(e)} p_q^{d_e}(x)}{x_2(e) - x_1(e) + 1} \tag{2}$$

3 Weighting tags and impacting tag weights

3.1 Weighting tags

A weight is computed for each tag $b \in B$, following the learning method introduced by [2]. It estimates the probability that b marks a relevant term or an irrelevant one. This weight is integrated afterwards in the influence function of the terms.

The queries set Q from INEX 2009 is used as a learning set. As presented in the table 1, $P_q(e)$ is the set of the relevant positions in the element $e \in E$ for a query $q \in Q$, and $M_b(e)$ is the set of the positions of e marked by the tag $b \in B$.

	$P_q(e)$	$\overline{P_q(e)}$
$M_b(e)$	$t_{pm}(b,q)$	$t_{\overline{p}m}(b,q)$
$\overline{M_b(e)}$	$t_{p\overline{m}}(b,q)$	$t_{\overline{p}\overline{m}}(b,q)$
Total $= P_q \cup \overline{P_q}$	$t_p^{coll}(q)$	$t_{\overline{p}}^{coll}(q)$

Table 1. Positions of the element $e \in E$, for each query q and for each tag b

The weight $w_b(q)$ of a tag b for a query q is defined by:

$$w_b(q) = \frac{\frac{t_{pm}(b,q)+s}{t_{pm}(b,q)+t_{p\overline{m}}(b,q)+s}}{\frac{t_{\overline{p}m}(b,q)+s}{t_{\overline{p}m}(b,q)+t_{\overline{p}\overline{m}}(b,q)+s}} \tag{3}$$

with:

- $t_{pm}(b,q) = \sum_{e \in E} |P_q(e) \cap M_b(e)|$: number of relevant positions for the query q marked by the tag b;
- $t_{p\overline{m}}(b,q) = \sum_{e \in E} |P_q(e) \cap \overline{M_b(e)}|$: number of relevant positions for the query q not marked by the tag b;
- $t_{\overline{p}m}(b,q) = \sum_{e \in E} |\overline{P_q(e)} \cap M_b(e)|$: number of irrelevant positions for the query q marked by the tag b;
- $t_{\overline{p}\overline{m}}(b,q) = \sum_{e \in E} |\overline{P_q(e)} \cap \overline{M_b(e)}|$: number of irrelevant positions for the query q not marked by the tag b;

The parameter s is a smoothing parameter, which was fixed to 0,5 in our experiments.

The weight w_b of a tag is then averaged using a set of 68 queries from INEX 2009, using the formula 4:

$$w_b = \frac{1}{|Q|} \sum_{q \in Q} w_b(q) \tag{4}$$

3.2 Impacting tag weights

The weights of the tags are impacted on the influence function with two methods. In the first one, the height of the triangle is impacted according to the formula:

$$ph_t^d(x) = \max_{i \in d^{-1}(t)} \left(\max \left(0, w_{b(i)} \cdot \frac{k - |x - i|}{k} \right) \right) \tag{5}$$

and in the second one, both the height and the width of the triangle are impacted:

$$phl_t^d(x) = \max_{i \in d^{-1}(t)} \left(\max \left(0, \frac{w_{b(i)} \cdot k - |x - i|}{k} \right) \right) \tag{6}$$

4 Building runs

For the experiments, we used the following sets of tags:

$$B_l = \{\texttt{article}, \texttt{sec}, \texttt{section}, \texttt{ss1}, \texttt{ss2}, \texttt{ss3}, \texttt{ss4}, \texttt{ss5}\}$$

$$B_t = \{\texttt{title}, \texttt{st}\}$$

For the runs Emse301 and Emse301R, the influence function of a query term is *phl*, and for the runs Emse303 and Emse303R, the influence function is *ph*.

As each document is analyzed, a score is computed for each logical element according to formula 2. A score is computed for a document as the maximum of the scores of its descendants.

To choose some elements within a document, the scores of the elements of the document are sorted in decreasing order in a ranked list. The top ranked element is inserted in the result list. To fulfill the non overlapping requirement, at the same time every descendants and every ascendants of this element are removed from the ranked list. This process is repeated util the ranked list is empty.

For the runs Emse301 and Emse303, the elements are sorted:

1. by document score
2. by document id
3. by element score

For the runs Emse301R and Emse303R, the same sorting keys are used but the Reference Run is also used. The elements of the documents that appear both in our results list and in the Reference Run are returned in the order of the Reference Run, then the elements of the documents that appear only in our list.

References

1. Beigbeder, M.: Focused retrieval with proximity scoring. In: Proceedings of the 2010 ACM Symposium on Applied Computing. SAC '10, New York, NY, USA, ACM (2010) 1755–1759
2. Géry, M., Largeron, C., Thollard, F.: Integrating structure in the probabilistic model for information retrieval. In: Web Intelligence. (2008) 763–769

LIP6 at INEX'10 : OWPC for Ad Hoc track

David Buffoni, Nicolas Usunier, and Patrick Gallinari

Université Pierre et Marie Curie - Laboratoire d'Informatique de Paris 6
4, place Jussieu, 75005 Paris, France
{buffoni, usunier, gallinari}@poleia.lip6.fr

Abstract. We present a Retrieval Information system for XML documents using a Machine Learning Ranking approach. We propose this year, to extract other features than the previous year, to enhance the precision of our machine learning runs.

1 Introduction

Learning to rank algorithms have been used in the Machine Learning field for a while now. In the field of IR, they have first been used to combine features or preferences relations in the meta search [5], [6]. Learning ranking functions has also lead to improved performances in a series of tasks such as passage classification or automatic summarization [1]. More recently, they have been used for learning the rank function of search engines [4], [11], and [10].

Ranking algorithms work by combining features which characterize the data elements to be ranked. In our case, these features will depend on the document or the element itself, its structural context and its internal structure. Ranking algorithms will learn to combine these different features in an optimal way, according to a specific loss function related to IR criteria, using a set of examples. This set of examples is in fact a set of queries where for each one, a list of documents is given. In ranking, starting from this list, we make a set of pairs of documents where one is relevant to the query and the other is irrelevant.

The main problem in ranking is that the loss associated to a predicted ranked list is the mean of the pairwise classification losses. This loss is inadequate for IR tasks where we prefer high precision on the top of the predicted list. We propose, here to use a ranking algorithm, named OWPC [10] which optimizes loss functions focused on the top of the list.

This year, we concentrated our attention on which features we have to take into account to enhance the precision of our runs.

In this paper, we describe the selection of features which represent an element according a query (section 2). We then present our learning to rank model, OWPC, in section 3. Finally, in section 4 we discuss the results obtained by our runs for the adhoc track.

2 Learning Base

INEX uses a document collection based on the Wikipedia where the original Wiki syntax has been converted into XML [8]. This collection is composed of 2,666,190 documents and for each document we can separate semantic annotation elements and content elements. In our case, we indexed only content elements without doing any preprocessing step along the corpus as stemming or using a stop word list. To use a machine learning algorithm on this collection we have to build manually a set of examples (i.e of queries) as a learning base. For each query, we must create a pool of retrieved elements, assess them and represent them as a vector of extracted features.

We assessed manually 40 randomly chosen queries from previous INEX competitions (from 2006 to 2009) and according the same process as in [3].

Now, after annotating a set of queries, we can select the elements we will propose to our learning algorithm. Thus, we have to select elements according to a query. The strategy used was to select objects by a pooling technique, more details can be found in [2] where the authors suggest selecting elements by pooling. Therefore we decided to build our learning base as a pool of the top t elements of three information retrieval models as BM25, LogTF and a Language Model with Absolute Discount smoothing function.

2.1 Basic features

Once, we have obtained a set of elements according a query, we have to represent each element by a feature representation. In our case each feature is a similarity function between a query and an element. We can separate the features in two families :

content feature: is a similarity function based on the content of an element and the content of the query. For example, models such as BM25 [7], TF-IDF or Language Models can be placed in this class.

element structure feature: provides information on the internal structure of an element. In practice, we used an indicator function on the type of the considered element.

We sum up in Table 1 all the features used in our experiments for the Restricted Focused and the Efficiency/Thorough tasks[1]. We denote $tf(t,x)$ as the term frequency of the term t of a query q in the element x, $[x]$ as the length of x, Col as the collection and $df(t)$ as the number of elements in the corpus containing t.

2.2 Context features

The aim of this work is to identify features that are informative to help a learning to rank algorithm to perform well on a set of elements. In traditional search engines, the features of the whole document are used, but the features of importance in XML retrieval, where the retrieved element must be both specific and

[1] These features are commonly employed in IR but we were interested in what kind of information could bring these features

ID	Feature Description
1	$[x]$
2	# of unique words in x
3	$\sum_{t\in q\bigcap x} tf(t,x)$ in x
4	$\sum_{t\in q\bigcap x} log(1+tf(t,x))$ in x
5	$\sum_{t\in q\bigcap x} \frac{tf(t,x)}{[x]}$ in x
6	$\sum_{t\in q\bigcap x} log(1+\frac{tf(t,x)}{[x]})$ in x
7	$\sum_{t\in q\bigcap x} log(\frac{[Col]+1}{df(t)+0.5})$ in x
8	$\sum_{t\in q\bigcap x} log(1+log(\frac{[Col]+1}{df(t)+0.5}))$ in x
9	$\sum_{t\in q\bigcap x} log(1+\frac{[Col]+1}{tf(t,x)})$ in x
10	$\sum_{t\in q\bigcap x} log(1+\frac{tf(t,x)}{[x]}*\frac{[Col]+1}{df(t)+0.5})$ in x
11	$\sum_{t\in q\bigcap x} log(1+\frac{tf(t,x)}{[x]}*\frac{[Col]}{tf(t,Col)})$ in x
12	$\sum_{t\in q\bigcap x} log(1+tf(t,x))*log(\frac{[Col]+1}{0.5+df(t)})$ in x
13	$\sum_{t\in q\bigcap x} tf(t,x)*log(\frac{[Col]+1}{0.5+df(t)})$ in x
14	$BM25^{k=2}_{b=0.2}$ in x
15	$log(1+BM25^{k=2}_{b=0.2})$ in x
16	$LM^{Jelinek-Mercer}_{\lambda=0.5}$ in x
17	$LM^{Dirichlet}_{\mu=2500}$ in x
18	$LM^{AbsoluteDiscount}_{\delta=0.88}$ in x
19-20-21-22-23-24-25	*family tag of the element*

Table 1. Extracted features for the joint representation of a document according to a query. x is either an element or a document (Reference runs). The 19-25th features give a boolean attribute for the family tag of the current element $x \in \{sec, ss, ss1, ss2, ss3, ss4, p\}$.

exhaustive, remain to be determined. The approach we present focuses on this problem.

We built a total of three sets of context features based on the 15 first features described in Table 1. We then added independently features 16 to 23.

Reference Run submission: This submission describes an element through 23 features (exactly as presented in Table 1 where x is the document, i.e. /article[1] element). This submission is used as a baseline and called **LIP6-OWPCRefRun***[2] .

Element-Document and ratio submission: In this case, an element x can take two values, that of the retrieved element and that of the document it belongs to. We add the ratio $r = \frac{feature_i(x)}{feature_i(document\ it\ belongs\ to)}$ too. This ratio should compare the proportion of the information included in the element to the overall information of the document, to determine whether the element is exhaustive. We inject into the algorithm both the local information held by the element and the global information brought by document scores. We suppose that the best element is going to be located in one of the most pertinent documents, and thus we select the best documents and then the best elements of these documents. This set gives the submission **LIP6-OWPCnormal***.

Element-Parent-Document and ratios submission: In this case, x can take three values, that of the retrieved element, that of his parent and that of the document it belongs to. We add two ratios: $r1 = \frac{feature_i(x)}{feature_i(parent(x))}$ and $r2 = \frac{feature_i(parent(x))}{feature_i(document\ it\ belongs\ to)}$. As previously, we added the context information of the element by computing the scores of his parent. We know that the information inside a parent is greater or equally exhaustive, compared to that of the element and it remains more specific than the information in the whole document. This submission is **LIP6-OWPCparent***.

3 Learning to rank Model

We outline here the learning to rank model described more in detail in [10]. We consider a standard setting for learning to rank. The ranking system receives as input a query q, which defines a sequence of candidates (XML elements) $\mathbf{X}(q) \overset{def}{=} \big(\mathbf{X}_1(q), ..., \mathbf{X}_{[q]}(q)\big)$ (where $[q]$ is used as a shortcut for $[\mathbf{X}(q)]$). In our case, the sequence is a subset of the collection filtered by the pooling technique as described in section 2. $\mathbf{X}_j(q)$ corresponds to the similarity representation, i.e the vector of extracted similarities, of the j-th object. A score (i.e. real-valued) function f takes as input the similarity representation of an element, thus $f\big(X_j(q)\big)$, denoted $f_j(z)$ for simplicity, is the score of the j-th element. The output of the ranking system is the list of the candidates sorted by decreasing scores.

[2] the * corresponds to the task, in our case, "Th" for Efficiency/Thorough and "Fo" for Restricted Focused.

3.1 Learning step

For clarity in the discussion, we assume a binary relevance of the elements: a sequence $\mathbf{y}$ contains the indexes of the *relevant objects* labeled by a human expert ($\bar{\mathbf{y}}$ contains the indexes of the *irrelevant* ones).

Given a training set $S = (q_i, \mathbf{y}_i)_{i=1}^{m}$ of m examples, learning to rank consists in choosing a score function f that will minimize a given *ranking error function* $\hat{R}^{\Phi}(f, S)$:

$$\hat{R}^{\Phi}(f, S) \overset{def}{=} \hat{\mathbb{E}}_{(q,\mathbf{y})\sim S} \, \mathtt{err}\big(\mathbf{rank}(f, q, \mathbf{y})\big)$$

$\hat{\mathbb{E}}_{(q,\mathbf{y})\sim S}$ is the mean on S of ranking errors and `err` is the number of misranked relevant documents in the predicted list.

Rank function We define the *rank* of a relevant document for a given query q, its relevant candidates $\mathbf{y}$, and a score function f, as follows:

$$\forall y \in \mathbf{y}, rank_y(f, q, \mathbf{y}) \overset{def}{=} \sum_{\bar{y} \in \bar{\mathbf{y}}} \mathrm{I}\left(f_y(q) \leq f_{\bar{y}}(q)\right) \quad (1)$$

where $\mathrm{I}\left(f_y(q) \leq f_{\bar{y}}(q)\right)$ indicates whether the score of a relevant document is lower than the score of an irrelevant one. However, directly minimizing the ranking error function $\hat{R}^{\Phi}(f, S)$ is difficult due to the non-differentiable and non-convex properties of function $\mathrm{I}\left(f_y(q) \leq f_{\bar{y}}(q)\right)$ of $rank_y(f, q, \mathbf{y})$. To solve this problem we generally take a convex upper bound of the indicator function which is differentiable and admits only one minimum. In [10], this bound is denoted $\ell\big(f_y(q) - f_{\bar{y}}(q)\big)$ and is set to the hinge loss function $\ell : t \mapsto [1-t]_+$ (where $[1-t]_+$ stands for $max(0, 1-t)$ and $t = f_y(q) - f_{\bar{y}}(q)$).

Error function With equation 1, we can define a general form of the ranking error functions `err` of a real valued function f on $(q, \mathbf{y})$ as:

$$\mathtt{err}(f, q, \mathbf{y}) \overset{def}{=} \frac{1}{[\mathbf{y}]} \sum_{y \in \mathbf{y}} \Phi_{[\bar{\mathbf{y}}]} \left(rank_y(f, q, \mathbf{y})\right)$$

where $\Phi_{[\bar{\mathbf{y}}]}$ is an aggregation operator over the position of each relevant document in the predicted list. Traditionnaly, this aggregation operator $\Phi_{[\bar{\mathbf{y}}]}$ was set to the mean in learning to rank algorithms as in [**?**, **?**]. Yet, optimizing the mean of the rank of relevant document does not constitute a related ranking error function to classical Information Retrieval measures. In fact, we obtain the same ranking error for a relevant element ranked on the top or on the bottom of the list. This behaviour is not shared by IR metrics where more consideration is given to the rank of the relevant documents on the top of the list.

To overcome this problem, the authors of [10] showed that fixing $\Phi_{[\bar{\mathbf{y}}]}$ by the convex Ordered Weighted Aggregation (OWA) operators [**?**] we can affect the degree to which the ranking loss function focuses on the top of the list. The definition of the OWA operator is given as follows :

Definition 1 (OWA operator [?]) *Let $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)$ be a sequence of n non-negative numbers with $\sum_{j=1}^{n} \alpha_j = 1$. The* Ordered Weighted Averaging (OWA) Operator *associated to $\boldsymbol{\alpha}$, is the function* $\mathrm{owa}^{\boldsymbol{\alpha}} : \mathbb{R}^n \to \mathbb{R}$ *defined as follows:*

$$\forall \mathbf{t} = (t_1, ..., t_n) \in \mathbb{R}^n, \mathrm{owa}^{\boldsymbol{\alpha}}(\mathbf{t}) = \sum_{j=1}^{n} \alpha_j t_{\sigma(j)}$$

where $\sigma \in \mathfrak{S}_n$ (set of permutations) such that $\forall j, t_{\sigma(j)} \geq t_{\sigma(j+1)}$.

It is then used by the authors to rewrite the ranking error function as follows:

$$\mathtt{err}(f, q, \mathbf{y}) \overset{def}{=} \frac{1}{[\mathbf{y}]} \sum_{y \in \mathbf{y}} \underset{\bar{y} \in \bar{\mathbf{y}}}{\mathrm{owa}} (rank_y(f, q, \mathbf{y})) \tag{2}$$

SVM formulation Thus, according to the authors, this provides a regularized version of the empirical risk of equation (2) and can be solved using existing algorithms as Support Vector Machines for structured output spaces [9].

$$\min_{w} \frac{1}{2} ||w||^2 + C \sum_{(q,\mathbf{y}) \in S} \frac{1}{[\mathbf{y}]} \sum_{y \in \mathbf{y}} \underset{\bar{y} \in \bar{\mathbf{y}}}{\mathrm{owa}} [1 - \langle w, X_y(q) - X_{\bar{y}}(q) \rangle]_+ \tag{3}$$

where the learning algorithm according to the training set S and the ranking error function $\mathtt{err}(f, q, \mathbf{y})$ will determine the parameter vector w. This weight vector will be used in the prediction step to compute the score of a document (f function). C is a trade-off parameter fixed by the user, to balance the learning model complexity $||w||^2$ and the upper bounded ranking loss function $\mathtt{err}(f, q, \mathbf{y})$.

To sum up, this algorithm learns a score function by minimizing a ranking error function focused on the top of the list. The user has to fix the non-increasing weights α of the OWA operators to vary the consideration on the errors incurred on the top of the list. It's the typical behaviour of a IR evaluation measure.

3.2 Ranking prediction

Given an unlabeled query q_u with a candidate $X_j(q_u)$ of the sequence of elements $\mathbf{X}(q_u)$, the corresponding predicted score based on the learned weight vector w is:

$$f_j(z) = \langle w, X_j(q_u) \rangle$$

where $\langle ., . \rangle$ is the scalar product between the weight vector and the similarity representation of $X_j(q_u)$. This allows us to sort all elements of $\mathbf{X}(q_u)$ by decreasing scores.

4 Experiments - Results

In this section, we present our experiments and results for the Restricted Focused and the Efficiency/Thorough tasks in Adhoc track for INEX'10. We concentrated on these two tasks to validate the extraction of elements features which are provided to the learning to rank algorithm.

To do that, for each query, we use a fetch and browse strategy. We retrieve the top t ($t = 1,500$) articles for the fetch step and we extract the list of all overlapping elements (top $t' = All$) which contained at least one of the search terms for the browse step. We strive to collect only small elements and we limit the domain to types $\in \{sec, ss, ss1, ss2, ss3, ss4, p\}$. Thus, the side effects due to waste labeling are reduced.

We fixed the parameters of our learning to rank algorithm on a validation set which is Inex'09's queries . Parameters are selected by using iP[0.01] measure. We set the weights of the OWA operator to be linearly decreasing as suggested by [10]. We fixed the C parameter of the equation (3) among $\{1, 10, 100\}$ using the iP[0.01] score on the Inex'09 set of queries. In our experiments, $C = 10$ gave us the best performances.

4.1 Efficiency/Thorough task

We present here only the runs experimented for the Thorough task. The performances of our models are summarized in Table 2 for the Efficiency/Thorough task.

As explained above (§2), LIP6-OWPCRefRunTh is our reference run and returning only top 1,500 articles. LIP6-OWPCnormalTh retrieves small elements which are bringing only information on the document and the considered element. In end, LIP6-OWPCparentTh gives top 1,500 elements ranked according the information given by the element, its parent and the document is belong to.

	MAiP	MAP
LIP6-OWPCRefRunTh	**0.2196**	**0.2801**
LIP6-OWPCnormalTh	0.1673	0.2561
LIP6-OWPCparentTh	0.1800	0.2581

Table 2. Test performances of OWPC model in the Efficiency/Thorough task in terms of MAiP and MAP.

As expected, runs which return articles have better performances in MAP and MAiP than runs returning only small elements. This shows that the strategy of retrieving articles is most informative both with respect to precision and recall. A simple explanation for this, is the effect of the limitation of the results list. In fact, only the top 1500 elements for each query are evaluated and in the case of the Efficiency/Thorough task, where the overlap is permitted, this penalizes runs returning a lot of small elements rather than one article.

In end, taking also into account information on the parent of the considered element, performs better than a run where information is only given by the element and the document. That is encouraging for our Restricted Focused runs.

4.2 Restricted Focused task

We present in this section the performances of our models for the Restricted Focused task. For this task, our model retrieves only small elements due to the limitation of retrieving 1,000 characters by query.

We report the performances of our models in the Restricted Focused task in Table 3 in terms of character precision measure.

	Char_prec	MAP
LIP6-OWPCRefRunFo	0.3391	0.0016
LIP6-OWPCnormalFo	0.3451	0.0013
LIP6-OWPCparentFo	**0.4125**	**0.0022**

Table 3. Test performances of OWPC in the Restricted Focused task in terms of Character Precision and MAP.

LIP6-OWPCRefRunFo is our reference run and returning only articles from the LIP6-OWPCnormalFo run and taking into account the size limitation of the query. LIP6-OWPCnormalFo retrieves small elements which are bringing only information on the document and the considered element. We remove overlapping elements, and we limit the size of the ranked list for each topic to 1,000 characters. As previously, LIP6-OWPCparentTh gives elements ranked according the information given by the element, its parent and the document is belong to.

We can see that in terms of character precision and MAP the learning to rank algorithm with information providing by the parent (LIP6-OWPCparentFo) outperforms other runs. This is the same trend that in the Efficiency/Thorough task.

In our case, the learning to rank algorithm performs better when an information on the parent of the element is given in terms of character precision. In terms of MAP, this difference is less significative.

5 Conclusion

In conclusion, we built a training set for our learning model named OWPC. We studied different ways to extract informative features and see their influence in terms of Character Precision, MAiP and MAP measures.

As in previous INEX competitions, retrieving articles rather than smaller elements gives better results in terms of MAiP (and MAP if there are not a limitation on the size of each topic). However, OWPC which retrieves only small elements performed well on the Restricted Focused task where the evaluation measure (character precision) gives more importance to the precision of the run than its recall.

References

[1] Massih-Reza Amini, Nicolas Usunier, and Patrick Gallinari. Automatic text summarization based on word-clusters and ranking algorithms. In David E. Losada and Juan M. Fernández-Luna, editors, *ECIR*, volume 3408 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2005.

[2] Javed A. Aslam, Evangelos Kanoulas, Virgil Pavlu, Stefan Savev, and Emine Yilmaz. Document selection methodologies for efficient and effective learning-to-rank. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 468–475, New York, NY, USA, 2009. ACM.

[3] David Buffoni, Nicolas Usunier, and Patrick Gallinari. Lip6 at inex'09: Owpc for ad hoc track. In Shlomo Geva, Jaap Kamps, and Andrew Trotman, editors, *INEX*, volume 6203 of *Lecture Notes in Computer Science*, pages 59–69. Springer, 2009.

[4] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *NIPS*, pages 193–200, 2006.

[5] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. In *NIPS*, 1997.

[6] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *JMLR*, 4:933–969, 2003.

[7] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at trec. In *TREC*, pages 21–30, 1992.

[8] Ralf Schenkel, Fabian M. Suchanek, and Gjergji Kasneci. Yawn: A semantically annotated wikipedia xml corpus. In Alfons Kemper, Harald Schöning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, and Christoph Brochhaus, editors, *BTW*, volume 103 of *LNI*, pages 277–291. GI, 2007.

[9] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[10] Nicolas Usunier, David Buffoni, and Patrick Gallinari. Ranking with ordered weighted pairwise classification. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *ICML*, volume 382 of *ACM International Conference Proceeding Series*, page 133. ACM, 2009.

[11] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *SIGIR*, pages 271–278, 2007.

UMD at INEX 2010

Carolyn J. Crouch, Donald B. Crouch,
Sandeep Vadlamudi, Ramakrisha Cherukuri, Abhijeet Mahule

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

Abstract. This paper reports the final results of our experiments involving the 2009 INEX Ad-Hoc Track and describes the methodology upon which our current, 2010 experiments are built. In 2009, our INEX investigations centered on indentifying a methodology for producing what we have referred to as *improved focused elements*—i.e., elements which when evaluated are competitive with others in the upper ranges of the official rankings. Our 2009 INEX paper [1] describes our approach to producing such elements, which is based on the combination of traditional document retrieval (to identify the document set of interest to the query) with our method of dynamic element retrieval (to generate and retrieve the elements of the document tree so identified) and the application of a specific focusing technique (to select the focused elements). The system is based on the Vector Space Model [2]; basic functions are performed using the Smart experimental retrieval system [3]. In this paper, we report the final results of these experiments, applied to the INEX 2009 Focused and Relevant-in-Context tasks. (Results of the Thorough task are reported as well.) These results show that our approach produces highly ranked results for all three of these Ad Hoc tasks. Significance tests, applied to these results as compared to the top-ranked runs [details included], show in which cases statistically significant results are obtained. Our 2010 work is ongoing at present.

References

[1] Crouch, C., Crouch, D., Bhirud, D., Poluri, P., Polumetla, C., Sudhakar, V. A methodology for producing improved focused elements. In: S. Geva, J. Kamps, and A. Trotman (Eds.): *INEX 2009*, LNCS 6203, pp.70-80, Springer, Heidelberg (2010)

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

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

A Result Reconstruction Method for Effective XML Fragment Search at INEX 2010

Atsushi Keyaki[1], Kenji Hatano[2], and Jun Miyazaki[3]

[1] Graduate School of Culture and Information Science, Doshisha University
1-3 Tatara-Miyakodani, Kyotanabe, Kyoto 610-0394, Japan
keyaki@ilab.doshisha.ac.jp,
[2] Faculty of Culture and Information Science, Doshisha University
1-3 Tatara-Miyakodani, Kyotanabe, Kyoto 610-0394, Japan
khatano@mail.doshisha.ac.jp,
[3] Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
miyazaki@is.naist.jp,

Abstract. In this paper, we propose an extension scheme of the result reconstruction method that we previously proposed. In the previous method, we aimed to extract more relevant fragments without irrelevant parts. Since the challenge of Ad hoc track in INEX 2010 is to generate "snippets" which return only small size of contents, we extend our method along the concept of INEX 2010. To generate meaningful snippets, we propose a novel way of extracting search results and adjust our previous methods to be suitable one for restricted results in its text size.

Keywords: XML fragment search, result reconstruction method, meaningful snippets

1 Introduction

The extensible markup language[4] (XML) is a markup language for structured documents that has become the de facto format for data exchange. A large number of XML documents are available on the Web. We expect this trend to continue in the future. As such, information retrieval techniques for searching XML documents are very important and required in the field.

An XML fragment is usually defined as a part of a larger XML document. The fragment is identified by the surrounding start and end tags. XML fragments are, therefore, nested and have containment relationships with each other, which can cause overlaps in XML fragments in an XML document. The key goal of XML search is to obtain relevant XML fragments for a query instead of just returning the entire XML documents. Therefore, XML search engines can generate a ranked list composed of a set of relevant XML fragments, while several Web search engines return a set of entire Web documents. By isolating the XML

[4] http://www.w3.org/TR/REC-xml/

fragments, users do not need to identify the relevant XML fragments from the larger XML documents.

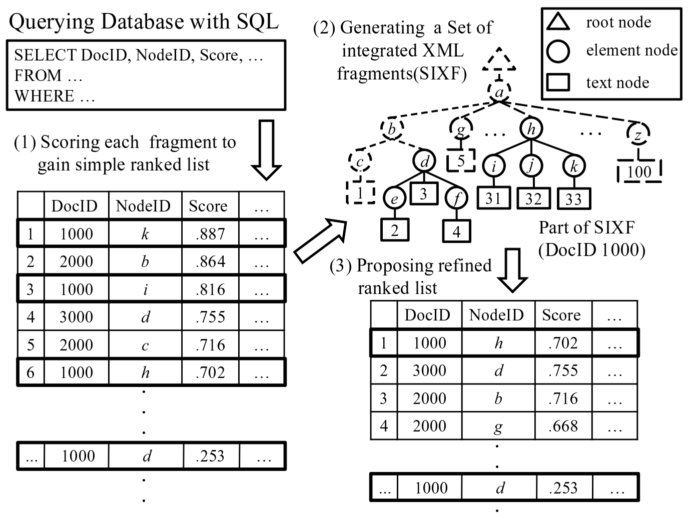
INEX Ad hoc track requires a ranked list which contains relevant XML fragments for a query. In addition to this, the Ad hoc track in 2010 requires very restricted results in the text size. This constraint is reasonable because users should want more focused results.

In our previous studies[5][6], we focused on how we identify the relevant fragments without irrelevant parts in each document. Even though these methods could improve search accuracy, they did not consider to return restricted results because they aimed to extract relevant fragments as much as possible. Therefore, we should modify these methods to return suitable results for the Ad hoc track in 2010.

To tackle this challenge, we propose a method for returning ideal search results. We consider the requirements of meaningful snippets while the previous methods focused on how to identify the appropriate granular fragments, because it returns very small sized texts as search results.

2 Result Reconstruction Method

In this section, we show our previous study[6]. In the study, we generate the *refined ranked list*, which is composed of relevant and informative XML fragments. As shown in Figure 1, the overview of the method is as follows:



Querying Database with SQL
SELECT DocID, NodeID, Score, …
FROM …
WHERE …
(1) Scoring each fragment to gain simple ranked list
(2) Generating a Set of integrated XML fragments(SIXF)
root node
element node
text node
a
b
g
h
z
c
d
i
j
k
1
e
3
f
5
31
32
33
100
2
4
Part of SIXF (DocID 1000)
(3) Proposing refined ranked list

	DocID	NodeID	Score	…
1	1000	*k*	.887	…
2	2000	*b*	.864	…
3	1000	*i*	.816	…
4	3000	*d*	.755	…
5	2000	*c*	.716	…
6	1000	*h*	.702	…
...	1000	*d*	.253	…

	DocID	NodeID	Score	…
1	1000	*h*	.702	…
2	3000	*d*	.755	…
3	2000	*b*	.716	…
4	2000	*g*	.668	…
...	1000	*d*	.253	…

Fig. 1. Overview of the previous study

(1) We first score each XML fragment by using a scoring method to obtain a simple ranked list. We use BM25E[9] to calculate relevance scores of each fragment for a query.
(2) We extract XML fragments from a simple ranked list and generate a set of relevant XML fragments by considering limitations of the XML fragment length and reconstruction of XML fragments.
(3) To present a refined ranked list, we rank the XML fragments extracted from Step (2); we re-rank and remove the XML fragments in the simple ranked list and incorporate them into the refined ranked list.

Hereafter, we denote the set of relevant XML fragments generated in Step (2) as the *Set of Integrated XML Fragments* (*SIXF*). Furthermore, to more readily explain the process of generating the SIXF, we translate each XML fragment into a tree structure. In general, XML documents can be translated into a tree structure by representing tags as nodes in the tree. Each XML fragment in an XML document is associated with a sub-tree in the entire XML tree of the XML document.

2.1 Generating a Set of Integrated XML Fragments

To generate an SIXF, we extract the relevant XML fragments from the simple ranked list generated in Step (1) of our method. Large XML fragments might contain irrelevant fragments and decrease the search accuracy. Therefore, we extract multiple relevant XML fragments from an XML document, as long as their sizes are properly restricted.

One base line approach for generating a nonoverlapped ranked list of XML fragments is to repeatedly extract XML fragments from a simple ranked list in descending order of their rank unless an overlap occurs. The overlapped XML fragments are simply discarded and ignored.

This operation continues as long as either a candidate of the XML fragments remains in the search results or the text size of the extracted XML fragments reaches a predefined upper limit[5].

Next, we aim to reconfigure XML fragments in a simple ranked list to produce results that are better than those of the established base line. We should consider how to identify and extract XML fragments of appropriate lengths in order to attain more accurate XML fragment search. We should also consider how to handle overlapping results, which we ignored in the base line approach.

From the above discussion, we derive the following requirements:

Requirement 1:

Since the traditional search results include several large XML fragments, we should impose an extraction limit on the fragment size.

[5] In our experiments, we extracted 1,000 or less characters for each query. We extract from the beginning of the string value in the text node to the end. When we obtain the first 1,000 characters as a search result, we ignore the rest of the text in the fragment.

Requirement 2:

The extracted XML fragments are appropriately abbreviated and reconstructed to resolve overlap problem.

Extraction Limit To satisfy Requirement 1, we need to limit the size of the extracted XML fragments to an *extraction limit* (EL). Furthermore, we should summarize each XML document, because showing entire documents in search results is not effective. Therefore, we limit the extracted text size for each XML document by defining the EL of an XML document D, whose document ID ($DocID$) is as follows:

$$EL_{DocID} = \alpha \cdot |D_{DocID}| \tag{1}$$

where $|D_{DocID}|$ is the size of the XML document D_{DocID} and α is the ratio of the size of the relevant fragment.

Given this definition, we extract XML fragments from a simple ranked list when the text size of the XML fragments in the SIXF is less than EL. This process repeats until its size exceeds EL.

Reconstructing Fragments To satisfy Requirement 2, we need to arrange the extracted XML fragments so that the SIXF contains useful search results. For generating a non-overlapped ranked list for the base line approach, we simply eliminate the overlapping XML fragments. This may prevent us from extracting relevant XML fragments. For example, in Figure 2, we assume that the XML fragment rooted at node c is the most relevant one in the tree; however, we cannot extract c if we have already extracted d.

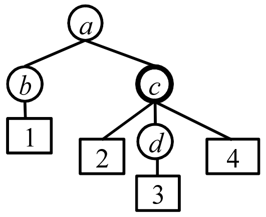


Fig. 2. Example of *overwrite* fragment

To address this problem, we search larger XML fragments and *overwrite* them. As a result, these relevant fragments are all contained in the SIXF, while the existing approaches extract the fragments with the higher score. As these *overwrite* operations are applied, XML fragment lengths in the SIXF increase; therefore, the *overwrite* operation is executed only when Requirement 1 is satisfied.

Again, consider Figure 2. Suppose that c has been extracted. If a is extracted, a overwrites c, because it is larger than c. Furthermore, we discard d, because it is smaller than c.

2.2 Generating a Refined Ranked List

After generating a SIXF, we score each XML fragment in the SIXF and finally generate a refined ranked list. In this process, we aim to obtain informative XML fragments for the higher ranked results. The simplest way to score these XML fragments is to use BM25E, which is used in Step (1) of our method. Unless noted otherwise, we use the BM25E score in the remainder of this paper.

In the following subsections, we propose two scoring methods for generating a refined ranked list: (1) *bottom-up* scoring, which utilizes statistics of descendant XML fragments in order to score an ancestor XML fragment; and (2) *top-down* scoring, which utilizes statistics of an ancestor XML fragment in order to score descendant XML fragments.

Bottom-Up Scoring The *overwrite* operation introduced in Section 2.1 is executed when overlaps exist in the SIXF. In the *overwrite* operation, an ancestor XML fragment is extracted in place of its descendants. The ancestor should be ranked lower in a simple ranked list as compared to its descendants, implying that the refined ranked list also treats the ancestor XML fragment as a lower rank if BM25E is used. As a result, the originally higher-ranked XML fragments cannot always be ranked higher. To score these XML fragments more appropriately, we propose *bottom-up* scoring in order to give more reasonable scores to the XML fragments that contain highly scored descendants.

When we score an XML fragment, we should consider the statistics of its descendant fragments. Conversely, it is not appropriate that XML fragments with low scores are ranked high. Therefore, we must integrate these scores properly. Portions of text nodes in an ancestor XML fragment are composed of descendant XML fragments; the scores of the text nodes in these descendants affect those in the ancestors. Therefore, the *bottom-up* score should be calculated using the BM25E score, as well as the ratio of the lengths of the ancestor XML fragment and that of its descendants.

Let f_a be an ancestor XML fragment and f_d be the descendant fragment with the highest score. We define the *bottom-up* (*BU*) scoring function as

$$s_{BU}(f_a) = \frac{|f_d|}{|f_a|} \cdot s(f_d) + \frac{|f_a| - |f_d|}{|f_a|} \cdot s(f_a) \tag{2}$$

where $|f_d|$ is the length of f_d, $|f_a|$ is the length of f_a, $s_{BU}(f_a)$ is the *bottom-up* score of f_a, $s(f_a)$ is the BM25E score of f_a, and $s(f_d)$ is the BM25E score of f_d.

Top-Down Scoring Since query keywords may have numerous meanings, it is often difficult to identify a proper one. One solution is to consider the co-occurrence of query keywords. If an XML fragment contains several distinct query keywords in its text nodes, we can assume the XML fragment to be closely related to the meaning of the given query keywords. In our previous study[5], we obtained informative XML fragments by considering the number of distinct query keywords in each XML fragment.

The larger XML fragments contain more query keywords, indicating that larger XML fragments tend to be ranked higher. In other words, we might overlook smaller XML fragments, although they are informative. To cope with this problem, we propose a scoring method that is independent of XML fragment size.

XML fragments contain numerous distinct query keywords and are identified as informative. Descendant XML fragments of these informative fragments should also be informative. We, therefore, consider *top-down* scoring by calculating the ratio of the number of distinct query keywords contained in an XML fragment to that of its top-level ancestor—i.e., the entire document.

Let f be a scored XML fragment and D_f be an XML document associated with f. We define the *top-down* (*TD*) scoring function as

$$s_{TD}(f) = s(f) \cdot count(D_f) \tag{3}$$

where $s(f)$ is the BM25E score of f and $count(D_f)$ is the number of distinct query keywords in D_f.

2.3 Example of Generating SIXF and Refined Ranked List

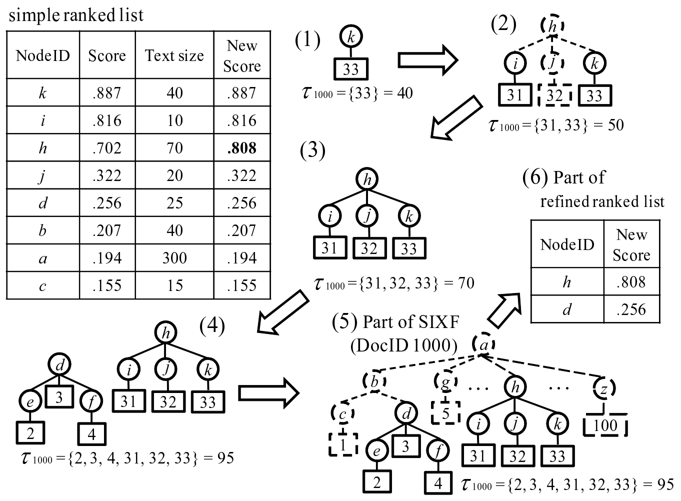
In summary, we illustrate an example of generating a part of an SIXF in which the document ID is 1,000 and its corresponding refined ranked list uses BU scoring. Figure 3 provides a graphical view of this example.

Suppose that $\alpha = \frac{1}{3}$, and $|D_{1,000}| = 300$. Then, $EL_{1,000} = \alpha \cdot |D_{1,000}| = 100$. Next, we introduce $\tau_{1,000}$, which is the total length of the extracted XML fragments from document ID 1,000.

We first obtain a simple ranked list calculated in Step (1). The obtained XML fragments are shown in the left table of Figure 3. For the sake of simplicity, we assume that the list contains only the XML fragments whose document ID is 1,000.

We extract the XML fragment with the highest score, which is node k, from the simple ranked list. Since the text length of k is 40 ($< EL_{1,000}$), k is extracted. This extraction process continues because $\tau_{1,000}$, which contains text node 33, is less than $EL_{1,000}$. Therefore, i is selected next, because i has the second-highest score in the simple ranked list. Thus, i is extracted because $\tau_{1,000}$, which contains text nodes 31 and 33, equals 50 ($< EL_{1,000}$). Node h is the next candidate to be extracted. Since i and k are the descendants of h, they are *overwritten* by h. In particular, i and k are removed from the SIXF and h is added. At this point, $\tau_{1,000}$ becomes 70, which is still less than $EL_{1,000}$.

Next, the *bottom-up* scoring is applied. Equation (2) is used to score node $h(s_{BU}(h)) = \frac{40}{70} \cdot 0.887 + \frac{70-30}{70} \cdot 0.702 = 0.808$.



simple ranked list

NodeID	Score	Text size	New Score
k	.887	40	.887
i	.816	10	.816
h	.702	70	**.808**
j	.322	20	.322
d	.256	25	.256
b	.207	40	.207
a	.194	300	.194
c	.155	15	.155

(6) Part of refined ranked list

NodeID	New Score
h	.808
d	.256

(1)
$\tau_{1000} = \{33\} = 40$
(2)
$\tau_{1000} = \{31, 33\} = 50$
(3)
$\tau_{1000} = \{31, 32, 33\} = 70$
(4)
$\tau_{1000} = \{2, 3, 4, 31, 32, 33\} = 95$
(5) Part of SIXF (DocID 1000)
$\tau_{1000} = \{2, 3, 4, 31, 32, 33\} = 95$

Fig. 3. Processing flow of our method

Following Figure 3 further, d is also extracted. Nodes b and c fail to be extracted because $\tau_{1,000}$ exceeds $EL_{1,000}$. In the end, the SIXF is formed by nodes d and h.

Finally, the refined ranked list is constructed by adding the scores calculated via the *bottom-up* scoring into the SIXF. In the same manner, we generate complete SIXF for all XML documents and construct the final refined ranked list in the descending order of their scores.

2.4 Experimental Evaluation on INEX 2010

We performed our experimental evaluation by using the INEX 2010 test collection. We conducted experiments to compare the previous methods, SIXF,*BU*, and *TD*, with the base line approach.

Note that our experiments show that *EL*, introduced in Section 2.1, does not perform well. As a consequence, we do not apply this approach to the experiments in the remaining subsections.

Table 1 shows that the precisions over the retrieved characters (char_prec) of SIXF are higher than those of the base line (non-overlapped ranked list). *TD* also overwhelms the base line, and *BU* decreases its search accuracy when compared with the base line.

As the result indicates, SIXF is the most effective in these methods, though we reported that *BU* and *TD* are more effective than SIXF when the INEX 2008 test collection is used. Therefore, we should refine *BU* and *TD* scorings if we utilize them for INEX 2010. In next section, we draw a rough sketch of our idea.

Table 1. Comparison of previous methods

	SIXF	*BU*	*TD*	base line
char_prec (INEX 2010)	.3884	.3277	.3565	.3391
iP[.01] (INEX 2008)	.6628	.6653	.6637	.6131

3 Investigation of Previous Methods and Exploration of New Methods

Our previous methods[6] tried to extract slightly wider range of relevant fragments. On the other hand, we should try to identify the fragments which can be strongly relevant for a query because our method returns only small sized contents for INEX 2010. Table 2 shows the average number of extracted fragments of each query. It indicates that only one or a few more fragments per a query are extracted. Therefore, we have to carefully determine which fragment is more suitable as a search result because the search accuracy mainly depends on the top-ranked result.

Table 2. Average Number of Extracted Fragments

	SIXF	*BU*	*TD*	base line
average number of extracted fragments	1.212	1.019	1.192	1.192

This suggests that returning the fragments which are the most relevant for a query, i.e., having the highest score, lead us better search results while Section 2.4 shows different. To be concrete, the base line which returns the fragments with the highest score is less effective than SIXF which *overwrites* fragments and extracts a larger fragment.

However, it does not mean that returning larger granular fragments shows higher search accuracy because *BU* which gives higher score to the larger granular fragments does not overwhelm the base line. In other words, it is not true that search accuracy increases as the granularity of fragments becomes larger[6].

From above discussion, it is more important to reveal the condition of strongly relevant fragments for a query rather than to identify the granularity of them. Therefore, we consider three aspects as follows:

- Two phase extraction
  We find relevant contents after identifying relevant documents. The text size

[6] Actually, the search accuracy of document retrieval is much lower than the base line (char_prec = .2044).

of each fragment affects their score[7]. Therefore, we relax such effect through two steps: 1) identifying a relevant document, and 2) extracting relevant contents to improve search accuracy.

– Management of the range of extraction
  In the step of returning search results in Section 2.4, we extract texts from the beginning of the fragments. However, we can imagine that the range of extracted texts in the fragments influences large effect on search accuracy when we obtain not entire texts in each fragment but part of them. Therefore, we suppose that it is reasonable to consider the condition of each fragment to return appropriate search results.
– Improvement of previous methods
  Some of our previous methods do not perform well as shown in Section 2.4. This is why we try to modify the previous methods along the requirements of INEX 2010. For example, *BU* which utilizes the statistics of descendant fragments emphasizes the ratio of the text sizes of ancestor and descendant fragments. It works well for INEX 2008. However, it does not take effect for INEX 2010 because the granularity of fragments is not so important in 2010. For this reason, we need to refine each scoring method.

4 Related Studies

There are two types of XML documents: (1) data-centric which mainly contain single or compound term in their text nodes and (2) document-centric which tend to contain natural languages in their text nodes[1]. The following subsections describe the existing studies related to both types of XML documents. Although we are primarily interested in search techniques for document-centric XML documents, information on data-centric XML documents will also be useful.

4.1 Data-Centric XML

Data-centric XML documents generally describe only one term in their text nodes. Therefore, studies investigating data-centric XML primarily focus on searching query keywords. The existing research efforts to attain efficient XML fragment search usually utilize the lowest common ancestor (LCA) approach[14]. As a part of this approach, the LCA itself may originate as the top-level common ancestor of arbitrary nodes in an XML tree; however, it is generally defined as the deepest node containing all query keywords in its descendants. Research involving LCA and XML fragments shows significant results related to *efficient* XML search; however, such techniques do not perform well in the context of accurate XML search. In other words, the retrieval accuracy of XML search engines decreases when we use LCAs as the most appropriate XML fragments for a given query[10].

[7] To avoid this effect, BM25E[9] and TF-IPF[8] are normalized by the number of indexed terms.

To address this problem, research efforts have also tried to identify and extract more relevant XML fragments from the sub-tree whose root node is an LCA. XSeek[10] is one such solution, producing a meaningful LCA (MLCA), which classifies and analyzes XML tags by using XML schema information and the positions of query keywords. In the case of XSeek, nodes related to a query are selected and extracted in the order of their relevance to a query. Another approach, eXtract[4], is an expansion of MLCA and infers a user's search purpose by analyzing queries. In eXtract, queries are classified into two cases: (1) extracting an explicit search target and (2) extracting neighbors of the search target.

The purpose of these approaches is similar to that of ours: we wish to return the reconstructed XML fragments. Our method is, however, based on an information retrieval technique for achieving effective XML search, while the abovementioned approaches utilize LCA for the purpose of efficiency.

4.2 Document-Centric XML

Since most document-centric XML documents contain multiple terms in their text nodes, the existing approaches for searching document-centric XML documents focuses on an effective search. In other words, the key objective is to rank more relevant XML fragments higher in the result list. Therefore, conventional information retrieval techniques are often utilized. Another approach is to refine the result list, for example, by removing insignificant fragments or reducing their scores.

Scoring methods for XML fragment search are often derived from the ones used in document search. For example, TF-IPF[2], a popular scoring method for XML fragment search, is an XPath[8]-based scoring method that extends the well-known TF-IDF[13] approach for document search. Another popular approach for XML fragment search is BM25E[9], which is based on Okapi's BM25[12] scoring method for document search.

Although these approaches have been successful, a gap between XML fragment search and document search remains. The goal of XML fragment search is to extract the relevant part of an XML document directly, whereas that of document search is to simply find relevant documents. Extracting relevant XML fragments for a given query is similar to generating snippets in a traditional document search. Snippets, which are short summaries of Web pages that help users judge whether the documents are worth reading, are generated by using information extraction techniques. Since both XML fragments and snippets present useful information to users, such information extraction techniques can be applied to our scoring method.

Manning et al. noted that snippets should have useful information and be maximally informative to a query[11]. To satisfy such requirements, we consider statistics calculated by query conditions[5]; such statistics consist of two components: (1) the ratio of XML fragments containing query keywords amongst XML

[8] `http://www.w3c.org/TR/xpath`

fragments satisfying the constraints of the structure of the given query and (2) the number of query keywords in each XML fragment. In this paper, we denote (1) as the query structure score and (2) as the query keyword score. In our previous study[5], [7], our experiments showed that such methods are more effective than the approaches that do not consider query statistics. We also found that the length of the retrieved XML fragments increases when we use these statistics.

As mentioned above, removing useless or low-scoring fragments is also effective. Although all granularities of XML fragments should be treated as search targets, the effectiveness of the results decreases sharply if a search engine returns noninformative XML fragments. Extremely small XML fragments are often not suitable for search results; Hatano et al. noted that when such meaningless XML fragments are removed, search accuracy improves[3]. Furthermore, our previous study suggests that large XML fragments are also inappropriate for search results[7]. Therefore, we should consider the length of XML fragments.

5 Conclusion

In this paper, we investigate the Ad hoc track of INEX 2010 and drew a rough sketch of our ongoing work. Since our previous methods could not always obtain relevant search results, we should propose a novel approach for returning search results and refine them to adjust to INEX 2010.

References

1. Henk Blanken, Torsten Grabs, Hans-Jörg Schek, Ralf Schenkel, and Gerhard Weikum. *Intelligent Search on XML Data: Applications, Languages, Models, Implementations, and Benchmarks*, volume 2818 of *Lecture Notes on Computer Science*. Springer-Verlag, September 2003.
2. Torsten Grabs and Hans-Jörg Schek. PowerDB-XML: A Platform for Data-Centric and Document-Centric XML Processing. In *Proceedings of the First International XML Database Symposium*, volume 2824 of *Lecture Notes on Computer Science*, pages 100–117. Springer Berlin, September 2003.
3. Kenji Hatano, Hiroko Kinutani, Masahiro Watanabe, Yasuhiro Mori, Masatoshi Yoshikawa, and Shunsuke Uemura. Keyword-based XML Portion Retrieval: Experimental Evaluation based on INEX 2003 Relevance Assessments. In *Proceedings of the Second Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 81–88, March 2004.
4. Yu Huang, Ziyang Liu, and Yi Chen. Query Biased Snippet Generation in XML Search. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 315–326. ACM, June/July 2008.
5. Atsushi Keyaki, Kenji Hatano, and Jun Miyazaki. A Query-oriented XML Fragment Search Approach on A Relational Database System. *Journal of Digital Information Management (JDIM)*, 8(3):175–180, June 2010.
6. Atsushi Keyaki, Kenji Hatano, and Jun Miyazaki. Result Reconstruction Approach for More Effective XML Fragment Search. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services (iiWAS2010)*, pages 115–123, November 2010.

7. Atsushi Keyaki, Jun Miyazaki, and Kenji Hatano. A Method of Generating Answer XML Fragment from Ranked Results. In *INEX 2009 Workshop Pre-Proceedings*, pages 563–574, December 2009.
8. Fang Liu, Clement Yu, Weiyi Meng, and Abdur Chowdhury. Effective Keyword search in Relational Databases. In *Proceedings of the 2006 ACM SIGMOD international Conference on Management of Data*, pages 563–574. ACM, June 2006.
9. Wei Liu, Stephen Robertson, and Andrew Macfarlane. Field-Weighted XML Retrieval Based on BM25. In *Advances in XML Information Retrieval and Evaluation*, volume 3977 of *Lecture Notes on Computer Science*, pages 161–171. Springer Berlin, June 2006.
10. Ziyang Liu and Yi Chen. Identifying Meaningful Return Information for XML Keyword Search. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 329–340. ACM, June 2007.
11. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*, pages 157–159. Cambridge University Press, July 2008.
12. Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. *The Third Text REtrieval Conference (TREC-3)*, pages 109–126, 1995.
13. Gerard Salton and Christopher Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Journal of Information Processing and Management*, 24(5):513–523, January 1988.
14. Albrecht Schmidt, Martin Kersten, and Menzo Windhouwer. Querying XML Documents Made Easy: Nearest Concept Queries. In *Proceedings of the 17th International Conference on Data Engineering*, page 321. IEEE, April 2001.

Searching the Wikipedia with public online search engines

Miro Lehtonen

Department of Computer Science
P. O. Box 68 (Gustaf Hällströmin katu 2b)
FI–00014 University of Helsinki
Finland
Miro.Lehtonen@cs.helsinki.fi

Abstract. Commercial search engines were put to a test as we searched the online Wikipedia which is a newer version of the INEX 2010 document collection. Although the INEX 2010 ad hoc search tasks and the search features of the public search engines are not 100% compatible, we were able to compare and evaluate the search results of online search engines with INEX 2010 topics, assessments, and metrics. Considering the first page of results, we cannot see a big difference between the performance of the best academic search engines and the best commercial ones.

1 Introduction

Public online search engines have not been seen in the official INEX evaluations in the past despite the highly competitive performance that they offer to web users. Whether they are competitive also when measured in such a standard IR evaluation has thus not been reported before. The purpose of this experiment was to find out how well some of the most popular search engines fare with the academic search engines and with each other. The search engines of choice are Google, Bing, Yahoo!, and the default Wikipedia search which specialises in searching the online version of the Wikipedia[1].

The comparison is not completely fair for a number of reasons. For one thing, the Wikipedia articles evolve constantly. Not only is the online Wikipedia different from the INEX version, but each online search engine indexes a slightly different version of the document collection, as well. For another thing, none of the search engines return 1500 results per query. It is possible to set a limit on the results per page, but ultimately, the number of retrieved results depends on the query. Sometimes the search engines do not return any results, which can be considered more user-friendly than returning nearly 1500 non-relevant results. User satisfaction is however not taken into account by all the metrics. Nevertheless, the results should be indicative of the true performance, given the variety of different metrics and appropriate interpretation of the results.

[1] This experiment has not been endorsed by any of the mentioned search engines.

Initial analysis of the results shows that the best academic search engines are just as competitive as the best commercial search engines when all the circumstances are taken into account, e.g. we only search the Wikipedia.

2 Running online search engines

The tests described in this section were conducted in August 2010. Although the results for individual queries may change over time as the search engines index and re-index updated pages, the changes should not affect the overall performance or the mutual rankings of the search engines.

All the search engines were used in a uniform fashion. One HTTP request was made for each INEX 2010 topic and for each search engine, after which the server response — the first page of results — was dumped into a file for further analysis and processing. The maximum number of results on the first page naturally depends on the search engine. The URLs for the HTTP requests came from entering the title field of the topic in the search box as written in the topic file. Quotations marks, plus and minus signs were all included as such.

2.1 Google

Of the all the different Google search sites, the one that is not specific to any country was chosen[2] although the rankings might be stable across different country versions when searching a single site. The maximum number of results per page was set to 100.

Most of the results that Google returns also exist in the INEX version of the Wikipedia. The retrieved pages that are not found in the INEX collection are either new articles or combinations of old ones with a new article ID.

2.2 Bing

Bing seems to assume that different rankings are called for in different countries — even when searching a single site such as the English Wikipedia. Therefore, choosing a country version for this experiment makes a real difference. American bing[3] was chosen for no other reason than the assumption that, as one of the most frequently used country versions of Bing, it should be up-to-date and the rankings should be rather stable. The maximum number of results on the first page was set to 50 which is the biggest number allowed.

Unlike Google, Bing retrieves a fair amount of pages that are not included in the INEX collection although the content is. For example, one of the pages that Bing returns has the title of "Marilyn Munroe" and the URL

`http://en.wikipedia.org/wiki/Marilyn_Munroe`.

The page redirects to a page correctly titled "Marilyn Monroe" which does exist in the INEX collection and which is retrieved by Google as a link to

[2] www.google.com/ncr

[3] www.bing.com/search?setmkt=en-US&q=...

`http://en.wikipedia.org/wiki/Marilyn_Monroe`.
The contents of these two pages are equivalent, but, if the page is relevant, only Google scores whereas Bing hits a missing page as it does not retrieve the same content twice. This might be a case where Bing is being unfairly penalized for retrieving pages with relevant content but a misspelled title.

2.3 Yahoo!

Yahoo[4] returns a maximum of 100 results on the first page with a note "Powered by Bing". The exact reliance on Bing is somewhat unclear but one of the prolems is the same: Yahoo too retrieves a number of redirected pages which are not included in the INEX version of Wikipedia.

2.4 Wikisearch

The on site Wikipedia search engine[5] is the only search engine in this experiment that returns focused results. While other search engines retrieve whole articles from the Wikipedia, the Wikipedia search engine also suggests which section might be relevant to the query. However, this feature of Wikisearch was ignored because the anchor of the section under focus cannot be converted into an entry point without opening the INEX version of the article, and even then the conversion is not completely reliable.

Wikisearch allows a total of 500 results to be shown on the first page. Like Google, the on site search does not return any pages that redirect further, so most of the retrieved articles also exist in the INEX verion of the Wikipedia.

3 From result pages to run submissions

Run submissions were created for two different tasks: Restricted Focused (RF) and Restricted Relevant in Context (RRIC). Processing the first page of results began the same way for both tasks. First, the article titles were collected by scanning the result page and extracting the title from the URL. Second, the corresponding pages were found in the INEX collection by matching the titles of the online article to the titles in the INEX 2009 version of the Wikipedia. Because the actual online article was not accessed, the reason for not finding a matching article in the INEX collection was not analysed. Once we had a ranked list of articles for each topic, we could create a task-specific run submission.

3.1 Restricted Focused

The ranked list of articles which was specific to each search engine did not contain any focused results. Therefore, the focus had to be artificially added to the result

[4] search.yahoo.com
[5] http://en.wikipedia.org/w/index.php?title=Special:Search&search=...

list. It had to be a blind process because the document and element scores of the search engines were not accessible and because we wanted to eliminate the effect of all external factors. A simple but heuristic way to meet the task requirement of 1000 characters per topic was to pick the top two articles from the list and return a passage consisting of the first 500 characters of each.

3.2 Restricted Relevant in Context

Creating a run submission for the RRIC task was straightforward. All of the articles on the first page of results were included. Restricting the results to 500 characters per article was a task requirement. Because none of the search engines provide ways to define such a restriction, a simple heuristic had to be defined for all of them. Assuming that the beginning of the article would be the best entry point, the first 500 characters of the article would be a good guess on the restricted passage. Retrieving 500 characters from the beginning of the article was also simple to implement.

The Wikipedia search is the only search engine where the first 500 characters is not always part of the result retrieved by the online search engine because Wikipedia search sometimes focuses the results to certain sections. In those cases, the real Wikipedia might get better scores than it gets in this experiment.

3.3 Summary

All the search engines are good at removing duplicate pages from the results so that the same content is not retrieved multiple times although it may exist under several different URLs. How many pages each search engine retrieved that also exist in the INEX Wikipedia collection is summarised in Table 1.

Search engine	First page	Total RF	Total RRIC	Max RRIC
Google	100	212	7,353	10,700
Bing	50	208	1,156	5,350
Yahoo!	100	209	5,893	10,700
Wikisearch	500	202	28,770	53,500

Table 1. Summary of submitted results for the 107 topics of INEX 2010.

There were a total of 107 topics in 2010, so the maximum number of submitted results for the RF task would be 214 and for the RRIC task 160,500 (1500 results per query), given the chosen heuristics. The number of results that was actually retrieved is bigger the number of submitted results because of new pages and redirecting pages.

4 Results

The evaluation for the runs submitted for the RF task is shown in Table 2. Although Google seems to retrieve relevant articles with the highest precision, Yahoo retrieves the highest ratio of relevant content to non-relevant content.

Search engine	art_prec	char_prec
Google	0.7596	0.3276
Yahoo	0.7404	0.3435
Bing	0.7212	0.3354
Wiki	0.6538	0.0047

Table 2. Runs submitted for the restricted focused task evaluated.

5 Conclusion

Four public online search engines were tested when searching the Wikipedia with the INEX 2010 topics. Google, Bing, Yahoo, and the on site search of Wikipedia all retrieve relevant articles from the Wikipedia with varying success.

Extended Language Models for XML Element Retrieval

Rongmei Li and Theo van der Weide

Radboud University, Nijmegen, The Netherlands

Abstract. In this paper we describe our participation in the INEX 2010 ad-hoc track. We participated in three retrieval tasks (restricted focused task, relevant-in-context, restricted relevant-in-context) and report our findings based on a single set of measure for all tasks. In this year's participation, we evaluate the performance of the standard language model that is more focused on a fixed number of relevant characters. Our findings are: 1) the simplest language model for document retrieval performs relatively good in the restricted focused task when using a fixed offset that is close to the average character distance from the beginning of a document to its main content; 2) a good result of document ranking does improve the performance of snippet retrieval.

1 Introduction

INEX offers a framework for cross comparison among content-oriented XML retrieval approaches given the same test collections and evaluation measures. The INEX ad-hoc track is to evaluate system performance in retrieving relevant document components (e.g. XML elements or passages) for a given topic of request. The relevant results should discuss the topic exhaustively and have as little non-relevant information as possible (specific for the topic). This year the retrieved components are restricted to a fixed number of characters as a form of snippet. Additionally, the system efficiency is evaluated. The ad-hoc track includes four retrieval tasks: the Restricted Focused task, the Relevant in Context task, the Restricted Relevant in Context task, and the system efficiency.

The 2010 collection is the same English Wikipedia as in 2009 with XML format. The ad-hoc topics are created by the participants to represent real life information need. Same to 2009, each topic consists of five fields. The `<title>` field (CO query) is the same as the standard keyword query. The `<castitle>` field (CAS query) adds structural constraints to the CO query by explicitly specifying where to look and what to return. The `<phrasetitle>` field (Phrase query) presents explicitly a marked up query phrase. The `<description>` and `<narrative>` fields provide more information about topical context. Especially the `<narrative>` field is used for relevance assessment.

Following our last year's work [1], the paper documents our primary and official results in the INEX 2010 ad-hoc track. Our aims are to: 1) evaluate the performance of standard IR engines (Indri search engine) used in full document retrieval and snippet retrieval; 2) investigate possible improvement techniques. We adopt the language modeling approach [2] and tailor the estimate of query term generation from a document to generation from an XML element according to the user request. The retrieval results are evaluated as: 1) focused (snippet) retrieval; 2) full document retrieval.

The rest of the paper describes our experiments in the ad-hoc track. The pre-processing and indexing steps are given in section 2. Section 3 explains how to convert a user query to an Indri structured query. The retrieval model and strategies are summarized in section 4. We present our results and analysis in section 5 and conclude this paper in section 6.

2 Pre-processing and Indexing

The original English XML Wikipedia is not stopped or stemmed before indexing. The 2010 collection has 2,666,190 documents taken on 8 October 2008. It is annotated with the 2008-w40-2 version of YAGO ([3]).

We index only the queried XML fields out of all presented XML fields as follows:

category, actor, actress, adversity, aircraft, alchemist, article, artifact, bdy, bicycle, caption, catastrophe, categories, chemist, classical_music, conflict, director, dog, driver, group, facility, figure, film_festival, food, home, image, information, language, link, misfortune, mission, missions, movie, museum, music_genre, occupation, opera, orchestra, p, performer, person, personality, physicist, politics, political_party, protest, revolution, scientist, sec, section, series, singer, site, song, st, theory, title, vehicles, village.

3 Query Formulation

We handle CO and CAS queries by full document retrieval while ignoring boolean operators (e.g. "-"or "+") in `<title>` and `<castitle>` fields. Additionally, we remove terms like "of", "or", "and", "in", "the", "from", "on", "by", and "for" from both. We extract all `<castitle>` terms within "about". Both types of queries use the Indri belief operator #combine [4].

4 Retrieval Model and Strategies

We first retrieve full articles using either CO or CAS queries. The retrieval model is based on cross-entropy scores between the query model and the document model that is smoothed using Dirichlet priors [2]. It is defined as follows:

$$score(D|Q) = \sum_{i=1}^{l} P_{ml}(t_i|\theta_Q) \cdot \log\left(\frac{tf(t_i, D) + \mu P_{ml}(t_i|\theta_C)}{|D| + \mu}\right) \quad (1)$$

where l is the length of the query, $P_{ml}(t_i|\theta_Q)$ and $P_{ml}(t_i|\theta_C)$ are the Maximum Likelihood (ML) estimates of respectively the query model θ_Q and the collection model θ_C. $tf(t_i, D)$ is the frequency of query term t_i in a document D. $|D|$ is the document length. μ is the smoothing parameter.

We set up our language model and model parameters based on the experimental results of similar tasks for INEX 2008. Here μ is considered to be 500.

4.1 Full Document Retrieval

Baseline runs retrieve full documents for CO or CAS queries. Only the #combine operator is used. Additional run uses the provided reference run as the final rank of retrieved documents for CO queries. The reference run uses BM25 ranking function with $k1 = 1.2$ and $b = 0.3$. It is stemmed by a simple s-stemmer. These parameters are learned using the INEX 2009 topics and assessments on the INEX 2009 Wikipedia collection using the Mean uninterpolated Average Precision (MAP) metric.

We submitted 9 results for three tasks. We do not have result on the system efficiency as our focus is paid on the effectiveness of retrieval model only.

4.2 Snippet Retrieval

Within the ad-hoc retrieval track, there are three considered sub-tasks that are a form of snippet retrieval:

- The `Restricted Focused task` is a variant of the Focused Task where the results to maximum 1,000 characters per topic is allowed. Retrieved elements are ranked by relevance and are not clustered by article. Overlapping among elements is not allowed.
- The `Relevant in Context (RiC) task` requires the system to return relevant elements or passages clustered per article. Within each article, reading order of retrieved element matters. The retrieval of non-relevant text is strongly penalized. Overlapping within article is not permitted.
- The `Restricted Relevant in Context task` is similar to RiC but only allow 500 characters per article to be retrieved.

Empirically the main content of an article does not start from the very beginning, namely at offset zero. We take the average offset from the main content that is assumed to be 20 characters for two restricted retrieval tasks.

5 Results

For each of the three sub-tasks, we submitted two XML article results for CO and CAS queries and one extra result using the reference run as the final rank of the CO query result. On the whole, we had 9 submissions and qualified runs to the ad-hoc track.

5.1 Measured as Document Retrieval

When measured as document retrieval, our reference based runs outperform our baseline runs in all sub-tasks in terms of MAP score. The results of CO queries perform better than the results of CAS queries in all sub-tasks. Compared to other participanting groups, our best run ranks 9 with MAP of 0.2593. However, the MAP scores for our baselines of CO and CAS queries are only 0.0243 and 0.0226 respectively.

The split performance overview in sub-tasks are summerized in Table 1. The total evaluated runs from participating groups are 34, 65, and 39 following the top-down order of sub-tasks in the Table. Our simple approach for restricted focused document retrieval is relatively effecitive, even when there is no help of the reference run.

Table 1. Measured as document retrieval

performance metrics	submitted runs for restricted focused		
	runFocCORef	runFocusCO10	runFCASart10
MAP	0.2593	0.0243	0.0226
rank in all runs (34)	6	7	8
	submitted runs for relevance in context		
	runRiCORef	runRiCCO10	runRiCCAS10
MAP	0.2593	0.0243	0.0226
rank in all runs (65)	32	61	62
	submitted runs for restricted relevance in context		
	runReRiCORef	runReRicCO10	runRRicCAS10
MAP	0.2593	0.0243	0.0226
rank in all runs (39)	14	35	36

5.2 Measured as Snippet Retrieval

When measured as the thorough task, our results for the RiC task ranks the first than our other results followed by results of the restricted focused task in terms of MAiP score. Our reference based runs are still better than their counterparts in each sub-task accordingly. Our best run as the thorough task ranks 3 among participating groups and 14 in all 217 evaluated runs with MAiP of 0.2230.

The performance overview of sub-tasks of snippet retrieval is represented seperately in the following sub-sections.

Restricted Focused The restricted focused task is to return to the user a ranked list of snippets with a maximum length of 1000 characters. Our results start from the 21th character of the retrieved relevant documents. The overall performance of our submissions is shown in Table 2. Our best run is based on the given reference with the score of char_prec 0.3361. It ranks 5 among participating groups and 8 in all 34 evaluated runs in terms of the score of character precision (char_prec). Our other two runs rank 12 and 13 in all runs with close char_prec score to our best run. The performance of all our runs drops quickly with the increase of recall. This is also true for the results of other groups. The detailed information is shown in Table 2.

Table 2. Measured as focused retrieval

runs for restricted focused	performance metrics					
	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP	char_prec
runFocCORef	0.3361	0.0964	0.0435	0.0000	0.0067	0.3361
runFCASart10	0.3241	0.0820	0.0357	0.0000	0.0061	0.3241
runFocusCO10	0.3237	0.0756	0.0357	0.0000	0.0059	0.3237

Relevant in Context The RiC task has the same goal as year 2009. It is to return the relevant information within the full article. Our baseline runs complete this task as document retrieval. Only the reference based run is officially evaluated and the result is presented in Table 3. Our reference based run ranks 6 among participating groups and 20 in all 47 evaluated runs with the score of MAgP 0.1377 (see Table 3).

Table 3. Measured as focused retrieval

runs for relevance in context	performance metrics				
	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
runRiCORef	0.2642	0.2310	0.1694	0.1431	0.1377

Restricted Relevant in Context The Restricted RiC task limits the result of the RiC task to be 500 characters at most. Similar to the restricted focused task, our results start from the 21 character of the baseline document retrieval. Again only the reference based run is officially evaluated and its performance is summerized in Table 4. It ranks 9 among participating groups and 12 in all 24 evaluated runs with the score of MAgP 0.1375 (see Table 4)

Table 4. Measured as focused retrieval

runs for restricted relevance in context	performance metrics				
	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
runReRiCORef	0.2641	0.2313	0.1686	0.1428	0.1375

5.3 Analysis

In general, BM25 ranking function generates better results than our simplest language model, which has no stopping and stemming. Our baseline runs perform relatively good in the restricted focused task. The result of CAS queries is better than that of CO queries in this task as it contains more query terms, thus more context information.

There are 52 topics used for evaluation in all snippet retrieval. 11 out of 52 is to return XML elements such as “sec”, “p”, and “person”. The rest is the normal document retrieval with restriction on a fixed number of characters. When a topic is about any XML element type (noted as *), we may also accept the whole document.

For possible performance improvement, it is important to identify the difficult topics that bring down the average measure score. The number of topics that have lower measure score than the average is presented in row 1 of the Table 5. The number of XML element topics is in row 2 while the total number of XML element topics is 11. Though our retrieval model also fails on document topics, there is slightly more percentage loss on XML element topics.

Table 5. Number of Topics with Measure Score Less than Average

	runFocCORef	runFCASart10	runFocusCO10	runRiCORef	runReRiCORef
all type	33	33	32	32	32
XML element	7	6	6	8	8

The number of topics that have zero measure score at average are 27 for all restricted focused runs (the left three columns in Table 5), 3 for the RiC run (the column 4), and 4 for the restricted RiC run (the right column).

Particularly, the most difficult topics that gain the least measure score in all tasks are topic 19, topic 31, and topic 107. Their contents are as follows:

topic 19:
```
<title>gallo roman architecture in paris</title>
<castitle>//*[about(.,gallo roman architecture in paris)]</castitle>
```
topic 31:
```
<title>science women few</title>
<castitle>//*[about(., science women few)]</castitle>
```
topic 107:
```
<title>factors determining human height</title>
<castitle>//article[about(., human)]//p[about(., factors determining
height)]</castitle>
```

The content of these topics does not differ much for other topics. Further investigation is needed to their ground truth.

6 Conclusion

In this paper, we present our results for the ad-hoc track of INEX 2010. Our offical runs contain two baseline runs, namely document retrieval for CO queries and CAS queries as a form of snippet retrieval. Additionally, we generate a reference based run. Our basline runs perform relatively good among participating groups in the restricted focused retrieval. Our performance loss happens on both document topics and XML element topics. The reference run does provide better document ranking, which in turn improve the performance of our baselines. Since our baseline runs are the simplest language models, our on-going experiments will remove the stop words and general terms in addition to the stemming techniques. The un-offical results will be in our final report.

References

1. Li, R.M., Weide, Th.P. van der: Language Models for XML Element Retrieval, In *Proceedings of INEX* (2009)
2. Zhai, C.X., Lafferty, J.: A Study of Smoothing Methods for Language Models Applied to Information Retrieval. ACM Trans. on Information Systems. 22(2), 179–214 (2004)
3. Schenkel, R., Suchanek, F.M., Kasneci, G.: YAWN: A Semantically Annotated Wikipedia XML Corpus. In *12. GI-Fachtagung fr Datenbanksysteme in Business, Technologie und Web*, Aachen, Germany. (March 2007)
4. Strohman, T., Metzler, D., Turtle, H., Croft, W.B.: Indri: A Language-model Based Search Engine for Complex Queries, In *Proceedings of ICIA* (2005)

Overview of the INEX 2010 Book Track: At the Mercy of Crowdsourcing

Gabriella Kazai[1], Marijn Koolen[2], Antoine Doucet[3], and Monica Landoni[4]

[1] Microsoft Research, United Kingdom
v-gabkaz@microsoft.com
[2] University of Amsterdam, Netherlands
m.h.a.koolen@uva.nl
[3] University of Caen, France
doucet@info.unicaen.fr
[4] University of Lugano
monica.landoni@unisi.ch

Abstract. The goal of the INEX 2010 Book Track is to evaluate approaches for supporting users in reading, searching, and navigating the full texts of digitized books. The investigation is focused around four tasks: 1) the Book Retrieval (Best Books to Reference) task aims at comparing traditional and book-specific retrieval approaches, 2) the Focused Book Search (Prove It) task evaluates focused retrieval approaches for searching books, 3) the Structure Extraction task tests automatic techniques for deriving structure from OCR and layout information, and 4) the Active Reading task aims to explore suitable user interfaces for eBooks enabling reading, annotation, review, and summary across multiple books. We report on the setup and the results of the track.

1 Introduction

The INEX Book Track was launched in 2007, prompted by the availability of large collections of digitized books resulting from various mass-digitization projects [1], such as the Million Book project[5] and the Google Books Library project[6]. The unprecedented scale of these efforts, the unique characteristics of the digitized material, as well as the unexplored possibilities of user interactions present exciting research challenges and opportunities, see e.g. [4].

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

– Information retrieval techniques for searching collections of digitized books,

[5] http://www.ulib.org/
[6] http://books.google.com/

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

Based around these main themes, the following four tasks were defined:

1. The Best Books to Referencel (BB) task, framed within the user task of building a reading list for a given topic of interest, aims at comparing traditional document retrieval methods with domain-specific techniques, exploiting book-specific features, e.g., back-of-book index, or associated metadata, e.g., library catalogue information,
2. The Prove It (PI) task aims to test the value of applying focused retrieval approaches to books, where users expect to be pointed directly to relevant book parts,
3. The Structure Extraction (SE) task aims at evaluating automatic techniques for deriving structure from OCR and layout information for building hyperlinked table of contents, and
4. The Active Reading task (ART) aims to explore suitable user interfaces enabling reading, annotation, review, and summary across multiple books.

In this paper, we report on the setup and the results of each of these tasks at INEX 2010. First, in Section 2, we give a brief summary of the participating organisations. In Section 3, we describe the corpus of books that forms the basis of the test collection. The following three sections detail the four tasks: Section 4 summarises the two search tasks (BR and FBS), Section 5 reviews the SE task, and Section 6 discusses ART. We close in Section 7 with a summary and plans for INEX 2010.

2 Participating Organisations

A total of 82 organisations registered for the track (compared with 84 in 2009, 54 in 2008, and 27 in 2007). As of the time of writing, we counted 10 active groups (compared with 16 in 2009, 15 in 2008, and 9 in 2007), see Table 1.

3 The Book Corpus

The track builds on a collection of 50,239 out-of-copyright books[7], digitized by Microsoft. The corpus is made up of books of different genre, including history books, biographies, literary studies, religious texts and teachings, reference works, encyclopedias, essays, proceedings, novels, and poetry. 50,099 of the books also come with an associated MAchine-Readable Cataloging (MARC) record, which contains publication (author, title, etc.) and classification information. Each book in the corpus is identified by a 16 character long bookID – the name of the directory that contains the book's OCR file, e.g., A1CD363253B0F403.

[7] Also available from the Internet Archive (although in a different XML format)

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

ID	Institute	Topics	Runs	Judged topics (book/page level)
6	University of Amsterdam	19-20,22	2 BB, 4 PI	
7	Oslo University College	02-06	5 PI	
14	Uni. of California, Berkeley		4 BB	
54	Microsoft Research Cambridge	00-01,07-09,24-25		
86	University of Lugano	15-18,21 ,23		
98	University of Avignon		9 BB, 1 PI	
386	University of Tokyo			
662	Izmir Institute of Technology			
663	IIIT-H	10-14		
732	Wuhan University			

The OCR text of the books has been converted from the original DjVu format to an XML format referred to as BookML, developed by Microsoft Development Center Serbia. BookML provides additional structure information, including markup for table of contents entries. The basic XML structure of a typical book in BookML is a sequence of pages containing nested structures of regions, sections, lines, and words, most of them with associated coordinate information, defining the position of a bounding rectangle ([coords]):

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

BookML provides a set of labels (as attributes) indicating structure information in the full text of a book and additional marker elements for more complex structures, such as a table of contents. For example, the first label attribute in the XML extract above signals the start of a new chapter on page 1 (label="PT_CHAPTER"). Other semantic units include headers (SEC_HEADER), footers (SEC_FOOTER), back-of-book index (SEC_INDEX), table of contents (SEC_TOC). Marker elements provide detailed markup, e.g., for table of contents, indicating entry titles (TOC_TITLE), and page numbers (TOC_CH_PN), etc.

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

4 Information Retrieval Tasks

Focusing on IR challenges, two search tasks were investigated: 1) Best Books to Reference (BB), and 2) Prove It (PI). Both these tasks used the corpus described in Section 3, and shared the same set of topics (see Section 4.3).

4.1 The Best Books to Reference (BB) Task

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

Participants of this task were invited to submit either single runs or pairs of runs. A total of 10 runs could be submitted, each run containing the results for all the 83 topics (see Section 4.3). A single run could be the result of either a generic (non-specific) or a book-specific IR approach. A pair of runs had to contain both types, where the non-specific run served as a baseline, which the book-specific run extended upon by exploiting book-specific features (e.g., back-of-book index, citation statistics, book reviews, etc.) or specifically tuned methods. One automatic run (i.e., using only the topic title part of a topic for searching and without any human intervention) was compulsory. A run could contain, for each topic, a maximum of only100 books (identified by their bookID), ranked in order of estimated relevance.

A total of 15 runs were submitted by 3 groups (2 runs by University of Amsterdam (ID=6); 4 runs by University of California, Berkeley (ID=14); and 9 runs by the University of Avignon (ID=98)), see Table 1.

4.2 The Prove It (PI) Task

The goal of this task was to investigate the application of focused retrieval approaches to a collection of digitized books. The scenario underlying this task is that of a user searching for specific information in a library of books that can provide evidence to confirm or reject a given factual statement. Users are assumed to view the ranked list of book parts, moving from the top of the list down, examining each result. No browsing is considered (only the returned book parts are viewed by users).

Participants could submit up to 10 runs. Each run could contain, for each of the 83 topics (see Section 4.3), a maximum of 1,000 book pages estimated relevant to the given aspect, ordered by decreasing value of relevance.

A total of 10 runs were submitted by 3 groups (4 runs by the University of Amsterdam (ID=6); 5 runs by Oslo University College (ID=7); and 1 run by the University of Avignon (ID=98)), see Table 1.

4.3 Topics

This year we explored the use of Amazon's Mechanical Turk (AMT) service to aid in the creation of topics for the test collection. This is motivated by the need to scale up the Cranfield method for constructing test collections where the significant effort required to create test topics and to collect relevance judgements is otherwise inhibiting. By harnessing the collective work of the crowds, crowdsourcing offers an increasingly popular alternative for gathering large amounts of data feasibly, at a relatively low cost and in a relatively short time. We are interested in using crowdsourcing to contribute to the building of a test collection for the Book Track, which has thus far struggled to meet this requirement by relying on its participants' efforts alone.

With this aim, we experimented with gathering topics both through Amazon's Mechanical Service and from the track participants. Our aim was to compare the quality of the collected topics and assess the feasibility of crowdsourcing topics (and relevance judgements later on). To this end, we first redefined the search tasks, simplifying them in order to make topic creation for them suitable as a Human Intelligent Task (HIT).

As mentioned already, in the Prove It task systems need to find evidence in books that can be used to either confirm or refute a factual statement given as the topic. In the Best Books task systems need to return the most relevant books on the general subject area of the topic. To collect the test topics for the two tasks, we created the following two HITs:

– Facts in books HIT (Book HIT): "Your task is to fnd a general knowledge fact that you believe is true in a book available at http://booksearch.org.uk. Both the fact and the book must be in English. The fact should not be longer than a sentence. For example, the fact that 'The first Electric Railway in London was opened in 1890 and run between the stations: Bank and Stockwell' can be found on page 187 of the book titled 'West London' by George Bosworth". Workers were asked to record the factual statement they found, the URL of the book containing the fact, and the page number.
– Facts in books and Wikipedia HIT (Wiki HIT): "Your task is to find a general knowledge fact that appears BOTH in a Wikipedia article AND in a

book available at http://booksearch.org.uk. You can start either by finding a fact on Wikipedia first, then locating the same fact in a book, or you can start by finding a fact in a book and then in Wikipedia. For example, the Wikipedia page on Beethoven's Symphony No. 3 claims that 'Beethoven dedicated the symphony to Napoleon, but when Napoleon proclaimed himself emperor, Beethoven tore up the title'. Page 144 of the book titled Beethoven by Romain Rolland describes this very fact". Workers needed to record the factual statement, the URL and page number of the book where the fact is found, as well as the Wikipedia article's URL.

We created 10 Wiki HITs, paying \$0.25 per HIT, and issued two batches of Book HITs, with 50 HITs in each batch, paying \$0.10 per HIT in the first batch and \$0.20 in the second batch. All 10 Wiki HITs were completed within a day, while only 32 Fact HITs were completed in 11 days out of the first batch. The second batch of 50 Book HITs was completed fully in 14 days. The average time required per Book HIT was 8 minutes in the first batch and 7 minutes in the second batch (hourly rate of \$0.73 and \$1.63, respectively), while Wiki HITs took on average 11 minutes to complete (hourly rate of \$1.31). These statistics suggest that workers found the Wikipedia task more interesting, despite it taking longer. However, as we show later, the attractiveness of a HIT does not guarantee good quality topics.

At the same time, INEX participants were asked to create 5 topics each, 2 of which had to contain factual statements that appears both in a book and in Wikipedia. A total of 25 topics were submitted by 5 groups. Of these, 16 facts appear both in books and in Wikipedia.

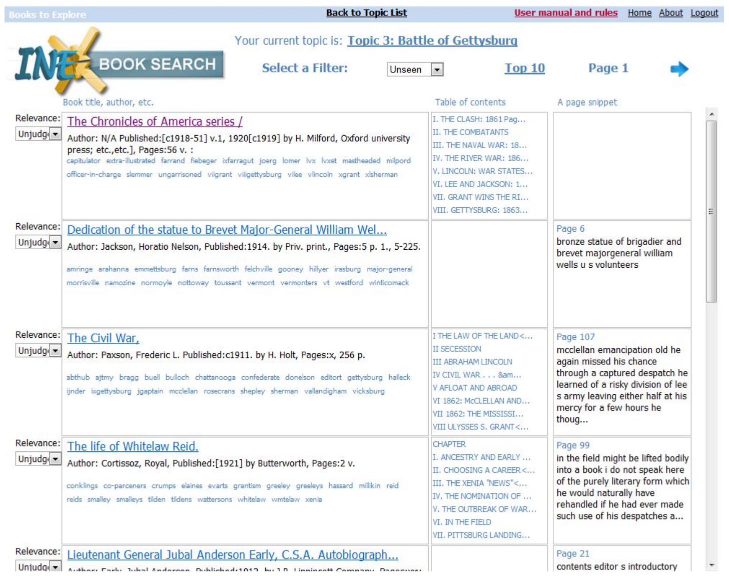
All collected topics were carefully reviewed and those judged suitable were selected into the set of test topics that is currently being used by the INEX Book Track. All topics contributed by INEX participants were selected, while filtering was necessary for topics created by AMT workers. Out of the 10 Wiki HITs, only 4 topics were selected. Of the 32 Book HITs in the first batch, 18 were acceptable, while 36 were selected from the 50 Book HITs in the second batch. HITs were rejected for a number of reasons: the information given was simply an extract from a book, rather than a fact (20), the fact was too specialised (5), or nonsensical (5), the HIT had missing data (3), or the worker submitted the example given in the task description (1). Of the total 58 accepted HITs, 18 had to be modified, either to rephrase slightly or to correct a date or name, or to add additional information. The remaining 40 HITs were high quality and reflecting real interest or information need.

From the above, it seems clear that crowdsourcing provides a suitable way to scale up test collection construction: MTurk workers contributed 58 topics, while INEX participants created only 25 topics. However, the quality of crowdsourced topics varies greatly and thus requires extra effort to weed out unsuitable submissions. We note that selecting workers based on their approval rate had a positive effect on quality: batch 2 of the Book HITs required workers to have a HIT approval rate of 95%. In addition, paying workers more also shows correlation with the resulting quality.

4.4 Relevance Assessment System

The Book Search System (http://www.booksearch.org.uk), developed at Microsoft Research Cambridge, is an online tool that allows participants to search, browse, read, and annotate the books of the test corpus. Annotation includes the assignment of book and page level relevance labels and recording book and page level notes or comments. The system supports the creation of topics for the test collection and the collection of relevance assessments. Screenshots of the relevance assessment module are shown in Figures 1 and 2.

In preparation for the relevance gathering stage, which will run in parallel, collecting judgements from INEX participants and from AMT workers, we simplified the assessment process from previous years.



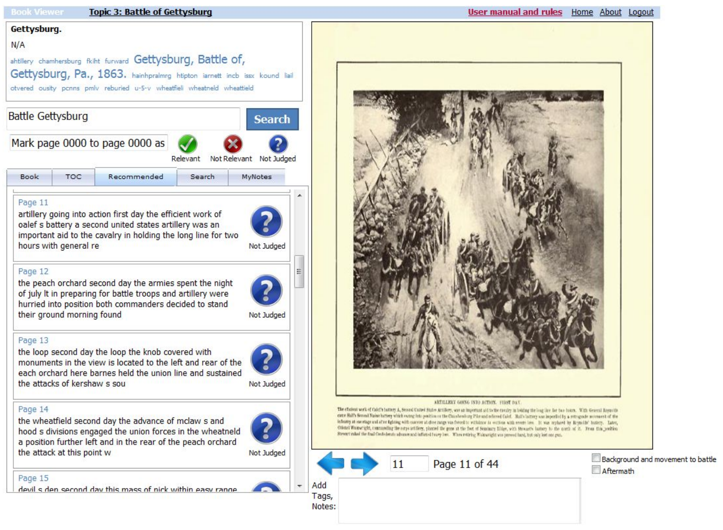
Your current topic is: Topic 3: Battle of Gettysburg

	Book title, author, etc.	Table of contents	A page snippet
Relevance: Unjudg	The Chronicles of America series / Author: N/A Published:[c1918-51] v.1, 1920[c1919] by H. Milford, Oxford university press; etc.,etc.], Pages:56 v. : capitulator extra-illustrated farrand fiebeger ixfarragut joerg lomer lvx lvxet mastheaded milpord officer-in-charge slemmer ungarrisoned viigrant viiigettysburg vilee vlincoln xgrant xlsherman	I. THE CLASH: 1861 Pag... II. THE COMBATANTS III. THE NAVAL WAR: 18... IV. THE RIVER WAR: 186... V. LINCOLN: WAR STATES... VI. LEE AND JACKSON: 1... VII. GRANT WINS THE RI... VIII. GETTYSBURG: 1863...	
Relevance: Unjudg	Dedication of the statue to Brevet Major-General William Wel... Author: Jackson, Horatio Nelson, Published:1914. by Priv. print., Pages:5 p. 1., 5-225. amringe arahanna emmettsburg farns farnsworth felchville gooney hillyer irasburg major-general morrisville namozine normoyle nottoway toussant vermont vermonters vt westford winticomack		Page 6 bronze statue of brigadier and brevet majorgeneral william wells u s volunteers
Relevance: Unjudg	The Civil War, Author: Paxson, Frederic L. Published:c1911. by H. Holt, Pages:x, 256 p. abthub ajtmy bragg buell bulloch chattanooga confederate donelson editort gettysburg halleck ijnder ixgettysburg jgaptain mcclellan rosecrans shepley sherman vallandigham vicksburg	I THE LAW OF THE LAND<... II SECESSION III ABRAHAM LINCOLN IV CIVIL WAR . . . &am... V AFLOAT AND ABROAD VI 1862: McCLELLAN AND... VII 1862: THE MISSISSI... VIII ULYSSES S. GRANT<...	Page 107 mcclellan emancipation old he again missed his chance through a captured despatch he learned of a risky division of lee s army leaving either half at his mercy for a few hours he thoug...
Relevance: Unjudg	The life of Whitelaw Reid. Author: Cortissoz, Royal, Published:[1921] by Butterworth, Pages:2 v. conklings co-parceners crumps elaines evarts grantism greeley greeleys hassard millikin reid reids smalley smalleys tilden tildens wattersons whitelaw wmtelaw xenia	CHAPTER I. ANCESTRY AND EARLY ... II. CHOOSING A CAREER<... III. THE XENIA "NEWS"<... IV. THE NOMINATION OF ... V. THE OUTBREAK OF WAR... VI. IN THE FIELD VII. PITTSBURG LANDING...	Page 99 in the field might be lifted bodily into a book i do not speak here of the purely literary form which he would naturally have rehandled if he had ever made such use of his despatches a...
Relevance: Unjudg	Lieutenant General Jubal Anderson Early, C.S.A. Autobiograph...		Page 21 contents editor s introductory

Fig. 1. Screenshot of the relevance assessment module of the Book Search System, showing the list of books in the assessment pool for a selected topic in game 1. For each book, its metadata, its table of contents (if any) and a snippet from a recommended page is shown.

4.5 Collected Relevance Assessments

This is still in progress at the time of writing.



Gettysburg.
N/A
ahtillery chamhersburg fkiht furward Gettysburg, Battle of, Gettysburg, Pa., 1863. hainhpralmrg htipton iarnett incb issx kound liail otvered ousity pcnns pmlv reburied u-5-v wheatfieli wheatneld wheattield

Battle Gettysburg

Mark page 0000 to page 0000 as

Page 11
artillery going into action first day the efficient work of oalef s battery a second united states artillery was an important aid to the cavalry in holding the long line for two hours with general re
Not Judged

Page 12
the peach orchard second day the armies spent the night of july lt in preparing for battle troops and artillery were hurried into position both commanders decided to stand their ground morning found
Not Judged

Page 13
the loop second day the loop the knob covered with monuments in the view is located to the left and rear of the each orchard here barnes held the union line and sustained the attacks of kershaw s sou
Not Judged

Page 14
the wheatfield second day the advance of mclaw s and hood s divisions engaged the union forces in the wheatneld a position further left and in the rear of the peach orchard the attack at this point w
Not Judged

Page 15
devil s den second day this mass of nick within easy range

11

☐ Background and movement to battle
☐ Aftermath

Fig. 2. Screenshot of the relevance assessment module of the Book Search System, showing the Book Viewer window with Recommended tab listing the pooled pages to judge with respect to topic aspects in game 2. The topic aspects are shown below the page images.

4.6 Evaluation Measures and Results

We will report on these once sufficient amount of relevance labels have been collected.

5 The Structure Extraction (SE) Task

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

The 2010 task was run as a follow-up of the conjoint INEX and ICDAR 2009 competition [2,3]. Participants were able to refine their approaches with the help of the groundtruth built in 2009.

Only one institution, the University of Caen, participated in this rerun of the 2009 task, while 2 contributed to the extension of the groundtruth data, since the University of Firenze joined the effort. The groundtruth now covers an additional 114 books and reaches a total of 641 annotated ToCs.

The performance of the 2010 run is given in Table 2 . A summary of the performance of the 2009 runs with the extended 2010 ground truth data is given in Table 3.

	Precision	Recall	F-measure
Titles	18.03%	12.53%	12.33%
Levels	13.29%	9.60%	9.34%
Links	14.89%	7.84%	7.86%
Complete except depth	14.89%	10.17%	10.37%
Complete entries	10.89%	7.84%	**4.86%**

Table 2. Score sheet of the run submitted by the University of Caen during the 2010 rerun of the SE competition 2009

RunID	Participant	F-measure (2010)	F-measure (2009)
MDCS	MDCS	43.39%	41.51%
XRCE-run2	XRCE	28.15%	28.47%
XRCE-run1	XRCE	27.52%	27.72%
XRCE-run3	XRCE	26.89%	27.33%
Noopsis	Noopsis	8.31%	8.32%
GREYC-run1	University of Caen	0.09%	0.08%
GREYC-run2	University of Caen	0.09%	0.08%
GREYC-run3	University of Caen	0.09%	0.08%

Table 3. Summary of performance scores for the 2009 runs with the extended 2010 groundtruth-rerun; results are for complete entries.

Naturally, the small increase in the size of the groundtruth does not make the results vary much (most of the groundtruth data was built for the 2009 experiments: 527 out 641 annotated books).

6 The Active Reading Task (ART)

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

ART is based on the evaluation experience of EBONI [5], and adopts its evaluation framework with the aim to guide participants in organising and running user studies whose results could then be compared.

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

6.1 Task Setup

Participation requires access to one or more software/hardware e-readers (already on the market or in prototype version) that can be fed with a subset of the INEX book corpus (maximum 100 books), selected based on participants' needs and objectives. Participants are asked to involve a minimum sample of 15/20 users to complete 3-5 growing complexity tasks and fill in a customised version of the EBONI subjective questionnaire, allowing to gather meaningful and comparable evidence. Additional user tasks and different methods for gathering feedback (e.g., video capture) may be added optionally. A crib sheet is provided to participants as a tool to define the user tasks to evaluate, providing a narrative describing the scenario(s) of use for the books in context, including factors affecting user performance, e.g., motivation, type of content, styles of reading, accessibility, location and personal preferences.

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

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

7 Conclusions and plans

For the evaluation of our two search tasks (best books and prove it), we are currently collecting relevance assessments from INEX participants. This will be used as gold set for collecting judgements from workers on Amazon's Mechanical

Turk (AMT). Results will then be distributed around February 2011. The ART and SE tasks were offered as last year.

This year the focused search task (prove it) was based on factual statements for which systems were asked to find book pages that either confirmed or refuted the fact. 70

The SE task was run (though not advertised), using the same data set as last year. One institution participated and contributed additional annotations.

Unless we get a LOT more ACTIVE participants, 2011 will probably be the last year of the book track. We hope that with the burden of topic creation and relevance assessments removed, we will however get higher participation next year. We also plan to re-shape the research agenda by significantly increasing the size of the collection on the one hand, and by defining more challenging tasks that are focused on user interaction on the other hand, placing the ART in the centre.

References

1. Karen Coyle. Mass digitization of books. *Journal of Academic Librarianship*, 32(6):641–645, 2006.
2. Antoine Doucet, Gabriella Kazai, Bodin Dresevic, Aleksandar Uzelac, Bogdan Radakovic, and Nikola Todic. ICDAR 2009 Book Structure Extraction Competition. In *Proceedings of the Tenth International Conference on Document Analysis and Recognition (ICDAR'2009)*, pages 1408–1412, Barcelona, Spain, july 2009.
3. Antoine Doucet, Gabriella Kazai, Bodin Dresevic, Aleksandar Uzelac, Bogdan Radakovic, and Nikola Todic. Setting up a competition framework for the evaluation of structure extraction from ocr-ed books. *International Journal on Document Analysis and Recognition*, pages 1–8, 2010.
4. 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.
5. Ruth Wilson, Monica Landoni, and Forbes Gibb. The web experiments in electronic textbook design. *Journal of Documentation*, 59(4):454–477, 2003.

The Book Structure Extraction Competition with the Resurgence software for part and chapter detection

Emmanuel Giguet*, Nadine Lucas*

GREYC Cnrs, Caen Basse Normandie University
BP 5186 F-14032 CAEN Cedex France
* name.surname@info.unicaen.fr

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

1. Introduction

The GREYC Island team participated for the second time in the Book Structure Extraction Competition part of the INEX evaluations [1]. The Resurgence software used at Caen University to structure various documents was modified to this end. This software processes academic articles (mainly in pdf format) and news articles (mainly in HTML format) in various text parsing tasks [2]. The team was also interested in handling voluminous documents, such as textbooks and cultural heritage books, hence the interest in INEX.

The experiment was conducted from pdf documents to ensure the control of the entire process. The document content is extracted using the pdf2xml software [3]. In the first experiment, we handled only the chapter level [4]. We still could not propagate our principles on all the levels of the book hierarchy at a time. We consequently focused on the higher levels of book structure, part and chapter detection.

In the following, we explain our strategy though the results on the INEX book corpus are still unknown at the time of submission. We provide estimated results. In the last section, we discuss the advantages of our method and make proposals for future competitions.

2. Our Book Structure Extraction Method

2.1. Challenges

In the first experiment, the huge memory needed to handle books was found to be indeed a serious hindrance, as compared with the ease in handling academic articles: pdf2xml required up to 8 Gb of memory and Resurgence required up to 2 Gb to parse the content of large books (> 150 Mb). This was due to the fact that the whole content of the book was stored in memory. The underlying algorithms did not actually require the availability of the whole content at a time. Resurgence was modified in order to load the necessary pages only. The objective was to allow processing on usual laptop computers.

The fact that the corpus was OCR documents also challenged our previous program that detected the structure of electronic academic articles. A new branch in Resurgence had to be written in order to deal with scanned documents. We propagated our document parsing principles on two levels of the book hierarchy at a time, part (meaning her part including a number of chapters) and chapter, hoping for an improvement of the results, but two levels proved insufficient to boost the quality.

2.2. Strategy

The strategy in Resurgence is based on document positional representation, and does not rely on the table of contents (ToC). This means that the whole document is considered first. Then document constituents are considered top-down (by successive subdivision), with focus on the middle part (main body) of the book. The document is thus the unit that can be broken down ultimately to pages. The main idea is to use a model describing relationships for elements in the document structure. The model is a periphery-center dichotomy. The periphery center relationship is calculated on the entire document and reflected on each page. The algorithm aims at retrieving the book main content bounded by annex material like preface and postface with different layout. It ultimately retrieves the page body in a page, surrounded by margins [2].

Implementation rules

For this experiment, we focused on part (if any) and chapter title detection so that the program detects only two levels, i. e. part titles and chapter titles.

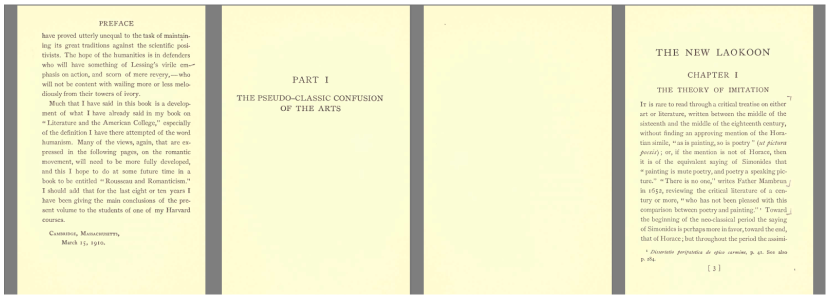
Part wrapping chapters are detected throughout the document using a sliding window of one page. The idea is to detect a page with few written lines. The transition page between two parts is characterized as follows:

the text body in the page is mainly blank,

- with a blank at least 5 times the standard line space height;
- followed by 1 to 3 written lines;
- with a blank at least 5 times the standard line space height.

A global test checks if there are at least two successive parts in the book.

Figure 1 illustrates the pattern. It applies on a single page. 3 context pages are given but are not used in the process.



PREFACE

have proved utterly unequal to the task of maintaining its great traditions against the scientific positivists. The hope of the humanities is in defenders who will have something of Lessing's virile emphasis on action, and scorn of mere revery,—who will not be content with wailing more or less melodiously from their towers of ivory.

Much that I have said in this book is a development of what I have already said in my book on "Literature and the American College," especially of the definition I have there attempted of the word humanism. Many of the views, again, that are expressed in the following pages, on the romantic movement, will need to be more fully developed, and this I hope to do at some future time in a book to be entitled "Rousseau and Romanticism." I should add that for the last eight or ten years I have been giving the main conclusions of the present volume to the students of one of my Harvard courses.

Cambridge, Massachusetts,
March 15, 1910.

PART I

THE PSEUDO-CLASSIC CONFUSION OF THE ARTS

THE NEW LAOKOON

CHAPTER I

THE THEORY OF IMITATION

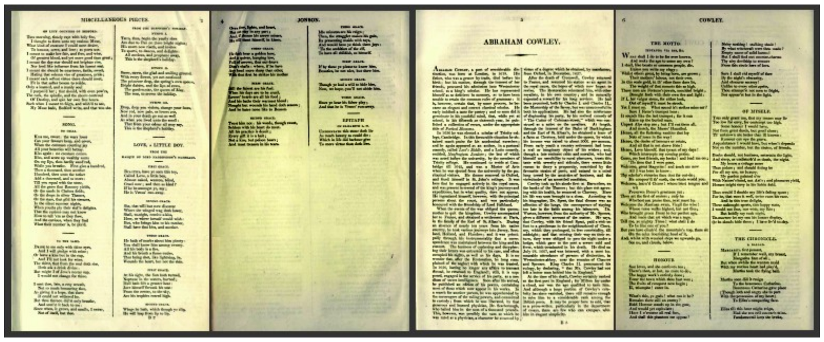
It is rare to read through a critical treatise on either art or literature, written between the middle of the sixteenth and the middle of the eighteenth century, without finding an approving mention of the Horatian simile, "as is painting, so is poetry" (*ut pictura poesis*); or, if the mention is not of Horace, then it is of the equivalent saying of Simonides that "painting is mute poetry, and poetry a speaking picture." "There is no one," writes Father Mambrun in 1652, reviewing the critical literature of a century or more, "who has not been pleased with this comparison between poetry and painting."[1] Toward the beginning of the neo-classical period the saying of Simonides is perhaps more in favor, toward the end, that of Horace; but throughout the period the assimi-

[1] *Dissertatio peripatetica de epico carmine,* p. 41. See also p. 284.

[3]

Fig. 1. View of the one-page sliding window to detect parts beginning. 3 context pages (1 page before, two pages after) are given but not used. Excerpt from 2009 book id = 2A5029E027B7427C

Chapter title detection throughout the document was conducted using a sliding window to detect chapter transitions with two patterns, as explained in [4].



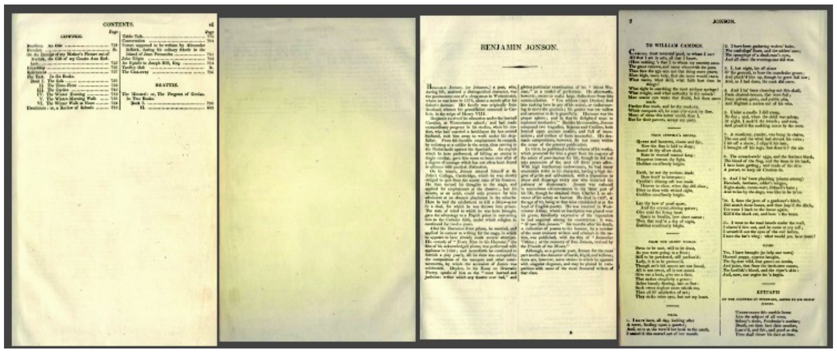
MISCELLANEOUS PIECES.

JONSON.

ABRAHAM COWLEY.

COWLEY.

Fig. 2. View of the four-page sliding window to detect chapter beginning. Pattern 1 matches. Excerpt from 2009 book id = 00AF1EE1CC79B277



CONTENTS.

BEATTIE.

BENJAMIN JONSON.

JONSON.

TO WILLIAM CAMDEN.

EPITAPH

Fig. 3. View of the four-page sliding window to detect chapter announced by a blank page. Pattern 2 matches. Excerpt from 2009 book id= 00AF1EE1CC79B277

Chapter title extraction is made from the first third of the top of the page body. The model assumes that the title begins at the top of the page. The title right end is detected, by calculating the line height disruption: a contrast between the would-be title line height and the rest of the page line height. A constraint rule allows a number of lines containing at most 40 words.

2.3. Calibrating the system

Working on the whole document requires the ability to detect and deal with possible heterogeneous layouts in different parts of the document (preface, main body, appendices). Layout changes can impact page formatting (e.g., margin sizes, column numbers) as well as text formatting (e.g., font sizes, text alignments).

The standard page structure recognition has been improved, by correcting a bug in the previous program that impaired the recognition of page header and footer [4]. It has also been improved by a better recognition of the shape of the body which is not always rectangular in scanned books.

Line detection, standard line height and standard space height detection were also improved. They are important in our approach, because the standard line is the background against which salient features such as large blanks and title lines can be detected. The improvements in line computation improved the results in chapter detection.

The standard line height and standard line space height are computed in the following way. The most frequent representative intervals are computed to cope with OCR variation in line height.

In the previous experiment, line height was calculated somewhat rigidly, after pdf2xml was used. Line recognition was dependent on bounding box heights. However, this is not very reliable for scanned text, and the program tended to create loose line segments. Moreover it also tended to artificially augment the line height, due to the presence of one capital letter for example, and thus the standard line was not contrasted against title lines, which are slightly bigger.

In the current experiment, the model drives the detection process. This means that unless there is a strong clue against it, the line is considered as continue. The line common characteristics are favored against occasional disruptions in bounding box height.

2.3. Experiment

The program detected only part and chapter titles. No effort was exerted to find the sub-titles. There was only one run.

2.4. Expected Results

The entire corpus was handled. The official results should be improved compared with the first official evaluation. However, the very bad results were due to a bug in page numbers. Corrected results were as given in table 2 [4]. The 2010 results should slightly outperform the corrected 2009 results. This is mainly due to improvements in the system calibration. Little gain should be obtained from part detection. This is mainly due to the fact that the evaluation favors title subsection detection, which is not addressed in this participation.

3. Discussion

We were the only candidates this year. The official results are expected as deceptive, although small corrections significantly improved them, as already explained [4]. The low recall will still be due to the fact that the hierarchy of titles was not addressed as mentioned earlier. This will be addressed in the future.

3.1. Reflections on the experiment

On the scientific side, some strong points of the Resurgence program, based on relative position and differential principles, were better implemented. We intend to further explore this way. The advantages are the following:

- The program deals with the entire document, not only the table of contents;
- It handles books without ToCs, and titles that are not represented in the ToC (e. g. preface);
- It is dependent on typographical position, which is very stable in the corpus;
- It is not dependent on lexicon, hence tolerant to OCR errors and language independent.
- Last, it is simple and fast.

Since no list of expected and memorized forms is used, but position instead, fairly common strings are extracted, such as CHAPTER or SECTION, but also uncommon ones, such as PSALM or SONNET. When chapters have no numbering and no prefix such as *chapter*, they are found as well, for instance a plain title “Christmas Day”.

Resurgence did not rely on numbering of chapters: this is an important source of OCR errors. Hence they were retrieved as they were by our robust extractor.

The approach reflects an original breakthrough to improve robustness.

3.2. Proposals

Concerning evaluation rules, generally speaking, it is unclear whether the ground truth depends on the book or on the ToC. If the ToC is the reference, it is an error to extract prefaces, for instance. The participants using the whole text as main reference would be penalized if they extract the whole hierarchy of titles as it appears in the book, when the ToC represents only higher levels, as is often the case.

Concerning details, it should be clear whether or when the prefix indicating the book hierarchy level (*Chapter*, *Section*, and so on) and the numbering should be part of the extracted result. The chapter title is not necessarily preceded by such mentions, but in other cases there is no specific chapter title and only a number. The ground truth is not clear either on the extracted title case: sometimes the case differs in the ToC and in the actual title in the book.

It would be very useful to provide results by title depth (level) as suggested by [5, 6], because it seems that providing complete results for one or more level(s) would be more satisfying than missing some items at all levels. It is important to get coherent and comparable text spans for many tasks, such as indexing, helping navigation or text mining.

The reason why the beginning and end of the titles are overrepresented in the evaluation scores is not clear and a more straightforward edit distance for extracted titles should be provided.

There is also a bias introduced by a semi-automatically constructed ground truth. Manual annotation is still to be conducted to improve the ground truth quality, but it is time-consuming.

It might be a good idea to give the bounding box containing the title as a reference for the ground truth. This solution would solve conflicts between manual annotation and automatic annotation, leaving man or machine to read and interpret the content of the bounding box. It would also alleviate conflicts between ToC-based or text-based approaches.

The corpus provided for the INEX Book track is very valuable, it is the only available corpus offering full books [7]. Although it comprises old printed books only, it is interesting for it provides various examples of layout.

References

1. Doucet, A. and Kazai, G. ICDAR 2009 Book Structure Extraction Competition. *10th International Conference on Document Analysis and Recognition ICDAR 2009*, Barcelona, Spain, IEEE. pp. 1408-1412. (2009).
2. Giguet, E., Lucas, N. & Chircu, C. *Le projet Resurgence: Recouvrement de la structure logique des documents électroniques*. JEP-TALN-RECITAL'08 Avignon (2008).
3. pdf2xml open source software, Déjean, H. last visited March 2010 http://sourceforge.net/projects/pdf2xml/
4. Giguet, E., Lucas, N. The Book Structure Extraction Competition with the Resurgence software at Caen University. In: S. Geva, J. Kamps and A. Trotman Eds. *Focused Retrieval and Evaluation*, LNCS 6203/2010, pp. 170-178, doi: 10.1007/978-3-642-14556-8_18. (2010).
5. Déjean, H. and Meunier, J.-L. "XRCE Participation to the Book Structure Task" in *Advances in Focused Retrieval*, Berlin, Springer. LNCS 5631/2009, pp. 124-131. doi: 10.1007/978-3-642-03761-0 (2009).
6. Déjean, H. and Meunier, J.-L. Reflections on the INEX structure extraction competition. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems* (DAS '10). ACM, New York, NY, USA, 301-308. DOI=10.1145/1815330.1815369 http://doi.acm.org/10.1145/1815330.1815369 (2010).

7. Gabriella Kazai, Antoine Doucet, Marijn Koolen, Monica Landoni. Overview of the INEX 2009 Book Track. *Advances in Focused Retrieval: 8th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2009*, Springer, Lecture Notes in Computer Science, Vol. 6203/2010. pp.145-159. (2010).

University of Amsterdam at INEX 2010: Ad hoc and Book Tracks

Jaap Kamps[1,2] and Marijn Koolen[1]

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

Abstract. In this paper we describe our participation in INEX 2010 in the Ad Hoc Track and the Book Track. In the Ad Hoc track we investigate the impact of propagated anchor-text on article level precision and the impact of an element length prior on the within-document precision and recall. Using the article ranking of an document level run for both document and focused retrieval techniques, we find that focused retrieval techniques clearly outperform document retrieval, especially for the Focused and Restricted Relevant in Context Tasks, which limit the amount of text than can be returned per topic and per article respectively. Somewhat surprisingly, an element length prior increases within-document precision even when we restrict the amount of retrieved text to only 1000 characters per topic. The query-independent evidence of the length prior can help locate elements with a large fraction of relevant text. For the Book Track we look at the relative impact of retrieval units based on whole books, individual pages and multiple pages.

1 Introduction

In this paper, we describe our participation in the INEX 2010 Ad Hoc and Book Tracks. Our aims for the Ad Hoc Track this year were to investigate the impact of an element length prior on the trade-off between within-document precision and recall. In previous years we merged article and element level runs—using the article ranking of the article run and the element run to select the text to retrieve in those articles—and found that this can improve performance compared to individual article and element retrieval runs. But how much text should we retrieve per article?

For the Book Track we look at the relative impact of books, individual pages, and multiple pages as units of retrieval for the Best Books and Prove It Tasks.

The rest of the paper is organised as follows. Then, in Section 2, we report our runs and results for the Ad Hoc Track. Section 3 briefly discusses our Book Track experiments. Finally, in Section 4, we discuss our findings and draw preliminary conclusions.

2 Ad Hoc Track

For the INEX 2010 Ad Hoc Track we aim to investigate:

- The effectiveness of anchor-text for focused ad hoc retrieval. Anchor-text can improve early precision in Web retrieval [8], which might be beneficial for focused retrieval in Wikipedia as well. The new Focused and Restricted Relevant in Context Tasks put large emphasis on (early) precision.
- The relation between element length and within-document precision and recall. With the new tasks restricting systems to return only a limited number of characters per article (Restricted Relevant in Context Task) or per topic (Focused Task), an element length prior might be less effective, as it increases the chances of retrieving irrelevant text.

We will first describe our indexing and retrieval approach, then the official runs, and finally per task, we present and discuss our results.

2.1 Indexing

In this section we describe the index that is used for our runs in the ad hoc track. We used Indri [14] for indexing and retrieval. Our indexing approach is based on earlier work [1–3, 11–13].

- *Section index*: We used the `<section>` element to cut up each article in sections and indexed each section as a retrievable unit. Some articles have a leading paragraph not contained in any `<section>` element. These leading paragraphs, contained in `<p>` elements are also indexed as retrievable units. The resulting index contains no overlapping elements.
- *Article index*: We also build an index containing all full-text articles (i.e., all wikipages) as is standard in IR.
- *Anchor text index*: For this index we concatenated all propagated anchor text of an article as a single anchor text representation for that article.

For all indexes, stop-words were removed, and terms were stemmed using the Krovetz stemmer. Queries are processed similar to the documents. This year we only used the CO queries for the official runs.

2.2 Category Evidence

Based on previous experiments, we used category distance scores as extra evidence for ranking [9]. We determine two target categories for a query based on the top 20 results. We select the two most frequent categories to which the top 20 results are assigned and compute a category distance score using parsimonious language models of each category. This technique was successfully employed on the INEX 2007 Ad hoc topics by Kaptein et al. [6] and on the larger INEX 2009 collection [10] with two sets of category labels [9]; one based on the *Wikipedia* category structure and one based on the *WordNet* category labels. Koolen et al. [9] found that the labels of the original Wikipedia category structure are more effective for ad hoc retrieval. In our experiments, we use the original Wikipedia category labels.

2.3 Runs

Combining the methods described in the previous section with our baseline runs leads to the following official runs.

Article an article index run with length prior ($\lambda = 0.85$ and $\beta = 1$).
ArticleRF an article index run with length prior ($\lambda = 0.85$ and $\beta = 1$) and relevance feedback (top 50 terms from top 10 results).
Anchor anchor text index run without length prior ($\lambda = 0.85$ and $\beta = 0$).
AnchorLen anchor text index run with length prior ($\lambda = 0.85$ and $\beta = 1$).
Sec a section index run without length prior ($\lambda = 0.85$ and $\beta = 0$).
SecLen a section index run with length prior ($\lambda = 0.85$ and $\beta = 1$).

From these initial runs we have constructed our baseline runs:

Base the ArticleRF combined with the category scores based on the 2 most frequent categories of the top 20 results.
Base Sec the Baseline run where an article is replaced by the sections of that article retrieved by the Sec run. If no sections for that article are retrieved, the full article is used.
Fusion a linear combination of the ArticleRF and the AnchorLen runs with weight $S(d) = 0.7 ArticleRF(d) + 0.3 AnchorLen(d)$. The combined run is used to compute category scores based on the 2 most frequent categories of the top 20 results, which are then combined with the merged article and anchor text scores.
Fusion Sec the Fusion run where an article is replaced by the sections of that article retrieved by the Sec run. If no sections for that article are retrieved, the full article is used.

For the Focused Task, we submitted two runs:

Base Sec F1000 Topic : The Base Sec run with only the first 1000 characters retrieved for each topic.
Base Sec F100 Article : The Base Sec run with only the first 100 characters retrieved per article, cut-off after 1000 characters retrieved for each topic.

With the first 1000 characters retrieved, we expect to return only very few documents per topic. With a restriction of at most N characters per document, we can control the minimum number of documents returned, thereby increasing the possible number of relevant documents returned. Both runs have the retrieved sections grouped per article, with the sections ordered according to the retrieval score of the Sec run. That is, if sections s1, s2 and s3 of document d1 are retrieved by the Sec run in the order (s2, s3, s1), then after grouping, s2 is still returned first, then s3 and then s1. The first 1000 characters retrieved will come mostly from a single document (the highest ranked document). With a limit of 100 characters per article, the first 1000 characters will come from at least 10 documents. Although precision among the first 10 documents will probably be lower than precision at rank 1, the larger number of retrieved documents might give the user access to more relevant documents. We will look at the set-based precision of the 1000 characters retrieved as well as the article-based precision and the number of retrieved and relevant retrieved articles.

For the Relevant in Context Task, we submitted two runs:

Base SecLen : the baseline run Base SecLen described above, cut off after the first 1500 results.

Fusion Sec : the baseline run Fusion Sec described above, cut off after the first 1500 results.

The Base and Fusion runs will allow us to see the impact of using propagated anchor-text for early precision.

For the Restricted Relevant in Context Task, we submitted two runs:

Base F500 Article : the Base run reduced to the first 500 characters per retrieved article, and cut off after the first 1500 results.

Base Sec F500 Article : the Base Sec run reduced to the first 500 characters per retrieved article, and cut off after the first 1500 results.

Article retrieval is a competitive alternative to element retrieval when it comes to focused retrieval in Wikipedia [2, 4]. The full per-article recall of article retrieval makes up for its lack in focus. However, for the Restricted Relevant in Context Task, the amount of text retrieved per article is limited to 500 characters, which reduces the high impact of full within-document recall and puts more emphasis on achieving high precision. Relevant articles tend to have relevant text near the start of the article [5], which could give fair precision with the first 500 characters of an article. On the other hand, using the more focused evidence of the section index on the same article ranking, we can select the first 500 characters of the most promising elements of the article. With a restricted number of characters per article, and therefore restricted recall, we expect to a see a clearer advantage in using focused retrieval techniques.

We discovered an error in the baseline runs, which caused our official runs to have very low scores. In the next sections, we show results for both the officially submitted runs and the corrected runs.

2.4 Thorough Evaluation

We first look at the performance of the baseline runs using the Thorough interpolated precision measure. Results can be found in Table 1. We make the following observations:

- The Fusion run is less effective than the Base run. The anchor text does not help early precision.
- The length prior on the sections increases recall for the cost of a slight drop in early precision.
- The focused runs have a lower MAiP but a higher early precision than the article level runs. The article level runs have a much higher recall, and thereby score better on average precision. But the focused runs retrieve less irrelevant text and score better on early precision.

Table 1: Interpolated precision scores of the baseline runs (runs in italics are official submissions, runs with an asteriks are the corrected versions)

Run id	**MAiP**	**iP[0.00]**	**iP[0.01]**	**iP[0.05]**	**iP[0.10]**
Base	**0.2139**	0.4398	0.4219	0.3810	0.3577
Fusion	0.1823	0.4001	0.3894	0.3370	0.3189
Base Sec	0.1555	**0.5669**	**0.5130**	0.4039	0.3600
*Base SecLen	0.1702	0.5507	0.5100	**0.4162**	**0.3784**
*Fusion Sec	0.1317	0.5447	0.4632	0.3439	0.2967
Base SecLen	0.0723	0.3308	0.2910	0.2184	0.1944
Fusion Sec	0.0678	0.3027	0.2694	0.2110	0.1906

Table 2: Results for the Ad Hoc Track Focused Task (runs in italics are official submissions, runs with an asteriks are the corrected versions)

Run id	**# ret.**	**# rel. ret.**	$P_{article}$	P_{char}	**iP[0.00]**	**iP[0.01]**
Base Sec F1000 Topic	65	20	0.3301	0.1232	0.1694	0.0386
Base Sec F100 Article	529	165	0.3105	0.1162	0.2468	0.0338
*Base Sec F1000 Topic	1.29	0.81	**0.6250**	0.3490	0.4012	0.1376
*Base Sec F100 Article	**10.06**	**5.27**	0.5229	0.2445	0.4626	0.1140
Base SecLen F1000 Topic	1.29	0.81	0.6186	**0.3526**	0.3903	**0.1518**
Base SecLen F100 Article	**10.06**	**5.27**	0.5229	0.2677	**0.5015**	0.1226
Base F1000 Topic	1.10	0.69	**0.6250**	0.2806	0.2828	0.0737
Base F100 Article	10.00	5.23	0.5231	0.1415	0.2623	0.0340

2.5 Focused Task

We have no overlapping elements in our indexes, so no overlap filtering is done. Table 2 shows the results for the Focused Task. We make the following observations:

– The first 1000 retrieved characters gives higher precision than first 100 per article up to 1000 characters. But restricting each article to 100 characters, many more articles, including relevant articles, are retrieved. Thus, although the set-based precision of the F100 Article runs is lower, they do give direct access to many more relevant documents.
– The focused runs Base Sec and Base SecLen have a higher set-based character precision than the article level Base run. The length prior on the section index has a positive impact on the precision of the first 1000 characters. The Base Sec and Base SecLen runs have the same number of retrieved articles and retrieved relevant articles, but the Base SecLen run has more relevant text in the first 1000 characters. The query-independent length prior helps locate elements with a larger proportion of relevant text.

2.6 Relevant in Context Task

For the Relevant in Context Task, we group results per article. Table 3 shows the results for the Relevant in Context Task. We make the following observations:

Table 3: Results for the Ad Hoc Track Relevant in Context Task (runs in italics are official submissions, runs with an asteriks are the corrected versions)

Run id	**MAgP**	**gP[5]**	**gP[10]**	**gP[25]**	**gP[50]**
Base Sec Len	0.0597	0.1492	0.1330	0.1080	0.1031
Fusion Sec	0.0563	0.1207	0.1068	0.1008	0.0963
Base	0.1613	0.2900	0.2619	0.2123	0.1766
Base Sec	0.1615	0.3026	0.2657	0.2112	0.1763
*Base Sec Len	**0.1646**	**0.3149**	**0.2790**	**0.2213**	**0.1817**
Fusion	0.1344	0.2849	0.2399	0.1945	0.1547
*Fusion Sec	0.1294	0.2840	0.2427	0.1917	0.1548

Table 4: Results for the Ad Hoc Track Restricted Relevant in Context Task (runs in italics are official submissions, runs with an asteriks are the corrected versions)

Run id	**MAgP**	**gP[5]**	**gP[10]**	**gP[25]**	**gP[50]**
Base F500 Article	0.0576	0.1439	0.1191	0.1053	0.0980
Base Sec F500 Article	0.0566	0.1375	0.1199	0.1040	0.0952
*Base F500 Article	0.1358	0.2516	0.2186	0.1696	0.1473
*Base Sec F500 Article	0.1503	0.2592	0.2288	0.1887	0.1624
Base SecLen F500 Article	0.1545	0.2666	0.2368	0.1868	0.1570

– The difference between the Base and Fusion runs is small.
– The length prior on the section index results in higher early and average precision.

2.7 Restricted Relevant in Context Task

The aim of the Restricted Relevant in Context task is to return relevant results grouped per article, with a restriction to return no more than 500 characters per article. Table 4 shows the results for the Best in Context Task. We make the following observations:

– Similar to the normal Relevant in Context task, the focused run Base Sec F500 Article has somewhat better precision than the run based on the full articles.
– A length prior over the element lengths (Base SecLen F500 Article) leads to a further improvement in precision. Thus, longer elements give higher precision in the first 500 characters.

In summary, with the restrictions on the amount of text per article and per topic that can be retrieved, focused retrieval techniques clearly outperform standard document retrieval. What is somewhat surprising is that a length prior on the section index is effective even when we use the article ranking of an article level run. The query-independent length prior helps locate elements with a large fraction and amount of relevant text.

3 Book Track

In the INEX 2010 Book Track we participated in the Best Book and Prove It tasks. Continuing our efforts of last year, we aim to find the appropriate level of granularity for focused book search. The BookML markup has XML elements on the page level. In the assessments of last year, relevant passages often cover multiple pages [7]. With larger relevant passages, query terms might be spread over multiple pages, making it hard for a page level retrieval model to assess the relevance of individual pages.

Can we better locate relevant passages by considering larger book parts as retrievable units? One simple option is to divide the whole book in sequences of n pages. Another approach would be to use the logical structure of a book to determine the retrievable units. The INEX Book corpus has no explicit XML elements for the various logical units of the books, so as a first approach we divide each book in sequences of pages.

Book index : each whole book is indexed as a retrievable unit.
Page index : each individual page is indexed as a retrievable unit.
5-Page index : each sequence of 5 pages is indexed as a retrievable unit. That is, pages 1-5, 6-10, etc., are treated as text units.

This year's topics are factual statements. For the Best Book Task the aim is to retrieve the most relevant books for the topic of the statement. For the Prove It Task the aim is to return pages that either confirm or refute the factual statement. We submitted six runs in total: two for the Best Book (BB) task and four for the Prove It (PI) task.

Book : a standard Book index run. Up to 100 results are returned per topic.
Book RF : a Book index run with Relevance Feedback (RF). The initial queries are expanded with 50 terms from the top 10 results.
Page : a standard Page index run.
Page RF : a Page index run with Relevance Feedback (RF). The initial queries are expanded with 50 terms from the top 10 results.
5-page : a standard 5-Page index run.
5-Page RF : a 5-Page index run with Relevance Feedback (RF). The initial queries are expanded with 50 terms from the top 10 results.

At the time of writing, no relevance assessments have been made. Therefore we cannot yet provide any evaluation results.

4 Conclusion

In this paper we discussed our participation in the INEX 2010 Ad Hoc and Book Tracks.

For the Ad Hoc Track we found that, with the restrictions on the amount of text per article and per topic that can be retrieved, focused retrieval techniques clearly outperform standard document retrieval. What is somewhat surprising is that a length prior on the section index is effective even when we use the article ranking of an article level run. The query-independent length prior helps locate elements with a large fraction and amount of relevant text.

For the Book Track, no evaluation results have been released. Hopefully, we can report results for the final proceedings.

Acknowledgments Jaap Kamps was supported by the Netherlands Organization for Scientific Research (NWO, grants # 612.066.513, 639.072.601, and 640.001.-501). Marijn Koolen was supported by NWO under grants # 639.072.601 and 640.001.501.

Bibliography

[1] K. N. Fachry, J. Kamps, M. Koolen, and J. Zhang. Using and detecting links in Wikipedia. In *Focused access to XML documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2007)*, volume 4862 of *LNCS*, pages 388–403. Springer Verlag, Heidelberg, 2008.

[2] J. Kamps and M. Koolen. The impact of document level ranking on focused retrieval. In *Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2008)*, volume 5631 of *LNCS*. Springer Verlag, Berlin, Heidelberg, 2009.

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

[4] J. Kamps, M. Koolen, and M. Lalmas. Locating relevant text within XML documents. In *Proceedings of the 31th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 847–849. ACM Press, New York NY, USA, 2008.

[5] J. Kamps, S. Geva, A. Trotman, A. Woodley, and M. Koolen. Overview of the INEX 2008 ad hoc track. In S. Geva, J. Kamps, and A. Trotman, editors, *Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2008)*, volume 5631 of *LNCS*, pages 1–28. Springer Verlag, Berlin, Heidelberg, 2009.

[6] R. Kaptein, M. Koolen, and J. Kamps. Using Wikipedia categories for ad hoc search. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM Press, New York NY, USA, 2009.

[7] G. Kazai, N. Milic-Frayling, and J. Costello. Towards methods for the collective gathering and quality control of relevance assessments. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 452–459,

New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: http://doi.acm.org/10.1145/1571941.1572019.

[8] M. Koolen and J. Kamps. The importance of anchor-text for ad hoc search revisited. In H.-H. Chen, E. N. Efthimiadis, J. Savoy, F. Crestani, and S. Marchand-Maillet, editors, *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 122–129. ACM Press, New York NY, USA, 2010.

[9] M. Koolen, R. Kaptein, and J. Kamps. Focused search in books and Wikipedia: Categories, links and relevance feedback. In S. Geva, J. Kamps, and A. Trotman, editors, *Focused Retrieval and Evaluation: 8th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2009)*, volume 6203 of *LNCS*, pages 273–291, 2010.

[10] R. Schenkel, F. Suchanek, and G. Kasneci. Yawn: A semantically annotated wikipedia xml corpus, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.140.5501.

[11] B. Sigurbjörnsson and J. Kamps. The effect of structured queries and selective indexing on XML retrieval. In *Advances in XML Information Retrieval and Evaluation: INEX 2005*, volume 3977 of *LNCS*, pages 104–118, 2006.

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

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

[14] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: a language-model based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, 2005.

Combining Page Scores for XML Book Retrieval

Ray R. Larson

School of Information
University of California, Berkeley
Berkeley, California, USA, 94720-4600
ray@ischool.berkeley.edu

Abstract. In the 2010 INEX Evaluation UC Berkeley participated only in the Book track, and specifically the "Best Books to Reference" task that seeks to produce a list of the "best books" for a topic. Last year and in 2008 we tried a variety of different approaches for our Book Track runs, including the TREC2 logistic regression probabilistic model as well as various fusion approaches including combining the Okapi BM-25 algorithm with other approaches. But in most cases our previous approaches used only full-book level indexes. This year we wanted to compare our best performing method from last year with approaches that combine the scores obtained by ranking at the page level to arrive at the ranking for a book. No results are yet available for the track, so this short paper concentrates on the approaches used in our submitted runs.

1 Introduction

In our 2009 results from the INEX Book Track, we observed that most of the fusion approaches that we had tried were not as effective as the TREC2 Logistic Regression with Blind Feedback, so we took that as our baseline for this year. After testing a number of approaches using book-level indexes and different fusion weights, we found that this was still the case, and that these attempts often led to poor matches being ranked highly. We decided instead to try some radical simplification of the ranking process for books. This was driven by observations from earlier INEX evaluations and from some of our digital library work that often the books with highly ranked *pages* turned out to be more better choices for the user than books with high overall ranking scores. Since we had generated page-level indexes for all of the books (see below), we decided to try two simple approaches. A probabilistic approach based on our logistic regression algorithm (but without blind feedback), and a simple coordination-level match for pages.

In this paper we will first discuss the algorithms and operators used in our official INEX 2010 Book Track runs. Then we will look at how these algorithms and operators were used in combination with page-level indexes for our submissions, and finally we will discuss possible directions for future research.

2 The Retrieval Algorithms and Fusion Operators

This section largely duplicates parts of earlier INEX papers in describing the probabilistic retrieval algorithms used for the Book track in INEX this year. Although the algorithms are the same as those used in previous years for INEX and in other evaluations (such as CLEF and NTCIR), 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 discuss the simple methods used to combine the results of searches of book page elements in the collections. All runs used our Cheshire II XML/SGML search engine [9, 8, 7] which also supports a number of other algorithms for distributed search and operators for merging result lists from ranked or Boolean sub-queries.

2.1 TREC2 Logistic Regression Algorithm

Once again the primary algorithm used for our INEX baseline 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 has provided good retrieval performance earlier evaluations[9, 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)[11] and TREC-8 (Voorhees and Harman 1999)[12].)

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. We used the same default as last year, i.e., 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 Coordination Level Matching

Coordination level matching is the first simple step towards ranking results beyond simple Boolean matches. Basic coordination level matching (CML) is simply the number of terms in common between the query and the document component or $|Q_c|$ as defined above. In the implementation that we use in the Cheshire II system, the coordination level match (CLM) also takes into account term frequency, thus it is simply:

$$CLM_c = \sum_{i=1}^{|Q_c|} tf_i \tag{2}$$

Where the variables are the defined the same as defined above. Obviously, with this simple form, it is possible for terms that have very high frequency to dominate. To combat this an additional filter removes all results that match on fewer than 1/4 of the search terms.

2.4 Result Combination Operators

As we have also reported previously, the Cheshire II system used in this evaluation provides a number of operators to combine the intermediate results of a search from different components or indexes. With these operators we have available an entire spectrum of combination methods ranging from strict Boolean operations to fuzzy Boolean and normalized score combinations for probabilistic and Boolean results. These operators are the means available for performing fusion operations between the results for different retrieval algorithms and the search results from different components of a document.

However, our approach for the page-level searches done for this evaluation was to simply sum the page-level results for each book. Thus, for the CLM runs, if a particular query retrieved 10 pages from a given book, the final ranking score would be the sum of the CLM values for each page. Although the runs using the TREC2 logistic regression algorithm return estimates of the probability of relevance for each page, we decided to treat these also as simple scores and sum each matching page estimate for each book.

3 Database and Indexing Issues

The Book Track data used this year was the same as last year. In indexing we attempted to use multiple elements or components that were identified in the Books markup including the Tables of Contents and Indexes as well as the full text of the book, since the goal of the "Best Books" task was to retrieve entire books and not elements, the entire book was retrieved regardless of the matching elements.

Table 2 lists the Book-level (/article) indexes created for the INEX Books database and the document elements from which the contents of those indexes were extracted.

Table 2. Book-Level Indexes for the INEX Book Track 2009-10

Name	Description	Contents	Vector?
topic	Full content	//document	Yes
toc	Tables of Contents	//section@label="SEC_TOC"	No
index	Back of Book Indexes	//section@label="SEC_INDEX"	No

Table 3. Components for INEX Book Track 2009-10

Name	Description	Contents
COMPONENT_PAGE	Pages	//page
COMPONENT_SECTION	Sections	//section

Cheshire system permits parts of the document subtree to be treated as separate documents with their own separate indexes. Tables 3 & 4 describe the XML components created for the INEX Book track and the component-level indexes that were created for them.

Table 3 shows the components and the paths used to define them. The first, refered to as COMPONENT_PAGE, is a component that consists of each identified page of the book, while COMPONENT_SECTION identifies each section of the books, permitting each individual section or page of a book to be retrieved separately. Because most of the areas defined in the markup as "section"s are actually paragraphs, we treat these as if they were paragraphs for the most part.

Table 4. Component Indexes for INEX Book Track 2009-10

Component or Index Name	Description	Contents	Vector?
COMPONENT_SECTION			
para_words	Section Words	* (all)	Yes
COMPONENT_PAGES			
page_words	Page Words	* (all)	Yes

Table 4 describes the XML component indexes created for the components described in Table 3. These indexes make the individual sections (such as COMPONENT_SECTION) of the INEX documents retrievable by their titles, or by any terms occurring in the section. These are also proximity indexes, so phrase searching is supported within the indexes.

We also have indexes created using the MARC data (book-level metadata) made available, but these were not used this year.

3.1 Indexing the Books XML Database

Because the structure of the Books database was derived from the OCR of the original paper books, it is primarily focused on the page organization and layout and not on the more common structuring elements such as “chapters” or “sections”. Because this emphasis on page layout goes all the way down to the individual word and its position on the page, there is a very large amount of markup for page with content. For this year’s original version of the Books database, there are actually NO text nodes in the entire XML tree, the words actually present on a page are represented as attributes of an empty word tag in the XML. The entire document in XML form is typically multiple megabytes in size. A separate version of the Books database was made available that converted these empty tags back into text nodes for each line in the scanned text. This provided a significant reduction in the size of database, and made indexing much simpler. The primary index created for the full books was the “topic” index containing the entire book content.

We also created page-level “documents” as we did last year. As noted above the Cheshire system permits parts of the document subtree to be treated as separate documents with their own separate indexes. Thus, paragraph-level components were extracted from the page-sized documents. Because unique object (page) level 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.

The system problems encountered last year have been (temporarily) corrected for this years submissions. Those problems were caused by the numbers of unique terms exceeding the capacity of the integers used to store them in the indexes. For this year, at least, moving to unsigned integers has provided a temporary fix for the problem but we will need to rethink how statistical summary information is handled in the future – perhaps moving to long integers, or even floating point numbers and evaluating the tradeoffs between precision in the statistics and index size (since moving to Longs could double index size).

4 INEX 2010 Book Track Runs

We submitted nine runs for the Book Search task of the Books track,

As Table 5 shows, a small number of variations of algorithms and search elements were tried this year. The small number was largely due to some issues in indexing (due to a bug in page indexes that took a lot of time to locate and fix). With more that 16 million pages, response time was very good for the basic search operations, but slowed dramatically whenever data from records was needed.

Table 5. Berkeley Submissions for the INEX Book Track 2009

Name	Description	Algorithm	Combined?
T2FB_BASE_BST	Uses book-level topic index and blind feedback	TREC2 +BF	NA
CLM_PAGE_SUM	Uses page components and page_words index	CLM	Sum
CLM_PAGE_SUM_300	Uses page components and page_words index	CLM	Sum
T2_PAGE_SUM_300	Uses page components and page_words index	TREC2	Sum

In Table 5 the first column is the run name (all of our official submissions had names beginning with "BOOKS10" which has been removed from the name), the second column is a short description of the run. We used only the main "fact" element of the topics in all of our runs. The third column shows which algorithms where used for the run, TREC2 is the TREC2 Logistic regression algorithm described above, "BF" means that blind relevance feedback was used in the run, and CLM means that the CLM algorithm described above (2) was used.

5 Conclusions and Future Directions

The results of the Books track are not yet available, but a few observations can be made about the runs. We suspect that the simple CLM on pages may outperform the TREC2 approach on pages. This is partially because of the tendency for the simple matching algorithm to give the strongest values to "best matches", from some eyeballing of the data. However, the real results often come as a surprise.

We also hope to find time to run some additional analyses of the data and try some alternative approaches before the final version of proceedings are published.

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, component fusion and blind feedback for XML retrieval. In *INEX 2005*, pages 225–239. Springer (Lecture Notes in Computer Science, LNCS 3977), 2006.
10. R. R. Larson. Ranking and fusion approaches for XML book retrieval. In *Focused Retrieval and Evaluation (INEX 2009)*, pages 179–189. Springer (Lecture Notes in Computer Science, LNCS 6203), 2010.
11. E. Voorhees and D. Harman, editors. *The Seventh Text Retrieval Conference (TREC-7)*. NIST, 1998.
12. E. Voorhees and D. Harman, editors. *The Eighth Text Retrieval Conference (TREC-8)*. NIST, 1999.

OUC's participation in the 2010 INEX Book Track

Michael Preminger[1] and Ragnar Nordlie[1]

Oslo University College

Abstract. In this article we describe the Oslo University College's participation in the INEX 2010 Book track. The 2010 tasks have been featuring a relatively large number of topics (82). This, combined with the character of the main task of finding book pages that prove or refute a given factual utterance, allow for a better calibration of retrieval methods.
The OUC has submitted retrieval results for the "prove it" task with traditional relevance detection combined with some rudimental detection of confirmation. We call for a broader discussion of a more meaning-oriented (semantics-aware) approach to retrieval in digitized books, with the "prove it" task (classifiable as a simple semantics- aware retrieval activity) providing the INEX milieu with a suitable context to start this discussion.

1 Introduction

In recent years large organizations like national libraries, as well as multinational organizations like Microsoft and Google have been investing labor, time and money in digitizing books. Beyond the preservation aspects of such digitization endeavors, they call on finding ways to exploit the newly available materials, and an important aspect of exploitation is book and passage retrieval.

The INEX Book Track[1], which has been running since 2007, is an effort aiming to develop methods for retrieval in digitized books. One important aspect here is to test the limits of traditional methods of retrieval, designed for retrieval within "documents" (such as news-wire), when applied to digitized books. One wishes to compare these methods to book-specific retrieval methods.

One important mission of such retrieval is supporting the generation of new knowledge based on existing knowledge. The generation of new knowledge is closely related to access to – as well as faith in – existing knowledge. One important component of the latter is claims about facts. This year's "prove it" task, may be seen as challenging the most fundamental aspect of generating knew knowledge, namely the establishment (or refutal) of factual claims encountered during research.

On the surface, this may be seen as simple retrieval, but proving a fact is more than finding relevant documents. This type of retrieval requires from a passage to "make a statement about" rather than "be relevant to" a claim, which traditional retrieval is about. The questions we pose here are:

- *what is the difference between simply being relevant to a claim and expressing support for a claim*
- how do we modify traditional retrieval to reveal support or refutal of a claim?

We see proving and denial of a statement as different tasks, both classifiable as semantics-aware retrieval, suspecting that the latter is a more complicated task. This paper attempts at applying some rudimentary techniques of detecting the confirmation (proving) of a statement. The rest of the paper discusses these tasks in the context of meaning-oriented retrieval in books.

2 Indexing and retrieval strategies

The point of departure of the strategies discussed here is that confirming or refuting a statement is a simple action of speech that does not require from the book (the context of the retrieved page) to be ABOUT the topic covering the fact. This means that we do not need the index to be context-faithful (pages need not be indexed in a relevant book context). It is more the formulation of the statement in the book or page that matters. This is why we need to look for words (or sequences of words) or sentences that indicate the stating of a fact. A simple strategy is looking for the occurrence of words like ”is”, ”are”, ”have”, ”has” a.s.o, that, in combination with nouns from the query (or fact formulation), indicate a possible act of confirming the fact in question.

Further focus may be achieved by detecting sentences that include (WH-) question indicators or a question-mark and pruning these from the index, so that pages that only match the query through such sentences are omitted or weighed down during retrieval.

Against this background we were trying to construct runs that emphasized pages that are confirmative in style. The pages in the collection where attempted divided into categories of how confirmative they are. Occurrences of the words *is, are, was, were, have, has* were counted in each page, and a ratio between this sum and the total number of words in the page was calculated. Based on a sample of the pages, three levels were defined, so that pages belonging to each of the levels were assigned a tag, accordingly.

These tags then facilitated weighing different pages differently when retrieving candidates of confirming pages.

3 Runs and Results

By the submission deadline no results were available for evaluation.

4 Discussion

Utilizing digital books poses new challenges on information retrieval. The mere size of the book text poses both storage, performance and content related challenges as compared to texts of more moderate size. But the challenges are even greater if books are to be exploited not only for finding factual facts, but also to support exploitation of knowledge, identifying and analyzing ideas, a.s.o

For example, we suspect that confirming and refuting a factual statement, the Book Track 2010 ”prove it” task, both belong to a class of activities that extend the current scope of information retrieval. Confirming a fact may have many facets, based on how complicated the fact is. A fact like: *The ten tribes forming the northern kingdom of Israel (aka the ten lost tribes) disappeared after being driven to exile by the Assyrians, several hundreds years before Christ* may be confirmed on several levels. Should all minor details be in place for the fact to be confirmed? What if the book states that it was the Babylonians, rather than the Assyrians who sent the tribes into exile, the rest of the details being in agreement with the statement: is the fact then confirmed? Moreover, detecting the refutal of a statement is arguably a totally different activity than detecting its confirmation.

Even though such activities may be developed and refined using techniques from e.g. Question Answering[2], we suspect that employing semantics-aware retrieval [3,**?**], which is closely connected to the development of the Semantic Web [4] would be a more viable (and powerful) path to follow.

Within the INEX Book track, the ”prove it” task can thus serve as a splendid start of a broader discussion around detecting meaning rather than only matching strings. Many projects under way are already using ontologies to aid in tagging texts of certain kinds (e.g. philosophical essays)[5] to indicate certain meaning, with the aim of supporting the analysis of these texts. Is this a viable task for the INEX Book track? Is it a viable path for information retrieval?

5 Conclusion

This article is an attempt to start a discussion about semantics-aware retrieval in the context of the INEX book track. Proving of factual statements is discussed in light of some rudimental retrieval experiments incorporating the detection of confirmation (proving) of statement. We also discuss the task of proving statement, raising the question whether it is classifiable as a semantics-aware retrieval task.

References

1. Kazai, G., Koolen, M., Landoni, M.: Summary of the book track. In: INEX 2009. (in press) 0–0
2. VOORHEES, E.M.: The trec question answering track. Natural Language Engineering **7** (2001) 361–378
3. Tim Finin, James Mayfield, A.J.R.S.C., Fink, C.: Information retrieval and the semantic web. In: Proc. 38th Int. Conf. on System Sciences, Digital Documents Track (The Semantic Web: The Goal of Web Intelligence). (2005) 0–0
4. Berners-Lee, T., H.J., Lassila, O.: The semantic web. Scientific American (2001)
5. Zllner-Weber, A.: Ontologies and logic reasoning as tools in humanities? DHQ: Digital Humanities Quarterly **3**(4) (2009)

Overview of the INEX 2010 Data Centric Track

Andrew Trotman

Department of Computer Science
University of Otago
Dunedin
New Zealand

Qiuyue Wang

School of Information
Renmin University of China
Beijing
China

Abstract. The INEX 2010 Data Centric Track is discussed. A dump of IMDb was used as the document collection, 28 topics were submitted, 36 runs were submitted by 8 institutes, and 26 topics were assessed. Most runs (all except 2) did not use the structure present in the topics; and consequently no improvement is yet seen by search engines that do so.

1 Introduction

2010 sees the introduction of the Data Centric Track at INEX. The results of INEX up-to and including 2009 showed that whole document retrieval was effective. This result was, perhaps, a consequence of the IEEE and Wikipedia collections used in the past. It is reasonable to assume the Wikipedia will include a whole document result to almost any ad hoc query.

In the Data Centric Track we ask: Is the same result seen when a highly structured document collection is used?

To answer this question a new highly structured collection was developed and made available for download from the INEX website. That collection was a snapshot of the IMDB taken early in 2010. Highly structured queries were solicited from participants. Together with the assessments these form the new INEX Data Centric Collection.

Most of the runs submitted to the track did not use the structure present in the topics. This is not surprising in a new track because participants are inclined to use their existing systems on a new collection before making modifications to it. Consequently the track has not yet seen improvements in precision from structure. It is hoped that in future years participating groups will prefer to conduct experiments using the structure present in the topics. The track has generated a topic set that can be used for training.

2 The Task

In its first year, the track focused on ad hoc retrieval from XML data. An XML document is typically modeled as a rooted, node-labeled tree. An answer to a keyword query was defined as a set of *closely related* nodes that are *collectively relevant* to the query. So each result could be specified as a collection of nodes from one or more XML documents that are related and collectively cover the relevant information. The task was to return a ranked list of results estimated relevant to the user's information need. The content of the collections of nodes was not permitted to overlap. This is similar to the focused task in the ad hoc track, but using a data-centric XML collection and allowing the construction of a result (i.e. a collection of nodes) from different parts of a single document or even multiple documents.

3 INEX Data Centric Track Collection

3.1 Document Collection

The track used the IMDb data collection newly built from www.imdb.com. It was converted from the plain text files (April 10, 2010) published on the IMDb web site. The plain text files were first loaded into a relational database by the Java Movie Database system[1]. Then the relational data are published as XML documents according to the DTDs. There are two kinds of objects in the IMDb data collection, movies and persons involved in movies, e.g. actors/actresses, directors, producers and so on. Each object is richly structured. For example, each movie has title, rating, directors, actors, plot, keywords, genres, release dates, trivia, etc.; and each person has name, birth date, biography, filmography, etc. Please refer to the Appendix A and B for the movie DTD and person DTD respectively.

Information about one movie or person is published in one XML file, thus each generated XML file represents a single object, i.e. a movie or person. In total, 4,418,102 XML files were generated, including 1,594,513 movies, 1,872,492 actors[2], 129,137 directors who did not act in any movies, 178,117 producers who did not direct or act in any movies, and 643,843 other people involved in movies who did not produce or direct nor act in any movies.

3.2 Topics

Each participating group was asked to create a set of candidate topics, representative of a range of real user needs. Both Content Only (CO) and Content And Structure (CAS) variants of the information need were requested. In total 30 topics were submitted by 4 institutes (IRIT / SIG, Renmin University of China, Universidade

[1] http://www.jmdb.de/

[2] 21 of the actor files were empty and removed in the new IMDB data collection.

Federal do Amazonas, and Universitat Pompeu Fabra). From these a total of 28 topics were selected. An example topic (2010001) is given in Fig**.** 1:

```
<topic id="2010001" ct_no="3">
<title>Yimou Zhang 2010 2009</title>
<castitle>//movie[about(.//director, "Yimou Zhang")
            and (about(.//releasedate, 2010) or about(.//releasedate, 2009))]</castitle>
<description>I want to know the latest movies directed by Yimou Zhang.</description>
<narrative>
            I am interested in all movies directed by Yimou Zhang,
           and I want to learn the latest movies he directed.
</narrative>
</topic>
```

Fig. 1. INEX 2010 Data Centric Track Topic 2010001

4 Submission Format

The required submission format was a variant of the familiar TREC format used by INEX, the so called TREC++ format. The following information was collected about each run:

- The participant ID of the submitting institute,
- Whether the query was constructed automatically or manually from the topic,
- Topic fields used (from: Title, CASTitle, Description, and Narrative),

A run was permitted to contain a maximum of 1000 results for each topic. A result consisted of one or more nodes from a single or multiple XML documents. A node is uniquely identified by its element path in the XML document tree. The standard TREC format is extended with one additional field for specifying each result node: <qid> Q0 <file> <rank> <rsv> <run_id> <column_7>

Here:

- the first column is the topic number.
- the second column is the query number within that topic (unused and should always be Q0).
- the third column is the file name (without .xml) from which a result node is retrieved.
- the fourth column is the rank of the result. Note that a result may consist of one or more related nodes, so there can be multiple rows with the same rank if these nodes belong to the same result.
- the fifth column shows the score that generated the ranking. This score must be in descending (non-increasing) order and is important to include so that assessment tools can handle tied scores (for a given run) in a uniform fashion (the evaluation routines rank documents from these scores, not from ranks). If you want the precise ranking that you submit to be evaluated, the scores should reflect that ranking.

- the sixth column is called the "run tag" and should be a unique identifier from within a participating group. It should also include a brief detail of the method used. The run tags contained 12 or fewer letters and numbers, with no punctuation.
- the seventh column gives the element path of the result node. Element paths are given in XPath syntax. To be more precise, only fully specified paths are allowed, as described by the following grammar:

```
Path          ::=   '/' ElementNode Path | '/' ElementNode | '/' AttributeNode
ElementNode   ::=   ElementName Index
AttributeNode ::=   '@' AttributeName
Index         ::=   '[' integer ']'
```

For Example the path */article[1]/bdy[1]/sec[1]/p[1]* identifies the element which can be found if we start at the document root, select the first *article* element, then within that, select the first *body* element, within which we select the first *section* element, and finally within that element we select the first *p* element.

An example submission is:

```
1 Q0 9996 1 0.9999 I09UniXRun1 /article[1]/bdy[1]/sec[1]
1 Q0 9996 1 0.9999 I09UniXRun1 /article[1]/bdy[1]/sec[2]/p[1]
1 Q0 9888 1 0.9999 I09UniXRun1 /article[1]/bdy[1]/sec[3]
1 Q0 9997 2 0.9998 I09UniXRun1 /article[1]/bdy[1]/sec[2]
1 Q0 9989 3 0.9997 I09UniXRun1 /article[1]/bdy[1]/sec[3]/p[1]
```

Here there are three results. The first result contains the first section and first paragraph of the second section from 9996.xml, and the third section from 9888.xml. The second result only consists of the second section in 9997.xml, and the third result consists of the first paragraph of the third section from 9989.xml.

5 Submitted Runs

Participants were permitted to submit up to 10 runs. Each run was permitted to contain a maximum of 1000 results per topic, ordered by decreasing value of relevance. Runs were permitted to use any fields of the topics, but only runs using either the <title>, or <castitle>, or a combination of them were regarded as truly automatic. The results of one run was contained in one submission file and so up to 10 files per group could be submitted.

In total 36 runs were submitted by 8 institutes. Those institutes were: Benemérita Universidad Autónoma de Puebla, Indian Statistical Institute, Kasetsart University, Peking University, Renmin University of China, Universidade Federal do Amazonas, Universitat Pompeu Fabra, and the University of Otago. Only 29 runs were assessed since other runs were submitted after the deadline. Of note is that more runs were submitted than topics, and more institutes submitted runs than that submitted topics. This suggests an increase in interest in the track throughout the year.

6 Assessment and Evaluation

Shlomo Geva ported the tool to work with the IMDb collection and in doing so identified some problems with the document collection. The collection was, consequently, cleaned for use with the assessment tool. The new collection will most likely be used in 2011, if the track continues.

Assessment was done by those groups that submitted runs. In total 26 of the 28 topics were assessed. Topics 2010003 and 20100013 were not assessed, all others were. The evaluation results presented herein were computed using just the 26 assessed topics with the other 2 topics dropped from the runs.

Jaap Kamps used the (unmodified) INEX and TREC evaluation tools on the runs. The TREC MAP metric was used to measure the performance of the runs at whole document retrieval. The INEX thorough retrieval MAiP metric and the INEX Relevant-in-Context MAgP T2I(300) metrics were used to measure Focused Retrieval. Although the run submission permitted the use of aggregated retrieval, it has not yet become clear how to measure aggregation and so the track organisers chose to fall-back to more traditional measures for 2010. Descriptions of the INEX and TREC measures are not given herein as they are well known and described elsewhere (see the ad hoc track overview paper pre-proceedings).

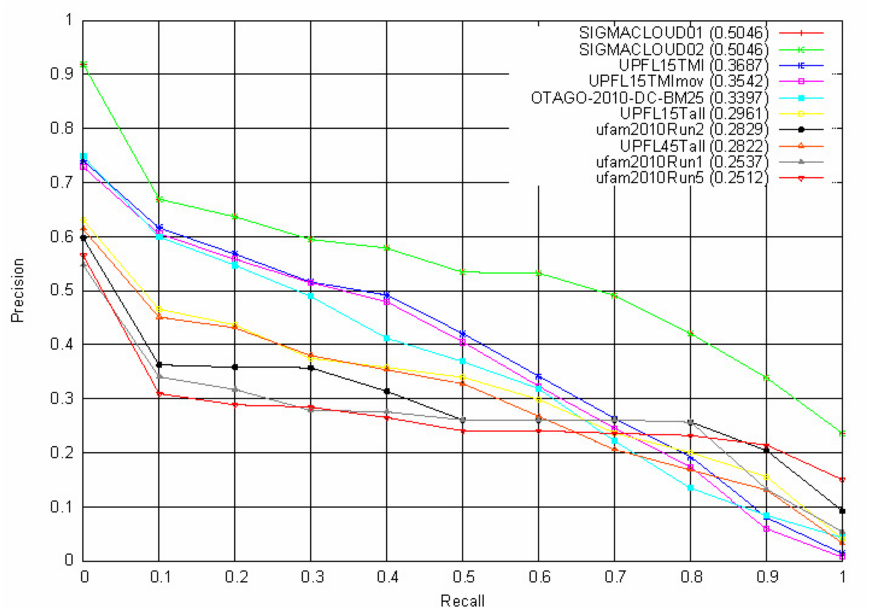


Fig. 2. Best runs measured with MAP

7 Results

The performance of the runs using the whole document based MAP metric are presented in Fig. 2. The best run, SIGMACLOUD01 was submitted by Peking University and performed substantially better than the next best run at all recall points. We note that this run used the description and narrative of the topic whereas the other runs did not (formally it is not an INEX automatic run and must be considered a manual run). The runs from Benemérita Universidad Autónoma de Puebla used the castitle and all other runs used the title. That is, despite being the data centric, most runs did not use structure in ranking.

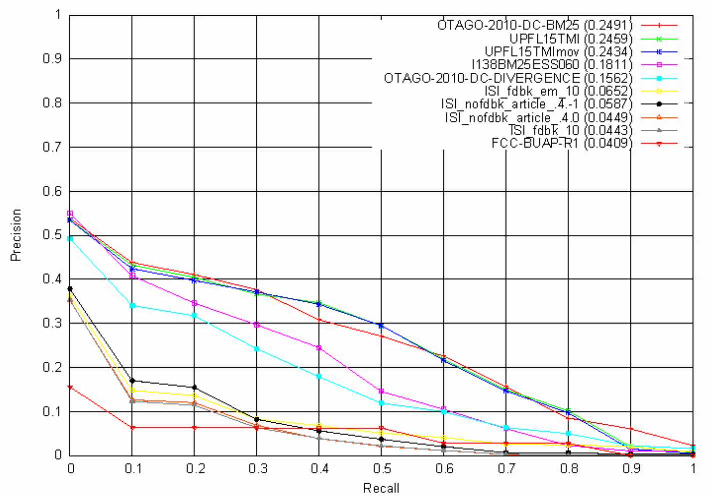


Fig. 3. Best runs measured with MAgP T2I(300)

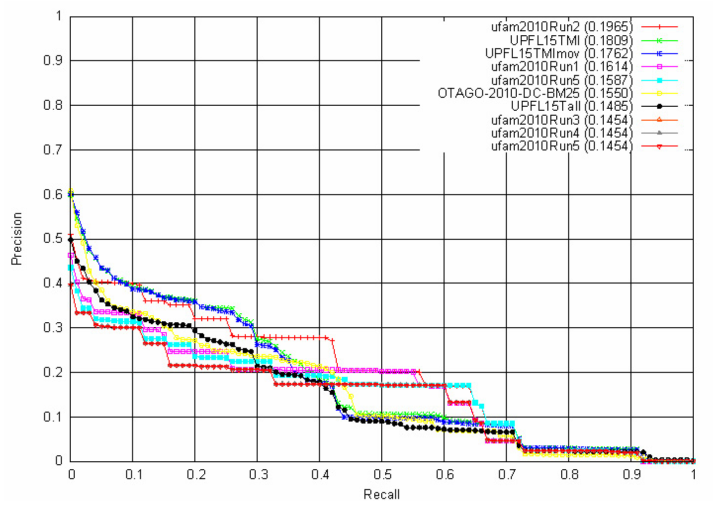


Fig. 4. Best runs measured with MAiP

From visual inspection, there is little difference between the next three runs. The Otago run (that placed 3rd amongst the automatic runs) is a whole document run generated from the title of the topic by using the BM25 ranking function trained on the INEX 2009 document collection – it is equivalent to the ad hoc reference run. It can be considered a baseline for performance.

When measured using the MAgP T2I(300) metric (see Fig. 3 the Otago reference-like run performs best, however there is a cluster of 3 runs performing at about (from visual inspection) the same level. When measured using MAiP (Fig. 4 the reference-like run shows high early precision but quickly decreases. Of course, whole document retrieval is not a good strategy for thorough retrieval because precisely 1 element is returned per document. Those runs that exhibited overlap were not evaluated using the MAgP metric.

8 Conclusions

The track has successfully produced a highly structured document collection including structured documents (IMDb), structured queries, and assessments. The participants of the track submitted runs and those runs were evaluated. Because most runs did not use structured queries no claim can be made about the advantage of doing so. This is expected to change in future years. The track was overly ambitions in allowing result aggregation. No method of measuring the performance of aggregated retrieval was developed for the track in 2010 and is left for future years.

9 Acknowledgements

Thanks are given to the participants who submitted the topics, the run, and performed the assessment process. Special thanks go to Shlomo Geva for porting the assessment tools, and to Jaap Kamps for performing the evaluation. Finally, some of the contents of this paper was taken from the INEX web site which was authored by many people – we thank each of those for their contribution to the text in this paper.

Appendix A: Movie DTD

```
<!ELEMENT movie (title, url, overview?, cast?, additional_details?, fun_stuff?)>
<!ATTLIST movie xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink">
<!ELEMENT title (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

```
<!ELEMENT overview (rating?, directors?, writers?, releasedates?, genres?, tagline?,
plot?, keywords?) >
<!ELEMENT rating (#PCDATA)>
<!ELEMENT directors (director+)>
<!ELEMENT director (#PCDATA)>
<!ELEMENT writers (writer+)>
<!ELEMENT writer (#PCDATA)>
<!ELEMENT releasedates (releasedate+)>
<!ELEMENT releasedate (#PCDATA)>
<!ELEMENT genres (genre+)>
<!ELEMENT tagline (#PCDATA)>
<!ELEMENT plot (#PCDATA)>
<!ELEMENT keywords (keyword+)>
<!ELEMENT keyword (#PCDATA)>

<!ELEMENT cast (actors?, composers?, cinematographers?, producers?,
production_designers?, costume_designers?, miscellaneous?)>

<!ELEMENT actors (actor+)>
<!ELEMENT actor (name, character?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT character (#PCDATA)>
<!ELEMENT composers (composer+)>
<!ELEMENT composer (#PCDATA)>
<!ELEMENT cinematographers (cinematographer+)>
<!ELEMENT cinematographer (#PCDATA)>
<!ELEMENT producers (producer+)>
<!ELEMENT producer (#PCDATA)>
<!ELEMENT production_designers (production_designer+)>
<!ELEMENT production_designer (#PCDATA)>
<!ELEMENT miscellaneous (person+)>
<!ELEMENT person (#PCDATA)>

<!ELEMENT additional_details
(aliases?,mpaa?,runtime?,countries?,languages?,colors?,certifications?,locations?,com
panies?,distributors?)>
<!ELEMENT aliases (alias+)>
<!ELEMENT alias (#PCDATA)>
<!ELEMENT mpaa (#PCDATA)>
<!ELEMENT runtime (#PCDATA)>
<!ELEMENT countries (country+)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT languages (language+)>
<!ELEMENT language (#PCDATA)>
<!ELEMENT colors (color+)>
```

```
<!ELEMENT color (#PCDATA)>
<!ELEMENT certifications (certification+)>
<!ELEMENT certification (#PCDATA)>
<!ELEMENT locations (location+)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT companies (company+)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT distributors (distributor+)>
<!ELEMENT distributor (#PCDATA)>

<!ELEMENT fun_stuff (trivias?,goofs?,quotes?,movielinks?)>
<!ELEMENT trivias (trivia+)>
<!ELEMENT trivia (#PCDATA)>
<!ELEMENT goofs (goof+)>
<!ELEMENT goof (#PCDATA)>
<!ELEMENT quotes (quote+)>
<!ELEMENT quote (#PCDATA)>
<!ELEMENT movielinks (movielink+)>
<!ELEMENT movielink (#PCDATA|link)*>
<!ELEMENT link (#PCDATA)>
<!ATTLIST link xlink:type CDATA #IMPLIED>
<!ATTLIST link xlink:href CDATA #IMPLIED>
```

Appendix B: Person DTD

```
<!ELEMENT person (name, overview?,filmography?, additional_details?)>
<!ELEMENT name (#PCDATA)>

<!ELEMENT overview (birth_name?, birth_date?, death_date?, height?, spouse*,
trademark*, biographies?, nicknames?, trivias?, personal_quotes?,
where_are_they_now?, alternate_names?, salaries?) >
<!ELEMENT birth_date (#PCDATA)>
<!ELEMENT death_date (#PCDATA)>
<!ELEMENT birth_name (#PCDATA)>
<!ELEMENT nicknames (name+)>
<!ELEMENT alternate_names (name+)>
<!ELEMENT height (#PCDATA)>
<!ELEMENT spouse (#PCDATA)>
<!ELEMENT trademark (#PCDATA)>
<!ELEMENT biographies (biography, by)>
<!ELEMENT biography (#PCDATA)>
<!ELEMENT by (#PCDATA)>
<!ELEMENT trivias (trivia+)>
<!ELEMENT trivia (#PCDATA)>
```

```
<!ELEMENT personal_quotes (quote+)>
<!ELEMENT salaries (salary+)>
<!ELEMENT quote (#PCDATA)>
<!ELEMENT salary (#PCDATA)>
<!ELEMENT where_are_they_now (where+)>
<!ELEMENT where (#PCDATA)>

<!ELEMENT filmography (act?, direct?, write?, compose?, edit?, produce?,
production_design?, cinematograph?, costume_design?, miscellaneous?)>
<!ELEMENT act (movie+)>
<!ELEMENT direct (movie+)>
<!ELEMENT write (movie+)>
<!ELEMENT compose (movie+)>
<!ELEMENT edit (movie+)>
<!ELEMENT produce (movie+)>
<!ELEMENT production_design (movie+)>
<!ELEMENT cinematograph (movie+)>
<!ELEMENT costume_design (movie+)>
<!ELEMENT miscellaneous (movie+)>
<!ELEMENT movie (title, year, character?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT character (#PCDATA)>

<!ELEMENT additional_details (otherworks?, public_listings?)>
<!ELEMENT otherworks (otherwork+)>
<!ELEMENT otherwork (#PCDATA)>
<!ELEMENT public_listings (interviews?, articles?, biography_prints?,
biographical_movies?, portrayed_ins?, magazine_cover_photos?, pictorials?)>
<!ELEMENT interviews (interview+)>
<!ELEMENT articles (article+)>
<!ELEMENT biography_prints (print+)>
<!ELEMENT biographical_movies (biographical_movie+)>
<!ELEMENT portrayed_ins (portrayed_in+)>
<!ELEMENT magazine_cover_photos (magazine+)>
<!ELEMENT pictorials (pictorial+)>
<!ELEMENT interview (#PCDATA)>
<!ELEMENT article (#PCDATA)>
<!ELEMENT print (#PCDATA)>
<!ELEMENT biographical_movie (#PCDATA)>
<!ELEMENT portrayed_in (#PCDATA)>
<!ELEMENT magazine (#PCDATA)>
<!ELEMENT pictorial (#PCDATA)>
```

Automatically Generating Structured Queries in XML Keyword Search

Felipe Hummel[1], Altigran S. da Silva[1],
Mirella M. Moro[2], and Alberto H. F. Laender[2]

[1] Departamento de Ciência da Computação
Universidade Federal do Amazonas
`{fch,alti}@dcc.ufam.edu.br`
[2] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
`{mirella,laender}@dcc.ufmg.br`

Abstract. In this paper, we present a novel method for automatically deriving structured XML queries from keyword-based queries and show how it was applied to the experimental tasks proposed for the INEX 2010 data-centric track. In our method, called StruX, users specify a schema-independent unstructured keyword-based query and it automatically generates a top-k ranking of schema-aware queries based on a target XML database. Then, one of the top ranked structured queries can be selected, automatically or by a user, to be executed by an XML DBMS. The generated structured queries are XPath expressions consisting of an entity path (e.g.,`dblp/article`) and predicates (e.g. `/dblp/article[author="john" and title="xml"]`). We use the concept of entity, commonly adopted in the XML keyword search literature, to define suitable root nodes as query results. Also, our method uses IR techniques to determine in which elements a term is more likely to occur.

Keywords: XML, Keyword Search, XPath

1 Introduction

Specifying queries through keywords is currently very common. Specially in the context of search engines in the *World Wide Web*, users with different levels of computer skills are used to keyword-based searching. As a consequence, this approach has been exploited outside the scope of the Web. For example, in relational databases, several methods [1–3, 8, 10, 21] have been proposed with this focus. Considering the vast number of applications that use such databases, it is clear why there is so much interest in adopting such an approach to develop more intuitive interfaces for querying in this environment.

In the last few years, there has been an increasing interest in the field of keyword-based search over XML documents, given the growth and consolidation of such a standard. Many works employ the concept of *Lowest Common Ancestor* (LCA) [7] with variations to specific requirements, including Meaningful

LCA (MLCA) [17], Smallest LCA (SLCA) [24], Valuable LCA (VLCA) [14], XSeek [18] and Multiway-SLCA [23]. Another structure-related concept, *Minimum Connecting Trees* [9], has led to a different approach to the problem. All of these approaches aim at finding, inside the XML document, subtrees where each query term occurs at least one time in one of its leafs, returning the root node of the subtrees as a query result. Specifically, LCA-based methods make restrictions on the choice of the root node. Notice that for one term queries, these methods tend to return a single-element subtree, which is generally not desired as a query result. Furthermore, after an initial indexing phase, those methods disregard the source XML document, as any queries will be answered only considering their own indexes. This behavior may not be suitable in a dynamic environment in which data is frequently updated or when the XML data is stored in a DBMS. Considering such an environment, it becomes interesting to develop XML keyword search approaches that can easily cope with data stored and managed by a DBMS. For instance, as current XML-aware DBMS can perform XQuery and XPath queries in an efficient way, one could use that to abstract the frequent updates and storage of the XML data.

This paper presents a novel method for keyword search over XML data based on a fresh perspective. Specifically, our method, called *StruX*[3], combines the intuitiveness of keyword-based queries with the expressiveness of XML query languages (such as XPath). Given a user keyword query, our method is able to construct a set of XPath expressions that maps all possible interpretations of the user intention to a corresponding structured query. Furthermore, it assigns each XPath expression a score that measures its likelihood of representing the correct interpretation of the user query. Then, a system can submit one or more of these queries to a DBMS and retrieve the results. The process of automatically transforming an unstructured keyword-based query into a ranked set of XPath queries is called *query structuring*.

This paper is organized as follows. Section 2 summarizes related work. Section 3 presents an overview of background concepts and introduces StruX. Section 4 describes how StruX was applied to the experimental tasks proposed for the INEX 2010 data-centric track. Finally, Section 5 presents concluding remarks and future work.

2 Related Work

The basic principle for StruX is similar to LABRADOR [21], which efficiently publishes relational databases on the Web by using a simple text box query interface. Other similar systems for searching over relational data include SUITS [6] and SPARKS [19]. Our proposal is different from those for two reasons: it works for a different type of data (XML instead of relational) and does not need any interaction with the user after she has specified a set of keywords. Therefore, this section focuses on XML keyword search.

[3] The name StruX is derived from the latin verb form *struxi*, which means to structure or build things.

Using keywords to query XML databases has been extensively studied. Current work tackles keyword-based query processing by modeling XML documents as labeled trees and considers that the desired answer is a meaningful part of the XML document, retrieving nodes of the XML graph that contain the query terms. In [11] and [24] the authors suggest that trees of smaller size bear higher importance. XSEarch [5] adopts an intuitive concept of *meaningfully related sets of nodes* based on the relationship of the nodes with its ancestors on the graph. XSEarch also extends the pure keyword-based search approach by allowing the user to specify labels and keyword-labels in the query. XRANK [7] employs an adaptation of Google's PageRank [22] to XML documents and aims at computing a ranking score for the answer trees. A distinct approach is XSeek [18], in which the query processing relies on the notion of entities inferred from DTDs. StruX shares a similar view with XSeek in this regard.

In [12], the authors propose the concept of *mapping probability*, which captures the likelihood of mapping keywords from a query to a related XML element. This mapping probability serves as weight to combine language models learned from each element. This method does no generate structured queries (as our work does). Instead, it uses the structural knowledge to refine a retrieval model.

There are also other works that go beyond simple keyword-based searching. NaLIX [16] is a natural language system for querying XML in which the queries are adjusted by interacting with the user until they can be parsed by the system. Then, it generates an XQuery expression by using Schema-Free XQuery [17], which is an XQuery extension that provides support for unstructured or partially unstructured queries. Another approach, called EASE, considers keyword search over unstructured, semi-structured and structured data [15]. Specifically, EASE builds a schema graph with data graph indexes. It also provides a novel ranking algorithm for improving the search results. Finally, LCARANK [4] combines both SLCA and XRank for keyword-based searching over XML streams. Even though these approaches work on XML keyword-based queries, their goals are slightly different from our work, since they consider broader perspectives (i.e., natural language, graph-oriented data and XML streams).

3 StruX Description

This section presents StruX, our method for generating structured XML queries from an input unstructured keyword-based query. First, it gives an overview of StruX and then it details each of its steps.

3.1 Overview

Algorithm 1 describes StruX general steps, which are illustrated within an example in Fig. 3.1. Given an unstructured keyword-based query as input (Fig. 3.1a), StruX first splits the input sequence of keywords into segments (Fig. 3.1b) and then generates combinations of these segments (Fig. 3.1c) to obtain possible semantic interpretations of the input unstructured query.

Algorithm 1 StruX Processing

```
1:  procedure StruX(Input: unstructured query $U$, schema tree $T$)
2:      $segs \leftarrow GenerateSegments(U)$
3:      $combs \leftarrow GenerateCombinations(segs)$
4:      for each combination $c$ in $combs$ do
5:          $cands \leftarrow GenerateCandidates(c)$
6:          for each candidate $d$ in $cands$ do
7:              $rank \leftarrow CalculateScores(d, S.root)$
8:              $localRank.add(rank)$                          ▷ sorted add
9:          $globalRank.add(localRank)$                         ▷ sorted add
10:     for $i$ from 1 to $k$ do
11:         $topK\_ranks \leftarrow GenerateXPath(globalRank[i])$
12:     $StructuredQuery \leftarrow topK\_xpaths[0]$                ▷ the top query
```

This process assumes that each keyword-based query is an *unstructured query* U composed of a sequence of n terms, i.e., $U = \{t_1, t_2, ..., t_n\}$. This assumption is based on the intuition that the user provides keywords in a certain order. For example, a keyword query "John Clark Greg Austin" is probably intended to represent interest in someone named "John Clark" and also in "Greg Austin". But we cannot say the same for the query "John Austin Greg Clark". Although both queries have the same terms, the order in which they are specified may be used to describe different intentions. Also, this intuition helps StruX dealing with possible ambiguous keywords.

In the next step, segment combinations are labeled with element names available on the target XML database, forming sets of element-segment pairs (Fig. 3.1d), called *Candidate Predicates*. Once these candidate predicates have been formed, StruX finds adequate *entities* for each candidate (Fig. 3.1e). In fact, StruX relies on the concept of *entities* [18, 20] in order to intuitively represent real world objects. For this, it uses a few simple heuristics.

For instance, consider an element y that has multiple sub-elements x, then y is considered as an *entity*. This can be observed by looking at the document schema (or by traversing the document) and verifying that x can occur multiple times within an instance of element y. For example, considering the DTD specification of `author` as "`<!ELEMENT author (book*, curriculum)>`", `book` is a possible entity, while `curriculum` is not.

In addition, we extend the concept of entity by adding another constraint to avoid too specific queries: an element x must have at least one direct descendant to be considered an entity. For example, if `book` is defined as an element with two sub-elements like "`<!ELEMENT book (title, pages)>`", then it is an entity.

StruX identifies in which elements the query keywords are more likely to occur. Then, it computes scores for candidate structured queries. Those scores are necessary to determine which XML query represents more accurately the user's intention. Finally, one or more top ranked structured queries will be evaluated, then returning the results to the users.

(a) ***Unstructured Query***
U = "John Smith XML"

(b) ***Segments***
S_{11} = "John"
S_{12} = "John Smith"
S_{13} = "John Smith XML"
S_{22} = "Smith"
S_{23} = "Smith XML"
S_{33} = "XML"

(c) ***Segment Combinations***
C_1 = {S_{11}} = {"John"}
C_2 = {S_{22}} = {"Smith"}
C_3 = {S_{33}} = {"XML"}
C_4 = {S_{12}} = {"John Smith"}
C_5 = {S_{12},S_{33}} = {"John Smith", "XML"}
C_6 = {S_{11},S_{22},S_{33}} = {"John", "Smith", "XML"}
(among others)

(d) Candidate predicates for *C_5*
D_1 = {<article/author: "John Smith">, <article/title: "XML">}
D_2 = {<article/author: "John Smith">, <procs/title: "XML">}
D_3 = {<article/author: "John Smith">}
D_4 = {<procs/editor: "John Smith">, <procs/title: "XML">}
D_5 = {<procs/editor: "John Smith">, <article/title: "XML">}
D_6 = {<procs/editor: "John Smith">}
(among others)

(e) Entity: /dblp/article
D_1 = {<article/author: "John Smith">, <article/title: "XML">} => /dblp/article[author="John Smith" and title="XML"]
D_3 = {<article/author: "John Smith">} => /dblp/article[author="John Smith"]
Entity: /dblp/procs
D_4 = {<procs/editor: "John Smith">, <procs/title: "XML">} => /dblp/procs[author="John Smith" and title="XML"]
D_6 = {<procs/editor: "John Smith">} => /dblp/procs[editor="John Smith"]

Fig. 1. StruX example.

The final result produced by StruX is a structured query expressed in XPath[4], which specifies patterns of selection predicates on multiple elements that have some specified tree structure relationships. Hence, XML queries are usually formed by (tree) path expressions. Those expressions define a series of XML elements (labels) in which every two elements are related to each other (for example, through a parent-child relationship). Although other works consider recursive schemas, we do not, since this kind of schema is not commonly found on the Web [13]. Also, notice that in this paper, elements are always identified by their complete path to the document root, not only by its tag label.

3.2 Input Transformation

After the general overview, this and the next sections detail the core procedures of StruX. The first step executed by StruX (Algorithm 1, line 2) is to split the input sequence of keywords into segments that represent possible interpretations of the query. A *segment* is a subsequence S_{ij} of terms from an unstructured query U, where S_{ij} contains the keywords from i to j. For example, considering the query U from Fig. 3.1a, the generated segments are in Fig. 3.1b. Notice that, following the intuition discussed in Section 3.1, we assume that users tend to specify related keywords in sequence. This intuition is captured by the segments.

[4] http://www.w3.org/TR/xpath.html

Therefore, sets of tokens that are not in the sequence such as ⟨ "`John`", "`XML`" ⟩ are not considered.

For each segment S_{ij}, StruX retrieves all elements in which *all* segment keywords appear at least once within a single leaf node. Segments that retrieve no elements are called *not viable* and are discarded from the structuring process. For example, the segment S_{23} = ⟨"`Smith`", "`XML`"⟩ would be considered not viable if the database includes no leaf having both "`Smith`" and "`XML`".

In order to evaluate the likelihood of a segment S_{ij} occurring within an element n, StruX uses a function called *Segment-Element Affinity (SEA)*, which is defined by Equation 1:

$$SEA(n, S_{ij}) = \sum_{k=i}^{j} \text{TF-IEF}(n, t_k), \tag{1}$$

where, TF-IEF(n, t_k) measures the relevance of a keyword t_k for an element type n on the XML database.

Such a function is similar to TF-IAF [21], which defines the relevance of a keyword with respect to the values of an attribute in a relational table. Nonetheless, StruX adapts the concept of "relational attributes" to "XML elements type". This new measure is defined by Equation 2:

$$\text{TF-IEF}(n, t_k) = TF(n, t_k) \times IEF(t_k), \tag{2}$$

where each frequency is calculated by Equations 3 and 4, respectively.

$$TF(n, t_k) = \frac{log(1 + f_{nk})}{log(1 + m)} \tag{3}$$

$$IEF(t_k) = log\left(1 + \frac{e}{e_k}\right) \tag{4}$$

In these equations: f_{nk} is the number of occurrences of keyword t_k as element type n, m is the total number of distinct terms that occur in n, e gives the total number of distinct elements types in the XML document, and e_k is the total number of element types in which the term k occurs.

Equation 1 evaluates every segment no matter its number of keywords. Note that a segment with 2 (or more) keywords is intuitively more selective than a segment with a single keyword. For example, S_{13} (from Fig. 3.1b) is more selective than segments S_{11}, S_{22} and S_{33}. Hence, we consider such heuristic and propose an advanced version for function *SEA* in Equation 5, called *Weighted SEA (WSEA)*, in which the number of keywords is used to favor more selective segments.

$$WSEA(n, S_{ij}) = (1 + j - i) \times \sum_{k=i}^{j} \text{TF-IEF}(n, t_k) \tag{5}$$

For representing all possible semantic interpretations of an unstructured query, StruX defines all possible combinations for a set of segments (Algorithm

1, line 3). Moreover, given a combination C_i, a keyword can belong to only one segment. For example, Fig. 3.1c illustrates some of the combinations for the segments in Fig. 3.1b.

For each segment combination C_i, StruX generates all possible sets of element-segment pairs (Algorithm 1, line 5). For example, using combination $C_5 = \{$"John Smith", "XML"$\}$, StruX obtains the sets of element-segment pairs illustrated in Fig. 3.1d, in which each set of pairs D_i is called a *Candidate Predicate*, or simply candidate. Note that $\langle procs/title \rangle$ in D_4 is different from $\langle article/title \rangle$ in D_5 as StruXidentifies elements by their complete path to the root. By the end of the input transformation procedure, the set of candidates is able to represent every possible interpretation of the user's unstructured query. It is now necessary to determine which interpretation is more suitable to represent the original user's intention.

3.3 Candidate Predicates Selection

Once the candidates have been defined, StruX needs to find adequate entities for each candidate (Algorithm 1, lines 7 and 8). This is accomplished by using the recursive function presented in Algorithm 2. This function, called *CalculateScores*, performs a postorder traversal of the schema tree (which is given as input in Algorithm 1). During the traversal, the scores are propagated in a bottom-up fashion.

Algorithm 2 CalculateScores Function

```
1: function CALCULATESCORES($d, node$)  ▷ Input d: candidate, node: node from the XML database
2:     for each child $h$ in $node.children$ do
3:         $CalculateScores(d, h)$
4:     for each element-segment $e$ in $c$ do
5:         if $e.element = node.element$ then
6:             $node.score \leftarrow node.score + e.score$
7:     for each child $h$ in $node.children$ do
8:         $node.score \leftarrow node.score + (\alpha * h.score)$
9:     if $node.score > 0$ AND $node.isEntity()$ then
10:        $Rank.add(root)$
```

The propagation constant α (Algorithm 2, line 8) determines the percentage of a child score that is assimilated on its parent score (bottom-up propagation). As a result of Algorithm 2, there is a rank which is then added to a local rank of entities for each candidate. All local ranks are merged into a sorted global rank (Algorithm 1, line 9).

Each entry in the rank is a structured query, containing a score, an entity element (structural constraint) and a candidate D_i (value-based predicates). The score of a structured query tries to measure how well it represents the users' original intention while writing her query. Through this ranking procedure, StruX is able to determine which interpretations of the keyword-based query are more suitable according to the underlying database.

Next, a structured query can be trivially translated to XPath. Specifically, for each top-k structured query, StruX generates an XPath query statement based on the corresponding entity and the candidate predicate, as illustrated in Fig. 3.1e.

One important final note is that we chose to transform the keyword-based queries to XPath query statements. However, StruX may also be extended in order to consider other XML query languages, such as XQuery.

3.4 Keywords matching element names

So far, we have only discussed how our method addresses matches between keywords from the query and the contents of the XML elements. Indeed, in StruX we regard such a match as the main evidence to be considered when evaluating the relevance of a structured query. However, to handle cases in which keywords match element labels, we use a very simple strategy: we boost the likelihood of all structured queries in which this is observed, by adding a constant α to its score value.

3.5 Indexing

In order to build a structured query from user-provided keywords, StruX relies on an index of terms. This index is defined based on the target database. Specifically, each term is associated with an inverted list containing the occurrences of this term in the elements of the database. Such an association allows the query structuring process to evaluate where a term is more likely to occur within some element. Each term occurrence in an element contains a list of leaves (each one is assigned with a *leaf id*) in which the term occurs. Hence, our method can determine if two or more keywords are likely to occur in a same leaf node.

4 Experiments

Following the INEX experimental protocol, we employed StruX to process the tasks on the data-centric track that considered the IMDB datasets. The execution was organized in *runs*, and each run consists of processing all topics in the track under a certain setup.

Specifically, given a topic T, we first generated an unstructured query U^T for this topic. Next, U^T was given as input to StruX, producing a list of structured queries $S_1^T, S_2^T, \ldots, S_n^T$ as a result. Each S_i^T is associated with a likelihood score, calculated as described in Section 3.3. Then, we executed the top-K structured queries over the IMDB datasets. Different target datasets were considered in each run. The complete description of the runs is presented in Table 1.

In the following, we discuss some details regarding the generation and the processing of the runs.

run	Structured Queries used	Target Datasets
1	top-5	`"movies"`
2	top-10	`"movies"`
3	top-5	`"movies"`, `"actors"`
4	top-5	all except `"other"`
5	top-5	all
6	top-10	all

Table 1. Description of the runs used in the experiments.

Dealing with entities. As we have already explained, StruX aims at generating structured queries that return single entities found in a collection of entities. In the IMDB datasets, every document root, such as `<movie>` and `<person>`, is intuitively an entity. However, StruX inherently considers a root element as not suitable to be an entity. To overcome this, we extend StruX to consider two *virtual root nodes*: (i) `<movies>` that has all `<movie>` elements as its descendants; and (ii) `<persons>` with all `<person>` elements as descendants. With such an extension, `<movie>` and `<person>` elements can now be considered entities.

Generating Queries from Topics. For each given topic from the data-centric track, we generated a keyword-based query to be used as input for StruX. In order to do so, we took the `<title>` field of the topic and applied a few *transformations*. This step is fully automated and aims mostly at dealing with topics specified using natural language expressions, such as: "*romance movies by Richard Gere or George Clooney*", "*Movies Klaus Kinski actor movies good rating*", "*Dogme movies*". Specifically, we applied the following transformations:

i) simple stemming of plural terms, e.g.: movies $\rightarrow$ movie, actors $\rightarrow$ actor;
ii) removal of stop-words, e.g: by, and, to, of;
iii) disregard of terms indicating advanced search operators, such as like "or";
iv) removal of terms preceded by "$-$", indicating exclusion of terms from the answers.

Fig. 2 illustrates a complete example of the whole process including: the `<title>` field of a topic, the corresponding keyword-based query and a path expression generated from it. This particular path expression corresponds to the top-1 structured query generated by StruX. The result obtained from applying this path expression over the IMDB dataset is also presented in the figure in two formats: as an XML sub-tree and using the INEX output format. Next, we detail how this result was obtained.

Processing Structured Queries. The final result for a given run is obtained by processing the top-K structured queries against the target IMDB datasets.

```
          Topic: <title> true story drugs +addiction -dealer </title>
       KB Query: true story drug addiction
Path Expression: /movie[ overview/plot, "true story drug addiction"]
         Result: <movie>
                 <title>Happy Valley (2008)</title>
                 <url>...</url>
                 <overview>
                 ...
                 <plot> ... The real-life true story, Happy Valley ... that
                 have been dramatically affected by prescription drug abuse
                 leading to street drug abuse and addiction</plot>
                 ...
                 </movie>
    INEX Format: 2010012 Q0 1162293 1 2.6145622730255127 ufam2010Run1
                 /movie[1]
```

Fig. 2. Example of the steps involved in processing an INEX data-centric topic with StruX.

This could be performed by using some native XML DBMS such as *eXists-db*[5]. However, for our experiments, we developed a simple XPath matcher, which is used to match a document against a structured query. By doing so, we could directly output the results in the INEX result submission format, instead of having to transform the output provided by the DBMS. Fig. 2 illustrates the result for one of the topics in the track using both formats.

Regarding the scores of the results, as the final answers for the structured queries are produced by an external system (in our case a simple XPath matcher), there is no relevance scores directly associated to them. Thus, we simply assigned to the result the same score StruX has generated for the structured query from which it was obtained. Nonetheless, a single ranking of results is generated for all top-k structured queries. In this ranking, results from the top-1 query occupy the topmost positions, followed by the results from the top-2 query and so on.

5 Conclusions

In this paper we presented a novel method (StruX) with a fresh perspective for keyword-based search over XML data. In summary, StruX combines the intuitiveness of keyword-based queries with the expressiveness of XML query languages. Given a user keyword-based query, StruX is able to construct a set of XPath expressions that maps all possible interpretations of the user intention to a corresponding structured query. We used StruX to perform the tasks proposed in the INEX 2010 data-centric track considering IMDB datasets. The results demonstrated that *query structuring* is feasible and yet effective.

As future work, we plan to optimize even further our method. Specifically, we need to improve StruX capabilities on indexing very large datasets. We also

[5] http://exist.sourceforge.net/

want to study other heuristics for improving the set of the structured queries generated. This should be accomplished by ranking the XML fragments to ensure that results closer to the original user's intention are presented first. Finally, we want to perform experiments with different Segment-Element Affinity (SEA) functions using other Information Retrieval techniques.

Acknowledgements. This work was partially supported by the Brazilian National Institute of Science and Technology for the Web (CNPq grant no. 573871/2008-6), the Amanajé project (CNPq grant no. 47.9541/2008-6), the SAUIM project (CNPq grant no. 554087/2006-5), the MINGroup project (CNPq grant no. 575553/2008-1), and the authors' scholarships and individual grants from CNPq and CAPES.

References

1. B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, P. Parag, and S. Sudarshan. BANKS: Browsing and Keyword Searching in Relational Databases. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 1083–1086, 2002.
2. S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A System for Keyword-Based Search over Relational Databases. In *Proceedings of the 18th International Conference on Data Engineering*, pages 5–16, 2002.
3. A. Balmin, V. Hristidis, and Y. Papakonstantinou. ObjectRank: Authority-Based Keyword Search in Databases. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pages 564–575, 2004.
4. E. G. Barros, M. M. Moro, and A. H. F. Laender. An Evaluation Study of Search Algorithms for XML Streams. *Journal of Information and Data Management*, 1(3):487–502, 2010.
5. S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. In *Proceedings of the 29th International Conference on Very Large Data Bases*, pages 45–56, 2003.
6. E. Demidova, X. Zhou, G. Zenz, and W. Nejdl. SUITS: Faceted User Interface for Constructing Structured Queries from Keywords. In *Proceedings of the International Conference on Database Systems for Advanced Applications*, pages 772–775, 2009.
7. L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 16–27, 2003.
8. V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-style Keyword Search over Relational Databases. In *Proceedings of the 29th International Conference on Very Large Data Bases*, pages 850–861, 2003.
9. V. Hristidis, N. Koudas, Y. Papakonstantinou, and D. Srivastava. Keyword Proximity Search in XML Trees. *IEEE Transactions on Knowledge and Data Engineering*, 18(4):525–539, 2006.
10. V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword Search in Relational Databases. In *Proceedings of 28th International Conference on Very Large Data Bases*, pages 670–681, 2002.

11. V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword Proximity Search on XML Graphs. In *Proceedings of the 19th International Conference on Data Engineering*, pages 367–378, 2003.
12. J. Kim, X. Xue, and W. Croft. A probabilistic retrieval model for semistructured data. *Advances in Information Retrieval*, pages 228–239, 2009.
13. A. H. F. Laender, M. M. Moro, C. Nascimento, and P. Martins. An X-ray on Web-Available XML Schemas. *SIGMOD Record*, 38(1):37–42, 2009.
14. G. Li, J. Feng, J. Wang, and L. Zhou. Effective Keyword Search for Valuable LCAs over XML Documents. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, pages 31–40, 2007.
15. G. Li, J. Feng, J. Wang, and L. Zhou. An Effective and Versatile Keyword Search Engine on Heterogenous Data Sources. *Proceedings of the VLDB Endowment*, 1(2):1452–1455, 2008.
16. Y. Li, H. Yang, and H. V. Jagadish. NaLIX: an Interactive Natural Language Interface for Querying XML. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 900–902, 2005.
17. Y. Li, C. Yu, and H. V. Jagadish. Schema-Free XQuery. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pages 72–83, 2004.
18. Z. Liu and Y. Chen. Identifying Meaningful Return Information for XML Keyword Search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 329–340, 2007.
19. Y. Luo, W. Wang, and X. Lin. SPARK: A Keyword Search Engine on Relational Databases. In *Proceedings of the 24th International Conference on Data Engineering*, pages 1552–1555, 2008.
20. F. Mesquita, D. Barbosa, E. Cortez, and A. S. da Silva. FleDEx: Flexible Data Exchange. In *Proceedings of the 9th ACM International Workshop on Web Information and Data Management*, pages 25–32, 2007.
21. F. Mesquita, A. S. da Silva, E. S. de Moura, P. Calado, and A. H. F. Laender. LABRADOR: Efficiently publishing relational databases on the web by using keyword-based query interfaces. *Information Process Management*, 43(4):983–1004, 2007.
22. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.
23. C. Sun, C. Y. Chan, and A. K. Goenka. Multiway SLCA-based keyword search in XML data. In *Proceedings of the 16th International Conference on World Wide Web*, pages 1043–1052, 2007.
24. Y. Xu and Y. Papakonstantinou. Efficient Keyword Search for Smallest LCAs in XML Databases. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 527–538, 2005.

Inferring Query Pattern for XML Keyword Retrieval

Qiushi Li[1,2] , Qiuyue Wang[1,2], Shan Wang[1,2]

[1]Key Laboratory of Data Engineering and Knowledge Engineering, MOE
[2]School of Information, Renmin University of China, Beijing 100872, China
{qiushili, qiuyuew, swang}@ruc.edu.cn

Abstract. In this paper, we present our experiments with query pattern inference method done on the IMDB XML data set used in the INEX 2010 data-centric track. After getting the node types for each XML node which contains the query keywords, we compute the LCA of the node types. If the LCA is an entity node, it is returned, else, we retrieval the ancestor of the LCA until we find an entity ancestor, or the document root is returned. With the entity LCA, we construct an XML node type subtree rooted at the entity LCA node type. Then we retrieve each XML result under the guidance of the node type subtree. If one of the node types for query keywords changes, we reconstruct the node type subtree. Each result is ranked by the combination of its content and structure scores.

Keywords: XML, keyword search, LCA.

1 Introduction

Most of existing keyword search systems on data-centric XML dataset return query results based upon notions of *Lowest Common Ancestor* (LCA) and its variants, such as LCA with pruning irrelevant nodes [1], node interconnected LCA [2], meaningful LCA (MLCA) [3], smallest LCA (SLCA) [4], valuable LCA (VLCA/CVLCA) [5], and etc. But none of them can cope with the disjunctive queries, for example “Yimou Zhang 2010 2009”. Moreover, all the above methods require the candidate result must contain all the query keywords, but in some cases some keywords are not needed, for example, keyword “by”, “or” in a query “romance movies by Richard Gere or George Clooney”.

In order to cope with such problems, we propose a dynamic LCA-based XML retrieval method. First, for each query keyword, we retrieve a list of XML nodes containing the keyword, and then we get the first node from each list and its node type. We construct a node type subtree rooted at their node type LCA node. With the guiding of LCA node type subtree, one partial or complete XML node subtree is returned as a query result. Each query result is ranked by the combination of its content and structure scores.

2 Data Model and Notions

Each XML document can be modeled as an XML tree and all XML trees can be combined together under a virtual root node. For example, in Fig. 1, we model two XML documents in the IMDB data collection as trees and combine them under the a virtual root node named as “imdb”.

To facilitate presenting our ideas, we formally define the following notions.

(1) **Node Type.** For a node *t*, its node type is the path from *root* to *t*. As shown in Fig. 1, the node type for “movie(2)” is “imdb.movie”. If there is no confusion, we use its last tag name to denote the path. For example, “imdb.movie” is simplified as “movie”.

(2) **Node Instance.** For a given node type *T*, its node instances are all the nodes that have the node type *T*. For example, the node instances for “movie” are node 2.

(3) **Node type *T* contains keyword *k*.** It means that at least one instance node of node type *T* contains keyword *k*. An instance node *t* contains keyword *k* means that *k* occurs in *t*’s textual content.

(4) **Structure Summary Tree.** We create a structure summary tree for the XML dataset according to [10, 11]. Each node in the tree represents a node type, and is annotated with its node identifier and some basic statistical information, such as number of instance nodes and etc. The node identifier is assigned as the pre-order traversal number of the summary tree. The structure summary tree for the XML tree in Fig. 1 is shown in Fig. 2.

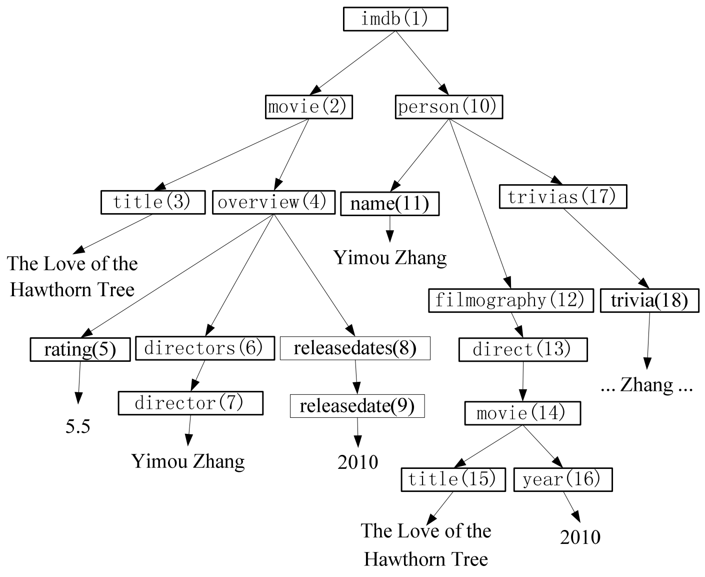


Fig. 1. The data graph for an IMDB XML fragment

(5) **Entity Node** Entity nodes are XML nodes that depict entities in real world and often have one or more attribute nodes, such as “movie(2)”, “person(10)”.

(6) **Attribute Node** Attribute nodes depict some attributes of an entity node, such as “title(3)” and “name(11)” in Figure 1.

(7) **Joint Node** Joint nodes represent neither entities nor attributes, but connect entity or attribute nodes with other entity or attribute nodes, or even other joint nodes. For example, nodes “directors(6)” and “releasedates(8)” are joint nodes in Figure 1. We use the algorithm proposed in [6] to identify entity, attribute and joint nodes.

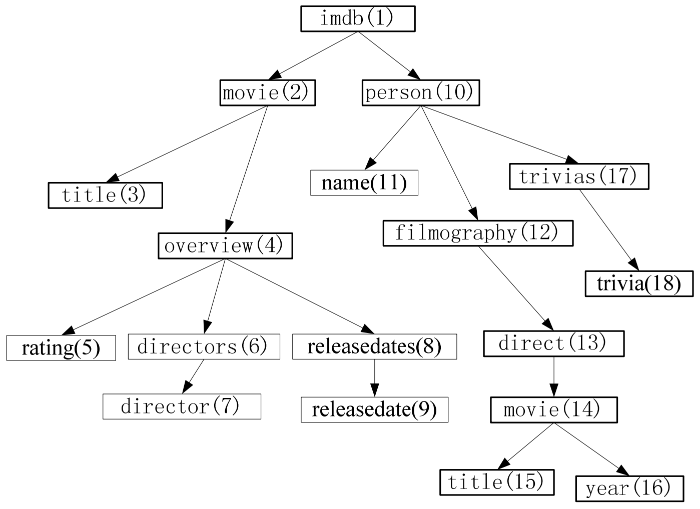


Fig. 2. Structure summary tree for XML data tree in Fig. 1

3 Indexing and Retrieving

3.1 Index Creation

We construct two indexes, XML node index and full-text index. During the index creation, all XML attributes are treated as XML nodes. Each XML node is assigned a unique identifier which represents the pre-order traversing order of the node. The XML node index entry contains information such as the node identifier, Dewey code of the node, node type identifier and other related information.

All XML node values are processed as text, parsed and indexed in a term inverted list. The index data entry for each term includes the term identifier which is a unique integer, total count of the term in the dataset, and a list of XML node identifiers that contain the term and the corresponding position list. The tag name for each XML node is treated as a term and indexed similar to the text terms in an XML node. The only difference is that the position list for an XML tag name has only one data item, -1, which means that the tag name of an XML node is viewed as a term in the XML node value and can occur in any position in the XML node content. If we compute the distance between a tag name and any term in the XML node value, the distance is always equal to 1.

We build a structure summary tree from the dataset according to [10, 11] to facilitate the construction of node type subtree rooted at an LCA node. Each node in the structure summary contains a node type identifier, semantic node type of the node type, such as entity node, attribute node or joint node.

3.2 Retrieve Query Results

We only consider CO queries in our system. CO queries at INEX are given in the title fields of topics. We remove all the signs, i.e. +, -, and quotes, i.e. “” in the title field. That is, a CO query is simply a bag of keywords in our system, $Q = \{w_1, w_2, \ldots, w_m\}$. Query keywords that are qualified by “-” or not existed in the IMDB dataset are ignored.

As shown in the Algorithm 1, we first retrieve a list of XML nodes containing the keyword for each query keyword, and sort the list by the node identifiers. Then we initialize a pointer variable pt_i to the beginning of each node list NL_i (line 1-4). For each loop, we get the node $e_{i,p}$ pointed by pt_i and its node type T_i. If pt_i is null, the node type T_i is the same as that used in the last loop. Now we have a list of node types TL: $\{T_1, T_2, \ldots, T_n\}$. The LCA node type is then computed on the structure summary tree according to the node type list TL (line 6-7).

Definition 1. Query Pattern Given a query $Q(k_1, k_2, \ldots, k_n)$, a query pattern is the node type subtree defined by a node type list TL: $\{T_1, T_2, \ldots, T_n\}$ and their LCA node type T_{lca}.

We construct a node type tree, and associate each tree node with current node type T_x, current node e_x(its initial value is set to null) and the XML node list NL_i if $T_x \in TL$. Then we compute the structural score of the query pattern (line 6-9). For each node type T_i for keyword k_i, we get the node type path from T_i to T_{lca}, then compute the ancestor node of $e_{i,p}$ which is corresponding to node type T_x in the path. Finally, all paths converge at the $T_{lca}.e_x$ which contains as many query keywords as possible (line 10-18). The comparison between $TN.e_x$ and e_p guarantees that $T_{lca}.e_x$ is the smallest query result node (line 14-16). A query result is created from $T_{lca}.e_x$. Because one query result node might contain multiple XML nodes of one query keyword, we use the result node to scan each node list from current position specified by pt_i and bypass the contained node (line 20-25).After the score of a query result has been computed, the result is inserted into the query result list by its score (line 26-28).

Algorithm 1: processQuery
Input: a list of keywords for a query: Q: $\{k_1,k_2,k_3,\ldots k_n\}$.
Output: a list of query results.

```
1:  for each $k_i$ in Q
2:      $NL_i \leftarrow$ retrieve the list of XML nodes for each keyword $k_i$, $NL_i = \{e_{i,1}, e_{i,2}, \ldots e_{i,s}\}$, sort $NL_i$ by
        XML node identifier of pre-traversal number;
3:      set node pointer $pt_i$ to the first node in $NL_i$;
4:  end for
5:  while not(all $pt_i$ is null)
6:      get node $e_{i,p}$ at the position of $pt_i$, $T_i \leftarrow$ get node type of $e_{i,1}$;
7:      $T_{lca} \leftarrow$ compute entity LCA node on the structure summary tree from node type list
        TL:$\{T_1,T_2,\ldots T_n\}$;
8:      create a tree node $TN_x$:$\{T_x, e_x\}$ for each $T_x$ in the node type subtree rooted at $T_{lca}$, initialize
        $TN_x.e_x$ to Null;
9:      $S_s \leftarrow$ compute the structural score of the query pattern according formula (8);
10:     for each $T_i$ in TL
11:             $e_p = e_{i,p}$;
12:             for each tree node $TN_x$ in the node type path from $TN_i$ to $TN_{lca}$;
13:                     $e_p \leftarrow$ compute parent node Parent($e_p$);
14:                     if $TN_x.e_x$ is Null or $e_p < TN_x.e_x$
15:                             $TN_x.e_x = e_p$;
16:                     end if
```

```
17:             end for
18:         end for
19:         create a query result qr from $TN_{lca}.e_x$;
20:         for each $T_i$ in TL
21:             while($e_{i,p}$ is the descendant of $TN_{lca}.e_x$ and $pt_i$ is not Null)
22:                 compute the node score score($e_{i,p}$) according to formula (4);
23:                 move node pointer $pt_i$ to next element in $NL_i$;
24:             end while
25:         end for
26:         $S_c \leftarrow$ compute query result qr's content score according to formula (2);
27:         compute score(qr) according to formula (1);
28:         insert the query result qr into result list by its score;
29:     end while
```

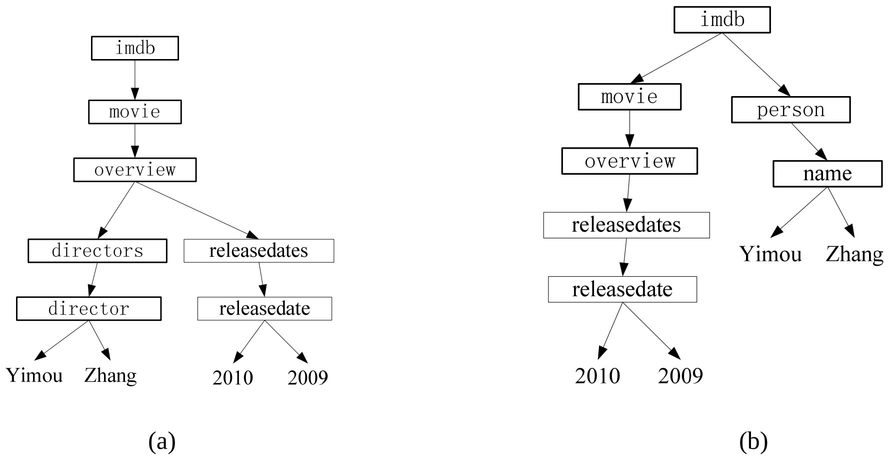


Fig. 3 Query pattern for query "Yimou Zhang 2009 2010

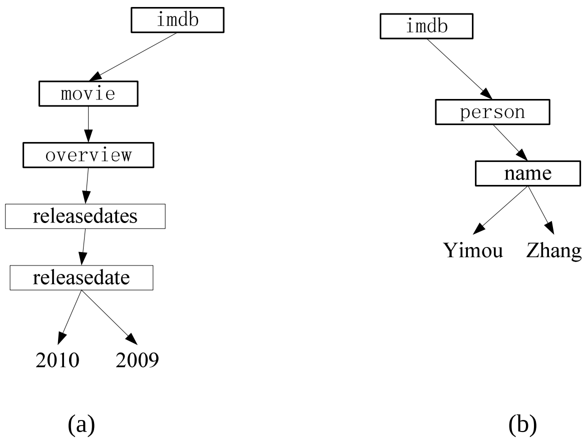


Fig 4 Two query patterns split from Fig. 3(b)

Example: By computing node type LCA subtree, the "or" condition keywords are merged into one node type. For example, "Yimou Zhang 2009 2010", if we compute LCA on the keywords directly, in the most cases, the root "imdb" would be returned as LCA. the node type for "Yimou" and "Zhang" is both T_1:"imdb.movie.directors.director.name" and T_2:"imdb.movie.releasedates.releasedate" for "2009" and "2010", so the node type LCA for "T_1, T_1, T_2, T_2" is "imdb.movie" ("imdb.movie.overview" is not an entity node type), the final query pattern is shown as Fig. 3 (a). We think the relationship among all keywords associated with a node type, such as "2009" and "2010" is "and", but if a query result can not contain both of them, it can still be returned with a decreased ranking score.

In some cases, the node types of "2010" and "2009" are still T_2, but the node type of "Yimou" and "Zhang" is "imdb.person.name", so the node type LCA is the root node type "imdb", as shown in Fig. 3 (b). Because returning the dataset root is meaningless, the query pattern is split into two query patterns with partially satisfied keywords for each pattern, as shown in Fig.4 (a),(b).

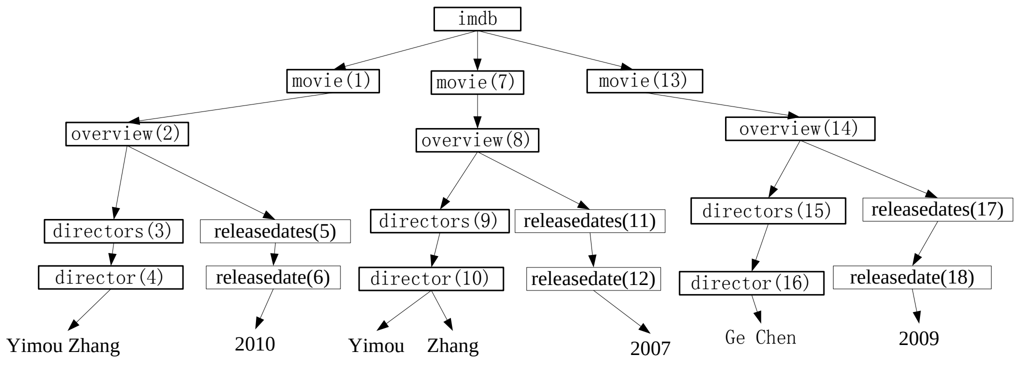


Fig. 5 A sample XML fragment.

For query "Yimou Zhang 2009 2010" and XML data shown in Fig. 5, the XML nodes list is shown in Table 1. For node 4,4,18,6, the query pattern is "imdb.movie" shown in Fig.3(a). we use the query pattern subtree to guide retrieve query results. For "Yimou", we compute its ancestor nodes corresponding "director", "directors", "overview" and "movie", the ancestors are "4, 3, 2, 1" respectively. Then we compute remained query keywords, "Zhang": "4, 3, 2, 1"; "2010": 6, 5, 2, 1; "2009": 18, 17, 14, 13". Because 14 > 2 for "overview", the first query is returned as "1 2 3 4 5 6". We use the result root node 1 to scan each XML node lists of the keywords, the remaining XML node lists are shown in Table 1 (step 1). Going on with the above procedure, we get the remaining query result: "7 8 9 10", "13 14 17 18".

Table 1 Index data items for query "Yimou Zhang 2009 2010"

Query Keywords	**Node List (Step 0)**	**Step 1**	**Step 2**
Yimou	director(4), director(10)	10	

Zhang	director(4), director(10)	10	
2009	releasedate(18)	18	18
2010	releasedate(6)		

4 Ranking Model

The score of a query result is determined by the keywords it contains, the context of keywords occurrence and its node structure, that is, the score of a query result r has two components: its content score S_c and its structure score S_s, as shown in formula (1).

$$Score(r) = S_c(r) * S_s(r) \quad (1)$$

4.1 Content Score of a Query Result

The content score of a query result is equal to the sum of the scores of the XML nodes it contains, as shown in Formula (2), where t_a is a XML node contained by the query result r.

Intuition 1. In one XML node, the score of one node which contains $\{k_1, k_2,..,k_n\}$ should be larger than that of one node which contains $\{k_1, k_2,..,k_{n-1}\}$.

Intuition 2. The score of one XML node containing two keywords should be larger than that of one XML node containing these two keywords but with a larger distance between them.

Based on the Intuition 1 and 2, the score of an XML node which contains query keywords $\{k_1, k_2,..,k_m\}$ is defined in Formula (3) and (4), where $tf(k_i,t_a)$ is the number of occurrence of k_i in t_a, and $|t_a|$ is the textual content length of t_a; k_i and k_j are the keywords that t_a contains and k_i is the keyword which has the maximal Score(k_i, t_a).

$$S_c(r) = \sum_{t_a \in r} Score(t_a) * w(T_a) \quad (2)$$

$$Score(t_a) = Score(k_i, t_a) + \sum_{1 \le j \le m \text{ and } j \ne i} [Score(k_i, t_a) * D(k_i, k_j)] \quad (3)$$

$$Score(k_i, t_a) = tf(k_i, t_a)/|t_a| \quad (4)$$

4.1.1 Importance Weights of Occurring Contexts of Keywords

The content score of an XML node not only depends on the keywords it contains, but also the context of keyword occurrence. A keyword is of different importance if it occurs in different contexts, that is, in different node types. For example, the two occurrences of keyword “Zhang” occurs in “name(11)” and “trivia(18)” should have different importance weights for locating the node “person(10)”. We use the average of inverse node frequency of keywords contained by the node type T_a to measure the importance weights of T_a, as shown in Formula (5). Here T_a is a given node type; K_a denotes the set of unique terms which are contained by T_a; N_a denotes the total number of node instances of T_a; $n_{a,i}$

denotes the number of node instances of T_a that contain the term k_i ($k_i \in K_a$). Being applied with arc-tangent function, $w(T_a)$ are normalized into the range [0, π /2].

$$w(T_a) = 2 * arctan\left(\frac{\sum_{k_i \in K_a}(N_a / n_{a,i})}{|K_a|}\right) / \pi \tag{5}$$

Example: In Figure 1, consider attribute node type "imdb.person.name" (T_a), then K_a = {yimou, zhang} and $|K_a|$ =2; T_a has only one instance nodes, so N_a =1; The numbers of instance nodes of T_a that contain keyword "yimou" and "zhang" are both equal to 1. Thus, $w(T_a)$ =2*arctan [(1/1 + 1/1)/2]/ π = 1/2.

4.1.2 Measure the Proximity of Keywords

The proximity of keywords in ranking query results has been proved in many literatures. We use $Dist(k_i, k_j)$ to measure the proximity of two keywords, as shown in Formula (4). Here d_{01} denotes a threshold given as a user parameter, which shows that when the distance between two keywords is larger than a given distance, their association becomes very little. $d(k_i, k_j)$ denotes the keyword distance between two keywords in one XML node, for example, d("yimou","zhang") in "name(11)" is 1.

$$D(k_i, k_j) = \begin{cases} 2 * arctan(d_{01} - d(k_i, k_j) / \pi + 1, \text{ if } d(k_i, k_j) \le d_{01} \\ 10^{(d_{01} - d(k_i, k_j))}, \text{ if } d(k_i, k_j) > d_{01} \end{cases} \tag{6}$$

4.2 Structural Score of a Query Result

The structural score of a query result is determined by its query pattern, as shown in Formula (7). The score of a query pattern depends on the distance between its node types. Based on the Intuition 3 and 4, the distance is defined as $D(T_i, T_j)$, as shown in Formula (8). Here T_i and T_j both belong to the node type set TL+{T_{lca}} (see Definition 1 for details); $d(T_i, T_j)$ denotes the distance between T_i and T_j and is measured by shortest path distance between T_i and T_j on the structure summary tree and the distance is equal to 0 if T_i=T_j; d_{02} is a user-specified parameter, which means that when $d(T_i, T_j)$ is larger than d_{02}, T_i and T_j are not related and the increase of node number in a query pattern does not increase its structural score, instead, it decreases the structural score of the query pattern.

Intuition 3. For two query patterns, the score of one pattern, which contains n related node types: {T_1,T_2,…T_n}, should be larger than that of one pattern which contains n-1 related node types: {T_1,T_2,…T_{n-1}}.

Intuition 4. The larger the distance between T_i and T_j in a query pattern, the smaller the score of the pattern would be.

$$D(T_i, T_j) = (d_{02} - d(T_i, T_j)) / d_{02} \tag{7}$$

$$S_s(r)= \Sigma D(T_i,T_j) \quad (8)$$

4 Experiments and Results Analysis

We implemented our system with C++ and Berkeley DB 5.0.26. All the experiments were performed on a 3.6 GHz duo-core Intel Xeon machine running Windows Server 2008 with 5GB memory.

5 Conclusion

References

1. Ziyang Liu, Yi Chen: Reasoning and identifying relevant matches for XML keyword search. PVLDB 1(1):921-932 (2008)
2. Guoliang Li, Jianhua Feng, Jianyong Wang, Lizhu Zhou: Effective keyword search for valuable lcas over xml documents. CIKM 2007:31-40
3. Sara Cohen, Jonathan Mamou, Yaron Kanza, Yehoshua Sagiv: XSEarch: A Semantic Search Engine for XML. VLDB 2003:45-56
4. V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword Proximity Search on XML Graphs, In ICDE2003.
5. G. Koutrika, A. Simitsis, and Y. E. Ioannidis. Pr´ecis: The Essence of a Query Answer. In ICDE, page 69, 2006.
6. Ziyang Liu, Yi Chen: Identifying meaningful return information for XML keyword search. SIGMOD 2007:329-340.
7. Zhifeng Bao, Tok Wang Ling, Bo Chen, Jiaheng Lu: Effective XML Keyword Search with Relevance Oriented Ranking. ICDE 2009:517-528.
8. Jiang Li and Junhu Wang, Effectively Inferring the Search-for Node Type in XML Keyword Search. DASFAA 2010.
9. Jianxin Li, Chengfei Liu, Rui Zhou, Wei Wang: Suggestion of promising result types for XML keyword search. EDBT 2010:561-572
10. C. Yu and H. V. Jagadish. Schema Summarization. In Proceedings of VLDB, 2006.
11. Roy Goldman and Jennifer Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. VLDB 1997.

BUAP: A First Approach to the Data-Centric track of INEX 2010*

Darnes Vilariño, David Pinto, Carlos Balderas[1], Mireya Tovar, Saul León
darnes,dpinto,mtovar@{cs.buap.mx}, [1]charlie_kanon@hotmail.com

Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla, México

Abstract. In this paper we present the results of the evaluation of an information retrieval system constructed in the Faculty of Computer Science, BUAP. This system was used in the following track of the Initiative for the Evaluation of XML retrieval (INEX 2010): Data-Centric. This track if focused on the extensive use of a very rich structure of the documents beyond the content. We have considered topics (queries) in two variants: Content Only (CO) and Content And Structure (CAS) of the information need. The obtained results are shown and compared with those presented by other teams in the competition.

1 Introduction

Current approaches proposed for keyword search on XML data can be categorized into two broad classes: one for document-centric XML, where the structure is simple and long text fields predominate; the other for data-centric XML, where the structure is very rich and carries important information about objects and their relationships [1]. In previous years, INEX focuses on comparing different retrieval approaches for document-centric XML, while most research work on data-centric XML retrieval cannot make use of such a standard evaluation methodology. This new track proposed at INEX 2010 aims to provide a common forum for researchers or users to compare different retrieval techniques on data-centric XML, thus promote the research work in this field [2].

Compared to traditional information retrieval, where whole documents are usually indexed and retrieved as single complete units, information retrieval from XML documents creates additional retrieval challenges.

Until recently, the need for accessing the XML content has been addressed diferently by the database (DB) and the information retrieval (IR) research communities. The DB community has focussed on developing query languages and eficient evaluation algorithms used primarily for data-centric XML documents. On the other hand, the IR community has focussed on document-centric XML documents by developing and evaluating techniques for ranked element retrieval.

* This work has been partially supported by the CONACYT project #106625, as well as by the PROMEP/103.5/09/4213 grant.

Recent research trends show that each community is willing to adopt the well-established techniques developed by the other to efectively retrieve XML content [3].

The track uses the IMDB data collection newly built from www.imdb.com. It consists of information about more than 1,590,000 movies and people involved in movies, e.g. actors/actresses, directors, producers and so on. Each object is richly structured. For example, each movie has title, rating, directors, actors, plot, keywords, genres, release dates, trivia, etc.; and each person has name, birth date, biography, filmography, and so on.

The Data-Centric track aims to investigate techniques for finding information by using queries considering content and structure. Participating groups have contributed to topic development and evaluation, which will then allow them to compare the effectiveness of their XML retrieval techniques for the data-centric task. This will lead to the development of a test collection that will allow participating groups to undertake future comparative experiments.

2 Description of the system

In this section we describe the manner we have indexed the corpus provided by the task organizers. Moreover, we present the algorithms developed for tackling the problem of searching information based on structure and content.

For the presented approach we have used an inverted index tree in order to store the XML templates of the corpus. We have considered to include both, the term and the XML tag in the dictionary of the inverted index. The posting list contains the reference of the document (document ID) and the frequency of the indexed term in the given context (according to the XML tag). The original XML file has been previously processed in order to eliminate stopwords and punctuation symbols. Moreover, we have traduced the original hierarchical structure given by XML tags to a similar representation which may be easily analyzed by our parser. In Figure 1 we may see an example of an XML file,whereas its pre-processed version is presented in Figure 2.

In Figure 3 we may observe the inverted index calculated on the basis of the XML files provided as corpus. We have integrated, as previously mentioned, the dictionary by using the XML tag (the last one in the hierarchy) and the indexed term (the number that follows the term is the document frequency of that term). The aim was to be able to localize the correct position of each term in the XML hierarchy and, therefore, to be able to retrieve those parts of the XML file containing the correct answer of a given query. In this way, the inverted index allows to store the same term which occurs in different contexts. We assumed that the last XML tag would be enough for identifying the complete path in which the term occurs, however, it would be better to use the complete hierarchy in the dictionary. Further experiments must verify this issue.

In the following subsection we present the algorithms developed for indexing and searching information based on content and structure.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE article SYSTEM "../movie.dtd">
<movie xmlns:xlink="http://www.w3.org/1999/xlink">
   <title>Hannibal (2001)</title>
   <url>http://www.imdb.com/Title?Hannibal (2001)</url>
   <overview>
      <directors>
         <director>Scott, Ridley</director>
      </directors>
      <writers>
         <writer>Harris, Thomas (I)</writer>
      </writers>
      <genres>
         <genre>Crime</genre>
 <genre>Thriller</genre>
      </genres>
   </overview>
   <cast>
      <actors>
         <actor>
            <name>Hopkins, Anthony</name>
            <character>Hannibal Lecter  &lt;1&gt;</character>
         </actor>
         <actor>
            <name>Moore, Julianne (I)</name>
            <character>Clarice Starling  &lt;2&gt;</character>
         </actor>
      </actors>
   </cast>
   <additional_details>
      <aliases>
         <alias>The Silence of the Lambs 2 [...]</alias>
      </aliases>
   </additional_details>
</movie>
```

Fig. 1. Example of an XML file (1161611.xml)

```
1161611::movie::title::hannibal 2001
1161611::movie::overview::directors::director::scott ridley
1161611::movie::overview::writers::writer::harris thomas
1161611::movie::overview::genres::genre::crime
1161611::movie::overview::genres::genre::thriller
1161611::movie::cast::actors::actor::name::hopkins anthony
1161611::movie::cast::actors::actor::character::hannibal lecter 1
1161611::movie::cast::actors::actor::name::moore julianne
1161611::movie::cast::actors::actor::character::clarice starling 2
1161611::movie::additional_details::aliases::alias::silence lambs [...]
   :                            :
```

Fig. 2. Pre-processed version of an XML file (1161611.xml)

```
title hannibal 40      : 981731:1 994171:1  78811:1 [...] 1161611:1 [...]

character hannibal 440 : 959641:1 959947:1  1161611:1 969446:1 [...]

name hopkins 3068      : 1154469:1 1154769:2 1154810:1 [...] 1161611:1 [...]

name anthony 31873     : 943773:1 [...] 944137:2 1161611:1 1224420:3 [...]

director scott 4771    : 1157203:1 1157761:1 1157773:1 [...] 1161611:1 [...]

director ridley 62     : 1289515:1 1011543:1 1011932:1 [...] 1161611:1 [...]

writer harris 2114     : 1120749:1 1121040:1 1121294:1 [...] 1161611:1 [...]

writer thomas 7333     : 115985:1 115986:1 [...] 1161611:1 1161616:2 [...]

       :                                             :
```

Fig. 3. Inverted index of the corpus in which appears the XML file (1161611.xml)

2.1 Data processing

Before describing the indexing techniques used, we firstly describe the manner we have processed the data provided for the competition. We have cleaned the XML files in order to obtain a easier way of identifying the tag for each data. For this purpose, as we mentioned beforehand, we have traduced the original hierarchical structure given by XML tags to a similar representation which may be easily analyzed by our parser. Thereafter, we have created five different inverted indexes, for the each one of the following categories: actors, directors, movies, producers and others. The inverted index was created as mentioned in the previous section.

Once the dataset was indexed we may be able to respond to a given query. In this case, we have also processed the query by identifying the corresponding logical operators (AND, OR). Let us consider the query presented in Figure 4, which is then traduced to the sentence shown in Figure 5. The first column is the topic or query ID; the second column is the number associated to the ct_no tag; the third column indicates the number of different categories that will be processed, in this case, we are considering only one category: movies.

```
<topic id="2010001" ct_no="3">
  <title>Yimou Zhang 2010 2009</title>
  <castitle>//movie[about(.//director, "Yimou Zhang") and
                    (about(.//releasedate, 2010) or
                     about(.//releasedate, 2009))]
  </castitle>
  <description>I want to know the latest movies directed by Yimou Zhang.
  </description>
  <narrative>I am interested in all movies directed by Yimou Zhang, and
             I want to learn the latest movies he directed.
  </narrative>
</topic>
```

Fig. 4. An example of a given query (topic)

```
2010001 3 1 //movie//director yimou zhang and //movie//releasedate 2010
            or //movie//releasedate 2009
```

Fig. 5. Representation of the topic

In the competition we submitted two runs. The first one considered the topics as mentioned in the previous paragraph. A second approach considered to split the data that corresponds to each content into unigrams, with the goal of being more specific in the search process. However, as will be seen in the experimental results section, both approaches perform similar. An example of topic showing the second approach is given in Figure 6, whereas its traduced version is given in Figure 7.

```
<topic id="2010025" ct_no="19">
  <title>tom hanks steven spielberg</title>
  <castitle>//movie[about(., tom hanks steven spielberg)]</castitle>
  <description>movies where both tom hanks and steven spielberg worked
              together
   </description>
  <narrative>The user wants all movies where Tom Hanks and Steven
             Spielberg have worked together (as actors, producers,
             directors or writers). A relevant movie is a movie
             where both have worked together.
  </narrative>
</topic>
```

Fig. 6. An example of another topic

```
2010025 19 1 //movie tom and //movie hanks and //movie steven and
             //movie spielberg
```

Fig. 7. The reprentation of a topic by splitting the data

In order to obtain the list of candidate documents for each topic, we have used the following equations used as similarity measures:

$$\mathrm{CR}(c_q, c_d) = \begin{cases} \frac{1+|c_q|}{1+|c_d|} & \text{if } c_q \text{ matches } c_d \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where c_q and c_d are the number of nodes in the query path and document path.

The similarity score between a topic and each corpus document is calculated as shown in Eq. (2), which was implemented as presented in Algorithm 1. All these techniques have been extracted from [4].

$$\mathrm{SIM}(q,d) = \sum_{c_k \in B} \sum_{c_l \in B} CR(c_k, c_l) \sum_{t \in V} weight(q,t,c_k) \frac{weight(d,t,c_l)}{\sqrt{\sum_{c \in B, t \in V} weight^2(d,t,c)}} \tag{2}$$

where V is the vocabulary of non-structural terms; B is the set of all XML contexts; and $weight(q,t,c)$ and $weight(d,t,c)$ are the weights of term t in XML context c in query q and document d, respectively.

Algorithm 1: Scoring of documents given a topic q

```
   Input: $q$, $B$, $V$, $N$, $normalizer$
   Output: $score$
1  for $n = 1$ to $N$ do
2      $score[n] = 0$
3      foreach $\langle c_q, t\rangle \in q$ do
4          $w_q = \text{Weight}(q, t, c_q)$
5          foreach $c \in B$ do
6              if $CR(c_q, c) > 0$ then
7                  $postings = \text{GetPostings}(c, t)$
8                  foreach $posting \in postings$ do
9                      $x = CR(c_q, c) * w_q * \text{PostingWeight}(posting)$
10                     $score[docID(posting)] += x$
11                 end
12             end
13         end
14     end
15 end
16 for $n = 1$ to $N$ do
17     score[n] = score[n]/normalizer[n]
18 end
19 return $score$
```

2.2 Experimental results

We have evaluated 25 topics with the corpus provided by the competition organizers. This dataset is made up of 1,594,513 movies, 1,872,492 actors, 129,137 directors, 178,117 producers and, finally, 643,843 files categorized as others.

As it was mentioned before, we submitted two runs which we have named: "FCC-BUAP-R1" and "FCC-BUAP-R2". The former uses the complete data of each record (n-gram), whereas the latter split the words contained in the query by unigrams. The obtained results when evaluating the task as focused retrieval (MAgP measure) may be seen in Table 1 and in Figure 8.

As may be seen, we have obtained a very poor performance, which we consider is derived of the fact of using only one tag for identifying each indexed term. We assumed that the last XML tag in each context would be enough for identifying the complete path in which the term occurs, however, it would be better to use the complete hierarchy in the dictionary. Once the gold standard is being released, we are considering to carry out more experiments in order to verify this issue.

The evaluation of the different runs at the competition measured as document retrieval may be found in Table 2.

#	MAgP	Institute	Run
1	0.24910409	University of Otago	OTAGO-2010-DC-BM25
2	0.24585548	Universitat Pompeu Fabra	UPFL15TMI
3	0.24337897	Universitat Pompeu Fabra	UPFL15TMImov
4	0.18113477	Kasetsart University	NULL
5	0.15617634	University of Otago	OTAGO-2010-DC-DIVERGENCE
6	0.06517544	INDIAN STATISTICAL INSTITUTE	ISI_fdbk_em_10
7	0.0587039	INDIAN STATISTICAL INSTITUTE	ISI_nofdbk_article_.4.-1
8	0.04490731	INDIAN STATISTICAL INSTITUTE	ISI_nofdbk_article_.4.0
9	0.04426635	INDIAN STATISTICAL INSTITUTE	ISI_fdbk_10
10	0.04091211	B. Univ. Autonoma de Puebla	FCC-BUAP-R1
11	0.04037697	B. Univ. Autonoma de Puebla	FCC-BUAP-R2
12	0.03788804	INDIAN STATISTICAL INSTITUTE	ISI_nofdbk_article_.6.0
13	0.03407941	INDIAN STATISTICAL INSTITUTE	ISI_nofdbk_article_.4.1
14	0.02931703	INDIAN STATISTICAL INSTITUTE	ISI_nofdbk_article_.2.0

Table 1. Evaluation measured as focused retrieval (MAgP)

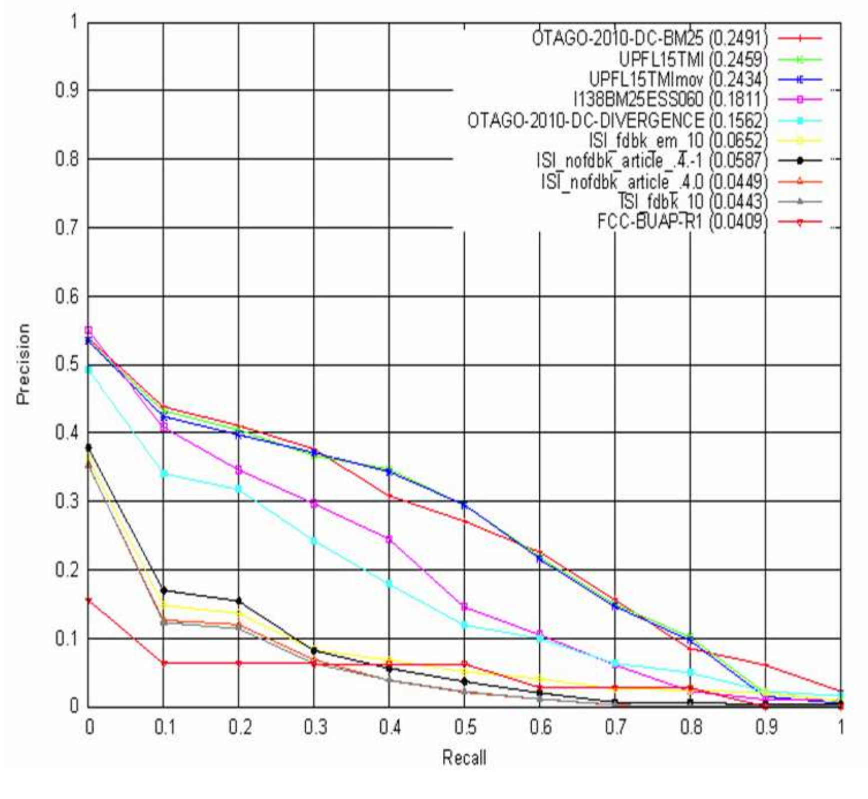


Fig. 8. Evaluation measured as focused retrieval (MAgP)

#	MAP	Institute	Run
1	0.5046	SEECS, Peking University	NULL
2	0.5046	SEECS, Peking University	NULL
3	0.3687	Universitat Pompeu Fabra	UPFL15TMI
4	0.3542	Universitat Pompeu Fabra	UPFL15TMImov
5	0.3397	University of Otago	OTAGO-2010-DC-BM25
6	0.2961	Universitat Pompeu Fabra	UPFL15Tall
7	0.2829	Universidade Federal do Amazonas	ufam2010Run2
8	0.2822	Universitat Pompeu Fabra	UPFL45Tall
9	0.2537	Universidade Federal do Amazonas	ufam2010Run1
10	0.2512	Universidade Federal do Amazonas	ufam2010Run5
11	0.2263	Universidade Federal do Amazonas	ufam2010Run3
12	0.2263	Universidade Federal do Amazonas	ufam2010Run4
13	0.2263	Universidade Federal do Amazonas	ufam2010Run5
14	0.2103	University of Otago	OTAGO-2010-DC-DIVERGENCE
15	0.2044	Kasetsart University	NULL
16	0.1983	Universitat Pompeu Fabra	UPFL15Tmovie
17	0.1807	INDIAN STATISTICAL INSTITUTE	ISI_elts.0
18	0.18	INDIAN STATISTICAL INSTITUTE	ISI_elts.1
19	0.1783	INDIAN STATISTICAL INSTITUTE	ISI_elts.-1
20	0.1578	Universitat Pompeu Fabra	UPFL45Tmovie
21	0.1126	INDIAN STATISTICAL INSTITUTE	ISI_fdbk_em_10
22	0.0888	INDIAN STATISTICAL INSTITUTE	ISI_nofdbk_article_.4.-1
23	0.0674	INDIAN STATISTICAL INSTITUTE	ISI_nofdbk_article_.4.0
24	0.0672	INDIAN STATISTICAL INSTITUTE	ISI_fdbk_10
25	0.0602	B. Univ. Autonoma de Puebla	FCC-BUAP-R2
26	0.0581	INDIAN STATISTICAL INSTITUTE	ISI_nofdbk_article_.6.0
27	0.0544	B. Univ. Autonoma de Puebla	FCC-BUAP-R1
28	0.0507	INDIAN STATISTICAL INSTITUTE	ISI_nofdbk_article_.4.1
29	0.0424	INDIAN STATISTICAL INSTITUTE	ISI_nofdbk_article_.2.0

Table 2. Evaluation measured as document retrieval (whole document retrieval)

3 Conclusions

In this paper we have presented details about the implementation of an information retrieval system which was used to evaluate the task of focused retrieval of XML documents. The implemented system was used in the Data-Centric track of the Initiative for the Evaluation of XML retrieval (INEX 2010).

We presented a indexing method based on inverted index with XML tags embedded. For each category (movies, actors, producers, directors and others), we constructed and independent inverted index. The dictionary of the index considered both, the category and the indexed term which we assumed to be sufficient to correctly identify the specific part of the XML file associated to the topic.

Based on the low scores obtained, we may conclude that a more detailed description in the dictionary (including more tags of the XML hierarchy) is needed in order to improve the precision of the information retrieval system presented.

References

1. Wang, Q., Li, Q., Wang, S., Du, X.: Exploiting semantic tags in xml retrieval. In: In Proc. of the INEX 2009. (2009) 133–144
2. Wang, Q., Trotman, A.: Task description of inex 2010 data-centric track. In: In Proc. of INEX 2010 (same volume). (2010)
3. Amer-Yahia, S., Curtmola, E., Deutsch, A.: Flexible and efficient xml search with complex full-text predicates. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data. SIGMOD '06, New York, NY, USA, ACM (2006) 575–586
4. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK (2009)

UPF at INEX 2010
Towards Query-type based Focused Retrieval

Georgina Ramírez

Universitat Pompeu Fabra, Barcelona, Spain
georgina.ramirez@upf.edu

Abstract. This paper describes our participation at INEX 2010. We participated in two different tracks: ad-hoc and data-centric. In the first one we study the performance effects of changing the article order (fetching phase) and experiment with query-type based retrieval. We present on-going work on INEX topics classification and an analysis of the characteristics of the relevance assessments for each of the topic classes. The goal of our study is to be able to use different retrieval strategies depending on the topic type. In the data-centric track we experiment with the use of different indices and retrievable element types. Our main finding is that indexing uniquely the movie documents performs much better than indexing the complete collection.

Keywords: XML, focused retrieval, INEX, query-based retrieval

1 Introduction

2 Ad-hoc track

This section describes our approach for the Ad-hoc track. We propose a classification of INEX topic types and present an analysis of the characteristics of the relevance assessments for each of the topic classes. In Subsection 2.3 we discuss different retrieval strategies that could be used for each of the topic types.

2.1 Topic classification

In this subsection we propose a classification of INEX topics. We extend our previous work on INEX information needs classification [2] with a new dimension: the type of information sought.

Our classification uses then the following three dimensions: 1) type of information sought (general or restricted), 2) Specificity of the topic (generic or specific), and 3) Complexity of the topic (simple or compound). Topics that are restricted regarding the type of information sought can be further divided according to the type of restriction (topical or structural).

In the *complexity* dimension, two categories are used: *Simple* (S) and *Compound* (C). *Simple* requests are those that ask for information about just one

Table 1. Example of topic descriptions from INEX 2009 belonging to each of the topic type categories.

Complex.	**Specific.**	**Info sought**	**Example**
Simple	Generic	General	Information about Nobel prize.
		Restricted	information about yoga **lesson (topically).**
	Specific	General	Information of classical movies.
		Restricted	I want to find out what **cars the female rally drivers prefer (topically).**
Compound	Generic	General	Find information about applications of bayesian networks in the field of bioinformatics.
		Restricted	Find **bands of folk metal coming from Finland (topically).**
	Specific	General	I am looking for solar power facilities in Europe.
		Restricted	Explain "mean average precision" and "reciprocal rank" with **images or plots.** Provide **references in proceedings and journals (structurally).**

topic or aspect of the topic (i.e., mono-faceted requests). While *Compound* requests are those that ask for information about several topics or several aspects of the same topic (i.e., multifaceted requests) or want information about the relationship between two topics (e.g. technique A in the field of B or information about A for B).

In the *specificity* dimension, we classify requests into *Specific* (Sp) and *Generic* (G), depending on the broadness of the information being searched for.

Regarding the type information sought, we classify requests into *General* (Gr) and *Restricted* (R). *General* requests are those that simply ask for information about a topic, in a general way, without any type of constraint. *Restricted* requests are those which specify any type of constraint on the type of information being sought. This type of restriction can be topical (e.g., exercises, experiments) or structural (e.g., references).

Table 1 shows the resulting categories and an example of each of them.

2.2 Relevance assessments analysis

2.3 Topic-based search strategies

3 Experiments

This Section describes the setup and discusses the results of the experiments carried out for both tracks. For all our experiments we have used the Indri Search Engine [1]. Indri uses a retrieval model based on a combination of language modeling and inference network retrieval frameworks. We have used linear smoothing and varying lambda value.

3.1 Ad-hoc track experiments

For the ad-hoc track experiments we have used the Indri search engine [1] with linear smoothing and lambda 0.45. The lambda value has been set to 0.45 after training on the INEX Wikipedia 2009 collection. The only indexed fields are articles, sections, and paragraphs, meaning that only these element types can be explicitly retrieved. Our first set of experiments in each of the subtasks study the importance of the fetching phase, i.e., the performance effects of changing the article order. The second set of experiments experiment with query based retrieval. We investigate the performance of different retrieval strategies for each of the query types described in Section 2[1].

Relevant in Context The aim of the Relevant in Context Task is to first identify relevant articles (the fetching phase), and then to identify the relevant results within the fetched articles (the browsing phase). As mention above, we first experiment with the performance effects of the fetching phase. For that, we use the same baseline run and reorder its articles in three different ways. Our baseline is a paragraph run (retrieving only paragraphs) grouped by article. The final article order is then given by 1) our own article run order (retrieving only articles), 2) the reference run order, 3) the baseline run order (i.e., the article where the most relevant paragraph appears, followed by the article were the second most relevant paragraph appears, etc.).

Unfortunately, due to a bug in our code, our official runs were not correct and we are not able to show the results of the new runs yet. We hope to do so in the final version of this paper.

Restricted Relevant in Context The Restricted Relevant in Context Task is a variant of the Relevant in Context task, where only 500 characters per article are allowed to be retrieved. Overlapping results are not permitted. For this task, we have followed a similar approach to the one of our Relevant in Context runs. This time however, our baseline is a section run (retrieving only sections) and a second post-processing step is made in order to return only 500 characters per article. That is, per each article we return the most relevant sections until the 500th character is reached. As in the previous task, the final article order is given by 1) our own article run order (retrieving only articles), 2) the reference run order, 3) the baseline run order (i.e., the article where the most relevant section appears, followed by the article were the second most relevant section appears, etc.).

Unfortunately, due to a bug in our code, our official runs were not correct and we are not able to show the results of the new runs yet. We hope to do so in the final version of this paper.

Restricted Focused The Restricted Focused task aims at giving a quick overview of the relevant information in the whole Wikipedia. Results are restricted to max. 1,000 characters per topic. For this task, we return a single paragraph per article (the most relevant) until we reach the 1000 characters per topic. The assumption is that users prefer to see an overview based on the most number of articles rather than seeing several relevant paragraphs of the same article. Our three official runs are again based on the different article order as in the previous tasks; The final article order is given by 1) our own article run

[1] At the moment of writing this paper the experiments on query based retrieval are not complete. We will discuss them in the final version of this paper.

order (retrieving only articles), 2) the reference run order, 3) the baseline run order (i.e., the article where the most relevant paragraph appears, followed by the article were the second most relevant paragraph appears, etc.).

The results of these runs are shown in Table 2.

Table 2. Official results for the restricted focused runs (measured as focused retrieval)

run name (number)	char prec	run position
UPFpLM45co (1)	0.3066	15
UPFpLM45co (3)	0.2984	19
UPFpLM45co (2)	0.1156	30

We can see that the article order is an important factor on the overall result of the run. There is a big difference in terms of performance from our article and paragraph runs order and the reference run order.

3.2 Data-centric track experiments

For our data-centric track experiments we used the Indri search engine [1] with linear smoothing and two different lambdas, 0.45 and 0.15. Since this is a new collection and we did not have training data to optimize lambda, we experiment with two different values that have been successfully used in other collections. We also experiment with the use of two different indices (indexing all the collection vs. indexing only movies) and by restricting the type of elements to be retrieved (no restriction vs. movie elements)[2].

Table 3 shows the parameters used for each of our official runs and Table 4 the official results.

Table 3. Official runs for the data-centric track

run name	index	retrievable elements	lambda
UPFL15Tall	all	no restriction	0.15
UPFL45Tall	all	no restriction	0.45
UPFL15Tmovie	all	movie	0.15
UPFL45Tmovie	all	movie	0.45
UPFL15TMI	movies	no restriction	0.15
UPFL15TMImov	movies	movie	0.15

Our best performing runs are the ones that use the movie index, indicating that, for this specific topic set, the use of other types of documents introduces

[2] Note that movie elements can have very different forms: from a complete movie document to a movie element within a list of movies played by an actor.

Table 4. Official results for the data-centric track. The number in parenthesis indicates the run position in the official ranking.

run name	MAgP	MAiP	Document Retrieval
UPFL15Tall	-	0.1486 (7)	0.2961 (6)
UPFL45Tall	-	0.1338 (11)	0.2822 (8)
UPFL15Tmovie	-	0.0770 (20)	0.1983 (16)
UPFL45Tmovie	-	0.0410 (24)	0.1578 (20)
UPFL15TMI	**0.2459 (2)**	**0.1809 (2)**	**0.3687 (3)**
UPFL15TMImov	0.2434 (3)	0.1762 (3)	0.3542 (4)

noise. We also see that lambda 0.15 performs always better than lambda 0.45, indicating that it is better to give less emphasis to the collection statistics. Figure 1 show the official graphs. In general terms we can see that using the movie index (our best runs) leads to high precision at early recall levels while, not surprisingly, does not manage to do so at middle and/or high recall ones (MAiP and MAP graphs). This is because a big part of the collection is not indexed so it is difficult (if not impossible) to have a high overall recall.

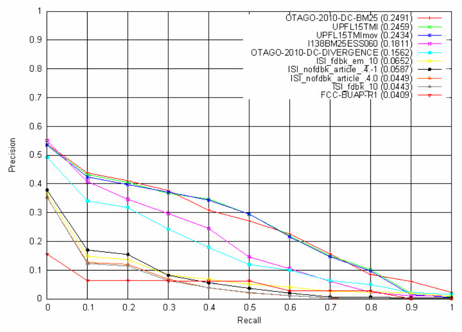
4 Conclusions

This paper described our participation at INEX 2010, in the ad-hoc and data-centric tracks. We presented on-going work on INEX topics classification and the analysis of the characteristics of the relevance assessments for each of the topic classes. In the data-centric track, we experimented with the use of different indices and retrievable element types. Our main finding is that indexing uniquely the movie documents performs much better than indexing the complete collection.

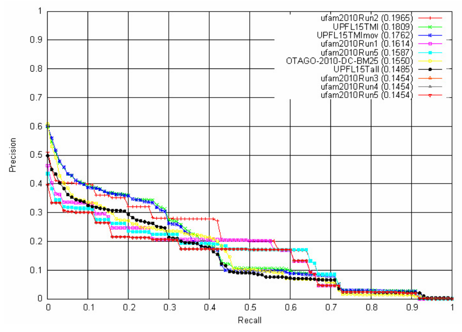
Acknowledgments This work has been supported by the Spanish Ministry of Science and Education under the HIPERGRAPH project and the Juan de la Cierva Program.

References

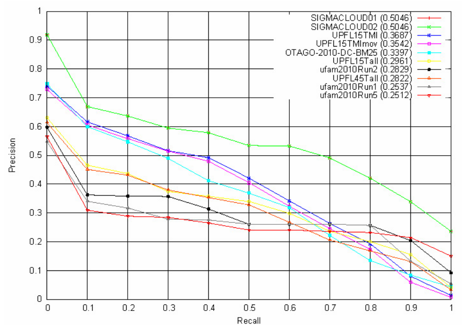
1. T. Strohman, D. Metzler, H. Turtle, and W. B. Croft Indri: a language model based search engine for complex queries Proceedings of the International Conference on Intelligent Analysis, 2005.
2. Georgina Ramírez Camps Structural Features in XML Retrieval PhD thesis, University of Amsterdam, 2007.



MAgP



MAiP



MAP

Fig. 1. Official evaluation graphs for the data-centric track

The INEX 2010 Interactive Track: an overview

Nils Pharo[1], Thomas Beckers[2], Ragnar Nordlie[1] and Norbert Fuhr[2]

[1]Faculty of Journalism, Library and Information Science, Oslo University College, Norway
nils.pharo@jbi.hio.no, ragnar.nordlie@jbi.hio.no
[2]Department of Computer Science and Applied Cognitive Science, University of Duisburg-Essen, Germany
tbeckers@is.inf.uni-due.de, norbert.fuhr@uni-due.de

Abstract. In the paper we present the organization of the INEX 2010 *interactive track*. For the 2010 experiments the iTrack has gathered data on user search behavior in a collection consisting of book metadata taken from the online bookstore Amazon and the social cataloguing application LibraryThing. The collected data represents traditional bibliographic metadata, user-generated tags and reviews and promotional texts and reviews from publishers and professional reviewers. In this year's experiments we designed two search task categories, which were set to represent two different stages of work task processes. In addition we let the users create a task of their own, which is used as a control taks. In the paper we describe the methods used for data collection and the tasks performed by the participants.

1 Introduction

The INEX interactive track (iTrack) is a cooperative research effort run as part of the INEX Initiative for the Evaluation of XML retrieval [1]. The overall goal of INEX is to experiment with the potential of using XML to retrieve relevant parts of documents. In recent years, this has been done through the provision of a test collection of XML-marked Wikipedia articles. The main body of work within the INEX community has been the development and testing of retrieval algorithms. Interactive information retrieval (IIR) [2] aims at investigating the relationship between end users of information retrieval systems and the systems they use. This aim is approached partly through the development and testing of interactive features in the IR systems and partly through research on user behavior in IR systems. In the INEX iTrack the focus over the years has been on how end users react to and exploit the potential of IR systems that facilitate the access to *parts* of documents in addition to the full documents.

The INEX interactive track was run for the first time in 2004, in the first two years the collection consisted of journal articles from IEEE computer science journals [3, 4]. In 2006/7 [5] and 2008 [6] the collection was collected from the Wikipedia. In 2009 [7] the iTrack switched to a collection consisting of book metadata collected from the bookstore Amazon and the social cataloguing application LibraryThing.

Throughout the years the design of the iTrack experiments has been quite similar:

- 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

In this way the participating institutions have gained access to a rich and comparable set of data on user background and user behavior, with a relatively small investment time and effort. The data collected has been subject for both qualitative and quantitative analysis resulting in a number of papers and conference presentations ([8], [9], [10], [11], [12], [13], [14], [15]).

In 2009, it was felt that although the "common effort" quality of the previous years was valuable and still held potential as an efficient way of collecting user behavior data, the Wikipedia collection had exhausted its potential as a source for studies of user interaction with XML-coded documents. It was therefore decided to base the experiments on a new data collection with richer structure and more semantic markup than has previously been available. The collection was based on a crawl of 2.7 million records from the book database of the online bookseller Amazon.com, consolidated with corresponding bibliographic records from the cooperative book cataloguing tool LibraryThing. A sub-set of the same collection was used in this year's experiments, with a change to a new IR system, of which two alternative versions were made (a more specific description of the system and collection is given below). The records present book descriptions on a number of levels: formalized author, title and publisher data; subject descriptions and user tags; book cover images; full text reviews and content descriptions. New this year is that more emphasis is given to the distinction between publisher data and user-generated data. The two systems differ in that it is not possible to query the reviews nor the book abstracts in one of the two versions. The database intended to enable investigation of research questions concerning, for instance

- What is the basis for judgments on relevance in a richly structured and diverse material? What fields / how much descriptive text do users make use of / chose to see to be able to judge relevance?
- How do users understand and make use of structure (e.g. representing different levels of description, from highly formalized bibliographic data to free text with varying degrees of authority) in their search development?
- How do users construct and change their queries during search (sources of terms, use and understanding of tags, query development strategies ..)?
- How do users search strategies differ at different stages of their work task processes?

2 Tasks

For the 2010 iTrack the experiment was designed with two categories of tasks constructed by the track organizers, from each of which the searchers were instructed to select one of three alternative search topics. In addition the searchers were invited to perform one semi-self-generated task, which would function as a control task. The two task categories were designed to be presented in contexts that reflect two different stages of a work task process [16]. The theory underlying our choice of tasks is that searchers at an early stage in the process will be in a more explorative and problem-oriented mode, whereas at a later stage they will be focused towards more specific data collection.

The first set of tasks is designed to let searchers use a broad selection of metadata, in particular combining topical searches with the use of review data. The tasks are thus designed to inspire users to create "polyrepresentative" [17] search strategies, i.e. to use explorative search strategies, which will give us data on query development, metadata type preference and navigation patterns.

At the second stage we try to simulate searchers that are in a rather mechanistic data gathering mode. The tasks also represent tasks designed to focus on non-topical characteristics of the books. Information will typically be found in publisher's texts and possible tags.

The self-selected task is intended to function as a "control" task, the performance of which we can compare the two others to.

The task groups are introduced in the following way:

Task group 1: The explorative tasks

You are at an early stage of working on an assignment, and have decided to start exploring the literature of your topic. Your initial idea has led to one of the following three research needs:

1. Find trustworthy books discussing the conspiracy theories which developed after the 9/11 terrorist attacks in New York.
2. Find controversial books discussing the climate change and whether it is man-made or not.
3. Find highly acclaimed novels that treat issues related to racial discrimination.

Task group 2: The data gathering tasks

You are in a data gathering stage of an assignment and need to collect a series of books for further analysis. This has led to one of the following three research needs:

4. Find novels that won the Nobel prize during the 1990's.
5. Find bestseller crime novels by female authors.

6. Find biographies on athletes active in the 1990's.

The semi self-selected task

7. Try to find books about a specific topic or of a certain type, but do not look for a specific title you already know.

3 Participating groups

Only 2 research groups were able to submit experiment data by the deadline for this report: Oslo University College and University of Duisburg-Essen. Data from a total of 126 searches performed by 42 test subjects were collected.

4 Research design

4.1 Search system

The experiments were conducted on a Java-based retrieval system built within the ezDL framework (http://www.is.inf.uni-due.de/projects/ezdl/, http://ezdl.de), which resides on a server at and is maintained by the University of Duisburg-Essen. The collection was indexed with Apache Solr 1.4, which is based on Apache Lucene. Lucene applies a variation of the vector space retrieval model. The basis of the search system is similar to the interfaces used for previous iTracks, but the interface has been modified extensively to accommodate the new data set, and a set of new functionalities have been developed. Two versions (A and B) were developed.

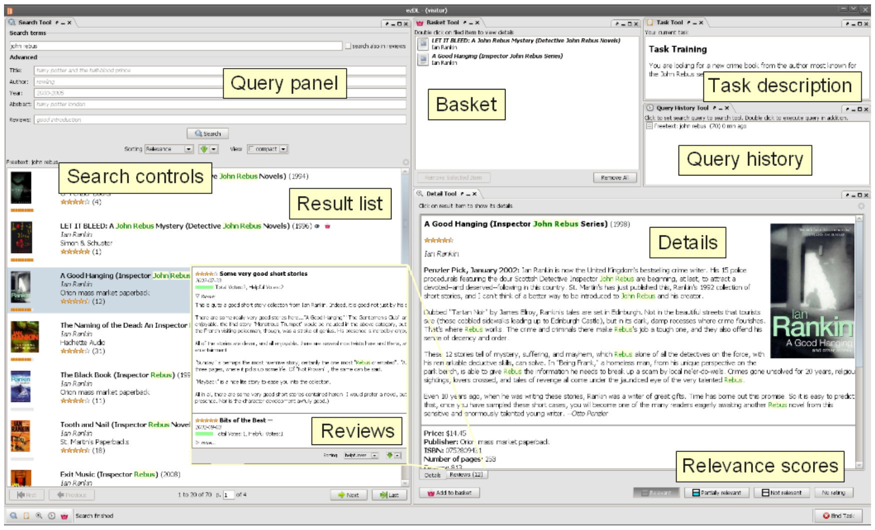


Figure 1: The search system interface

Figure 1 shows the interface of the system (A version). The main features available to the user are:

- The query interface provides a Google-like query field as well as additional query fields for title, author, year, abstract and reviews. When a search term is entered, the searcher can choose if he wants to search also in the reviews.
- The system can order the search results according to “relevance” (which books the system considers to be most relevant to your search terms), “year” (publication year of the book), or “average rating” (in the cases where quality ratings from readers were available).
- The system will show results twenty titles at a time, with features to assist in moving further forwards or backwards in the result list.
- A double click on an item in the result list will show the book details in the “Details” window.
- If the book has been reviewed, the reviews can be seen by clicking the “Reviews” tab at the bottom of this window. Each review shows the title, the rating, the date and the helpfulness rating. A simple click on a review extends the review by the full review text
- The users are instructed to determine the relevance of any examined book, as “Relevant”, “Partially relevant” or “Not relevant”, by clicking markers at the bottom of the screen. Any book decided to constitute part of the answer to the search task can be moved to a result basket by clicking the “Add to basket” button next to the relevance buttons.
- A “Query history” button in the right of the screen displays the query terms used so far in the current search session. A single click sets a query to the search tool. A double-click also executes this query

- A line of yellow dots above an item in the result list is used to indicate the system's estimate of how closely related to the query the item is considered to be.
- Query terms are highlighted in the result list and the detail tool

The B version of the search system did not allow the user to search in reviews or abstracts, that is no query fields for abstract and reviews were available to the user.

4.2 Document corpus

The collection contains metadata of 2 780 300 English-language books. The data has been crawled from the online bookstore Amazon and the social cataloging web site LibraryThing in February and March 2009 by the University of Duisburg-Essen. The MySQL database containing the crawled data has a size of about 190 GB. Cover images are available for over one million books (100 GB of the database). Several millions of customer reviews were crawled. For this year's run of the track we cleaned up the data by removing all records that do not have an image of the book cover. This was thought to be a good heuristic for removing records that only have very sparse data. After the clean-up, metadata from approximately 1.5 million books remained in the database.

The records present book descriptions on a number of levels: formalized author, title and other bibliographic data; controlled subject descriptions and user-provided content-descriptive tags; book cover images; full text reviews and publisher-supplied content descriptions. The following listing shows what items were crawled from either Amazon or LibraryThing:

Amazon
ISBN, title, binding, label, list price, number of pages, publisher, dimensions, reading level, release date, publication date, edition, Dewey classification, title page images, creators, similar products, height, width, length, weight, reviews (rating, author id, total votes, helpful votes, date, summary, content) editorial reviews (source, content).

LibraryThing
Tags (including occurrence frequency), blurbs, dedications, epigraphs, first words, last words, quotations, series, awards, browse nodes, characters, places, subjects.

4.3 Online questionnaires

During the course of the experiment, the system presents the searchers with online questionnaires to support the analysis of the log data. The searchers were given a pre-experiment questionnaire, with demographic questions such as searchers' age, education and experience in information searching in general and in searching and buying books online. Each search task was preceded with a pre-task questionnaire, which concerned searchers' perceptions of the difficulty of the search task, their familiarity with the topic etc. After each task, the searcher was asked to fill out a post-task questionnaire. The intention of the post-task questionnaire is to learn about the searchers' use of and their opinion on various features of the search system, in relation to the just completed task. Each experiment sessions were closed with a post-experiment questionnaire, which elicited the searchers' general opinion of the search system.

4.4 Relevance assessments

The searchers were instructed to indicate the relevance of the items in the result list, using a three-part relevance scale of "relevant", "partly relevant" and "not relevant".

4.5 "Shopping" basket

To simulate the purchase of relevant books the system provided a shopping basket feature in which searchers were asked to add books they would have purchased for solving the task. Books can be added and removed from the basket.

4.6 Logging

All search sessions were logged and saved to a database. The logs register and time stamp the events in the session and the actions performed by the searcher, as well as the responses from the system.

5 Experimental Procedure

The experimental procedure for each searcher is outlined below.

1. When recruiting searchers for the experiment, the experimenter gives the searchers the instructions for the self-selected task.
2. Experimenter briefs the searcher, and explains format of study. The searcher reads and signs the Consent Form.
3. The experimenter logs the searchers into the system. This presents the searcher with the task assignments and the questionnaire. The experimenter hands out and explains the User guidelines document. It is important to take

good time to demonstrate and explain how the system works. A tutorial of the system with a training task is provided.

4. The experimenter answers questions from user.
5. The searcher selects his/her tasks from each of the two categories. In addition the self-selected task is input into the appropriate form. Tasks are rotated by the system, thus any of the three tasks may be the first to be solved by the searcher.
6. The searcher answers the Pre-experiment questionnaire provided by the system.
7. The searcher answers the Pre-task questionnaire provided by the system.
8. The task is started by clicking the link to the IR system. Each task has a duration of 15 minutes, at which point the system will tell the user time has run out. The IR system is closed by clicking the "End task" button.
9. The searcher answers the Post-task questionnaire provided by the system.
10. Steps 6-9 repeated for the two other tasks.
11. The searcher answers the Post-experiment questionnaire provided by the system.
12. At the end of the evaluation session the user presses the "Finish" button in the evaluation/questionnaire system to store his data into the database

6 Data analysis

To be filled out.

References

[1] Malik, S., Trotman, A., Lalmas, M. & Fuhr, N. (2007): Overview of INEX 2006. In: Fuhr, N., Lalmas, M. and Trotman, A. eds. *Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, December 2006*. Berlin: Springer, p. 1-11.

[2] Ruthven, I. (2008): Interactive Information Retrieval. In: Annual Review of Information Science and Technology, 42, p. 43-91.

[3] Tombros, A., Larsen, B. and Malik, S. (2005): The Interactive Track at INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S. and Szlávik, Z. eds. *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004*. Berlin: Springer, p. 410-423

[4] Larsen, B., Malik, S. and Tombros, A. (2006): The interactive track at INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S. and Kazai, G. eds. *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005*. Berlin: Springer, p. 398-410.

[5] Larsen, B., Malik, S. & Tombros, A. (2007): The Interactive track at INEX 2006. In: Fuhr, N., Lalmas, M. and Trotman, A. eds. *Comparative Evaluation of XML*

Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, December 2006. Berlin: Springer, p. 387-399.

[6] Pharo, N., Nordlie, R. & Fachry, K. N. (2009): Overview of the INEX 2008 Interactive Track. In: Geva, S., Kamps, J. and Trotman, A. eds. *Advances in Focused Retrieval, 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008, Dagstuhl Castle, Germany, December 2008*. Berlin: Springer, p. 300-313.

[7] Pharo, N., Nordlie, R., Fuhr, N., Beckers, T. & Fachry, K. N. (2010): Overview of the INEX 2009 Interactive Track. In: Geva, S., Kamps, J. and Trotman, A. eds. *Focused Retrieval and Evaluation, 8th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2009, Brisbane, Australia, December 200*. Berlin: Springer, p. 303-311.

[8] Pharo, N. & Nordlie, R. (2005): Context Matters: An Analysis of Assessments of XML Documents. In: F. Crestani and I. Ruthven eds. *Information Context: Nature, Impact, and Role: 5th International Conference on Conceptions of Library and Information Sciences, CoLIS 2005, Glasgow, UK, June 4-8, 2005*. Berlin: Springer, p. 238-248.

[9] Hammer-Aebi, B., Christensen, K. W., Lund, H. and Larsen, B. (2006): Users, structured documents and overlap: interactive searching of elements and the influence of context on search behaviour. In: Ruthven, I. et al. eds. *Information Interaction in Context : International Symposium on Information Interaction in Context : IIIiX 2006 : Copenhagen, Denmark, 18-20 October, 2006 : Proceedings.* Copenhagen: Royal School of Library and Information Science, p. 80-94.

[10] 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.

[11] Malik, S., Klas, C.-P., Fuhr, N., Larsen, B. and Tombros, A. (2006): Designing a user interface for interactive retrieval of structured documents: lessons learned from the INEX interactive track? In: Gonzalo, J. et al. eds. *Research and Advanced Technology for Digital Libraries, 10th European Conference, ECDL 2006*. Alicante, Spain, September 17-22, 2006, Proceedings. Berlin: Springer, p. 147-156.

[12] Kim, H. & Son, H. (2006): Users Interaction with the Hierarchically Structured Presentation in XML Document Retrieval. In: Fuhr, N., Lalmas, M., Malik, S. & Kazai, G. eds. *Advances in XML Information Retrieval and Evaluation: 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005.* Berlin: Springer, p. 422-431.

[13] Kazai, G. & Trotman, A (2007): Users' perspectives on the Usefulness of Structure for XML Information Retrieval. In: Dominich, S. & Kiss, F. eds. *Proceedings of the 1st International Conference on the Theory of Information Retrieval*. Budapest: Foundation for Information Society, p. 247-260.

[14] Larsen, B., Malik, S & Tombros, A. (2008): A Comparison of Interactive and Ad-Hoc Relevance Assessments. In: Fuhr, N., Kamps, J., Lalmas, M. & Trotman, A. eds. *Focused Access to XML Documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007 Dagstuhl Castle, Germany, December 17-19, 2007.* Berlin: Springer, p. 348-358.

[15] Pharo, N. (2008): The effect of granularity and order in XML element retrieval. Information Processing and Management. 44(5), 1732-1740.

[16] Kuhlthau, C.C. (2004). *Seeking meaning: a process approach to library and information services. 2nd ed*. Libraries Unlimited, Westport, CT.

[17] Larsen, B., Ingwersen, P & Kekäläinen, J. (2006). The Polyrepresentation continuum in IR. In: I. Ruthven et al. eds. *Information interaction in context: International Symposium on Information Interaction in Context, IIiX 2006*. New York: ACM Press, p. 148-162.

[13] 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.

.

Using Eye-Tracking for the Evaluation of Interactive Information Retrieval

Thomas Beckers and Dennis Korbar

Universität Duisburg-Essen
Information Engineering
Duisburg, Germany
{tbeckers, korbar}@is.inf.uni-due.de

Abstract. In this paper we present the INEX 2010 Interactive Track experiments that have been performed with eye-tracking in addition to the other collected data (system logs and questionnaires). We present our tool *AOILog* for recording AOI data for dynamic user interfaces. Finally, we show how these eye-tracking and AOI data could be used for further analysis of the user interaction with the search system.

1 Introduction

In this year's run of the Interactive Track (iTrack) we wanted to investigate how users act with an interactive search system. The working tasks do not only rely on the classical topic aspect but also on other aspects such as book reviews or structure information. In addition to the standard experiments (see [1] for more detailed explanations of the experiment design and the data collection) in the 2010 run of the iTrack, we additionally used an eye-tracking system to record the user's gaze data while interacting with the search system.

2 System Description

The search system (see figure 1) was developed at the University of Duisburg-Essen. It is based on the digital library system *ezDL*[1]. Pharo et al. provide a more detailed explanation (see [1]).

The **search tool** offers a Google-like search field as well as advanced search fields for title, author, year, abstract and reviews (depends on the system version, see [1]). A combo box allows the user to search also in reviews with the Google-like search field. Below this query panel the user can select fields for the sorting of the results. Furthermore, the user can choose the display style of the result list. The lower half of the search tool contains the result list and the result page navigation buttons.

A double-click on a result item shows book details in the **detail tool**. Users can indicate the relevance of an examined book as either *relevant*, *partially relevant*, or *not relevant*, by clicking markers at the bottom of the tool. A second

[1] Live system: `http://www.ezdl.de/`
Developer site: `http://www.is.inf.uni-due.de/projects/ezdl/`

tab shows reviews of the selected book. Initially the title, author, rating, date and the utility rating of the review is shown. By clicking on a review, the actual review text is added to the review.

Users can mark any book as part of the answer to the search task by moving it to the **basket tool**. This can be performed either via drag-and-drop or by clicking the *add to basket* button next to the relevance buttons.

A history of performed search queries is provided by in the **query history tool**. Finally, the **task tool** shows the current working task.

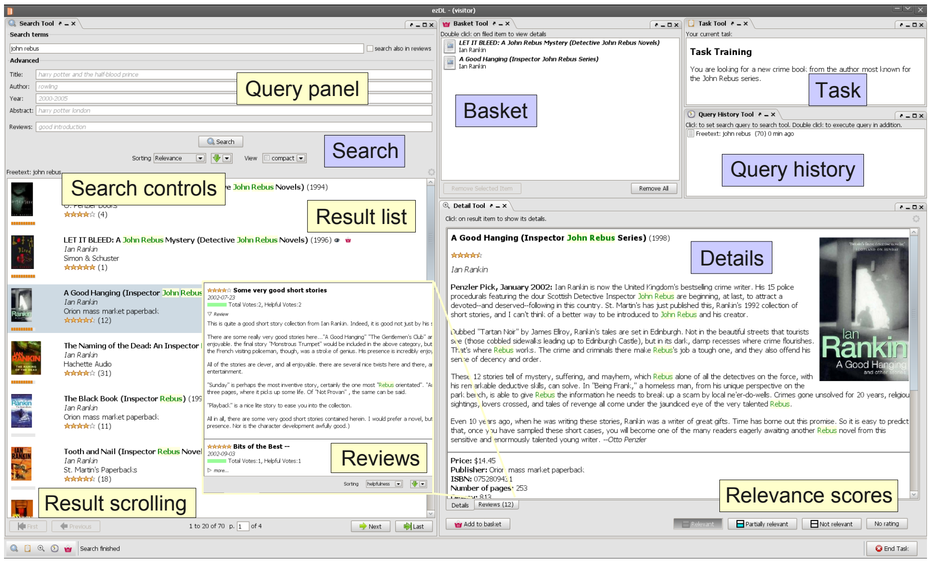


Fig. 1: The search system interface (blue boxes: tools mentioned in the description in section 2)

3 Eye-Tracking for Interactive Information Retrieval Systems

In addition to the questionnaires and system log data we also used an eye-tracking system[2] to record the user's eye gaze data. For analysing the eye-tracking data, the user interface of the system (see figure 1) was divided into so called Areas of Interests (AOIs) (see figure 2). AOIs define larger and logically connected gaze areas. These areas are used to capture not only fixations, but also more peripheral perceptions. Since some of the tools in the search system are on top of other tools and the position of user interface components changes

[2] SMI RED: `http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/red-red250-red-500.html`

dynamically, it is necessary to record the visibility information of tools to create dynamic AOIs automatically. The manual creation of AOIs would take too much time, thus is is generally not applicable in practice, especially when trying to analyze very dynamic components.

We developed a framework called *AOILog* which automatically keeps track of position, visibility and size of registered Java Swing components, thus enabling us to consider even small and very dynamic areas of interest, such as result items in a scrollable list widget. Also included in the AOILog framework is a small converter application which will convert our AOI XML format into the proprietary SMI AOI format. This will enable us to use the recorded data in the statistical analysis software *BeGaze* which is part of SMI's software suite. In order to use our framework with eye tracking devices by other manufacturers, one only has to implement an appropriate converter, the logging functionality on Java Swing based applications can be used out of the box.

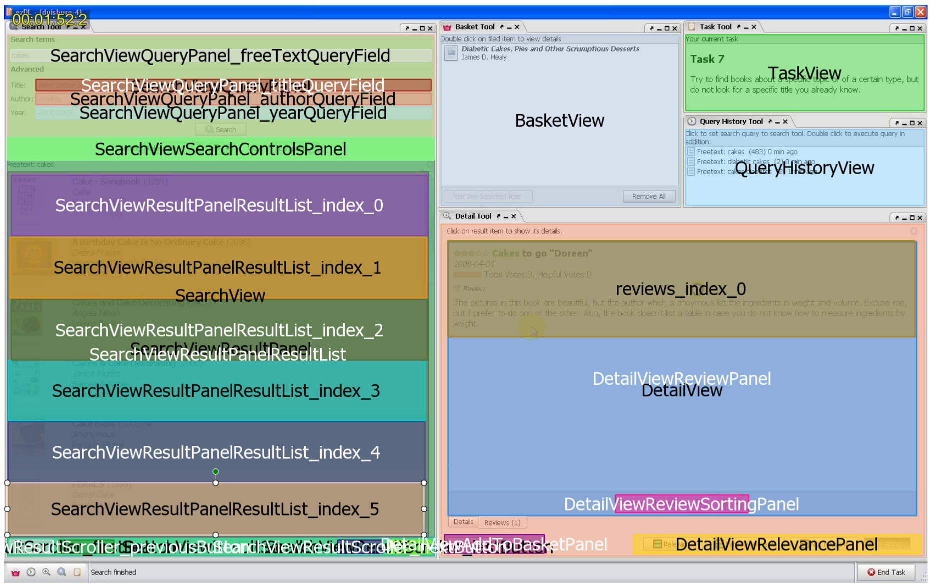

00:01:52:2
SearchViewQueryPanel_freeTextQueryField
SearchViewQueryPanel_titleQueryField
SearchViewQueryPanel_authorQueryField
SearchViewQueryPanel_yearQueryField
SearchViewSearchControlsPanel
SearchViewResultPanelResultList_index_0
SearchViewResultPanelResultList_index_1
SearchView
SearchViewResultPanelResultList_index_2
SearchViewResultPanel
SearchViewResultPanelResultList
SearchViewResultPanelResultList_index_3
SearchViewResultPanelResultList_index_4
SearchViewResultPanelResultList_index_5
BasketView
TaskView
Task 7
QueryHistoryView
reviews_index_0
DetailViewReviewPanel
DetailView
DetailViewReviewSortingPanel
DetailViewAddToBasketPanel
DetailViewRelevancePanel
Remove All
End Task
Search finished

Fig. 2: Areas of Interest (screenshot from BeGaze analysis software)

Figure 2 shows the AOIs of the search interface. We defined the following AOIs:

– tools (search, details, task, query history)
– query panel and query fields
– search controls and result scrolling
– result list
– parts of the details
– reviews and review sorting

With these dynamic AOIs it will be possible for future analyses to investigate the use of tools and book details as well as reviews in general by the users.

4 A Model for Interactive Information Retrieval

In 2010 we started a new project called HIIR (Highly Interactive Information Retrieval)[3] aiming to create an interactive information retrieval system based on efficient retrieval algorithms combined with a theoretic model for interactive IR, the IIR-PRP[4][2]. The IIR-PRP tries to offer decision lists and information based on a cost/benefit ratio. The basic idea is that users move from situation to situation. In every situation a list of choices is presented to the user, s/he then decides about each of these choices sequentially. Upon the first positive decision the user will move to a new situation.

In order to use the IIR-PRP as a model to implement interactive IR systems, its assumptions have to be confirmed. Furthermore we will have to measure some of its constants in order to be able to apply the model's ranking method. The AOI logging framework described above will help us to accomplish these goals. Its implementation in the current INEX 2010 iTrack search system will provide us with a first indication about the validity of the IIR-PRP's assumption concerning user behaviour while scanning a list of choices. In addition we might be able determine if there is a connection between a result item's textual length and the effort to evaluate that item. We will also try to analyze the effort to evaluate other objects provided by the search system (e.g. review items). This might enable us to create a first measurement to calculate the expected effort for a given choice.

5 Conclusion and Outlook

We presented how eye-tracking data can be used for evaluation of interactive information retrieval systems.

We plan to analyze the collected data (questionnaires, system logs and eye-tracking data) to find out how users interact with interactive search systems. Our framework *AOILog* will be extended and included into *ezDL*. Furthermore it is planned to develop tools to make the analysis of the eye-tracking data and the linkage to the system logs easier.

References

1. Pharo, N., Beckers, T., Nordlie, R., Fuhr, N.: The INEX 2010 Interactive Track: an overview. In: Pre-Proceedings of the 9th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2010). (2010)
2. Fuhr, N.: A probability ranking principle for interactive information retrieval. Information Retrieval **11**(3) (2008) 251–265

[3] `http://www.is.inf.uni-due.de/projects/hiir/index.html.en`
[4] Interactive Information Retrieval Probability Ranking Principle

Overview of the INEX 2010 Link the Wiki Track

Andrew Trotman and David Alexander

Department of Computer Science
University of Otago
Dunedin
New Zealand

Abstract. Short status report on the INEX 2010 Link the Wiki track

1 Introduction

In 2010 the link the wiki track changed from the Wikipedia collection to the Te Ara collection. The use of the new collection saw a decline in the number of participating groups with only QUT and Otago submitting runs.

A total of 29 runs were submitted, 5 from QUT and 24 from Otago. The track is currently in the assessment phase. At time of writing 7 pools of 10 topics per pool (70 documents in total) have been assessed.

University of Otago at INEX 2010

Xiang-Fei Jia, David Alexander, Vaughn Wood and Andrew Trotman

Computer Science, University of Otago, Dunedin, New Zealand

Abstract. In this paper, we describe University of Otago's participation in Ad Hoc, Link-the-Wiki Tracks, Efficiency and Data Centric Tracks of INEX 2010. In the Link-the-Wiki Track, we show that the simpler relevance summation method works better for producing Best Entry Points (BEP). In the Ad Hoc Track, we discusses the effect of various stemming algorithms. In the Efficiency Track, we compare three query pruning algorithms and discusses other efficiency related issues. Finally in the Data Centric Track, we compare the BM25 and Divergence ranking functions.

1 Introduction

In INEX 2010, University of Otago participated in the Ah Hoc, Link-the-Wiki Tracks, the Efficiency and Data Centric Tracks. In the Link-the-Wiki Track, we talk about how our linking algorithm works using the Te Ara collection and the newly developed assessment tool. In the Ad Hoc Track, we show the performance of our stemming algorithm using Genetic Algorithms and how it performs against other stemming algorithms. In the Efficiency Track, we discusses the performance of our three pruning algorithms; The first is the original *topk* (originally described in INEX 2009), an improved version of the *topk* and the *heapk*. Finally in the Data Centric Track, we compare the BM25 and Divergence ranking functions.

In Section 2, related work is discussed. Section 3 explains how our search engine works. Section 4, 5, 6 and 7 talk about how we performed in the corresponding Tracks. The last section provides the conclusion and future work.

2 Related Work

2.1 The Link-the-Wiki Track

The aim of the INEX Link-the-Wiki track is to develop and evaluate link recommendation algorithms for large hypertext corpora.

Before 2009, Wikipedia was the only corpus used in the Link-the-Wiki track; the task was to link related Wikipedia documents to each other, with or without providing specific anchor locations in the source documents. In 2009, the *Te Ara Encyclopedia of New Zealand* was used alongside Wikipedia, and tasks included producing links within each of the two corpora, and linking articles in one corpus to articles in the other.

Work has been done on the topic of hypertext link recommendation by a number of people both within the INEX Link-the-Wiki track and outside of it. It is difficult to compare INEX-assessed algorithms with non-INEX-assessed algorithms because the assessment methodology plays a large part in the results, so this section will focus on algorithms from within INEX.

For Wikipedia, the two most successful link-recommendation algorithms are due to Kelly Itakura [1] and Shlomo Geva [2].

Itakura's algorithm chooses anchors in a new document by calculating the probability (γ) that each phrase, if found in the already-linked part of the corpus, would be an anchor. If γ exceeds a certain threshold (which may be based on the length of the document), the phrase is used as an anchor. The target for the link is chosen to be the most common target for that anchor among existing links. The formula for γ for a given phrase P is:

$$\gamma = \frac{\text{number of occurrences of } P \text{in the corpus as a link}}{\text{number of occurrences of } P \text{in the corpus altogether}}$$

Geva's algorithm simply searches for occurrences of document titles in the text of the orphan document. If such an occurrence is found, it is used as an anchor. The target of the link is the document whose title was found.

2.2 The Ad Hoc Track

In the Ad Hoc Track, we compare the performance of our stemming algorithm using Genetic Algorithms with other stemming algorithms.

The S Stripper consists of three rules. These rules are given in Table 1. It uses only the first matching rule. It has improved MAP on previous INEX Ad Hoc collections, from 2006-2009. This serves as a baseline for stemmer performance, and is an example of a weak stemmer (It does not conflate many terms).

ies	→	y
es	→	
s	→	

Table 1: S Stripper rules. The first suffix matched on the left is replaced by the suffix on the right.

The Porter stemmer[3] has improved some runs for our search engine on previous INEX collections. It serves as an example of a strong stemmer. We use it here as a baseline for comparing stemmer performance.

People have found ways to learn to expand queries using thesauruses generation or statistical methods. Jones[4] used clustering methods for query expansion. We have been unable to find any mention of symbolic learning used for stemming.

A similar method for improving stemming by using term similarity information from the corpus was used by Xu and Croft[5]. Their work uses the Expected

Mutual Information Measure. Instead we have used Pointwise Mutual Information and the Jaccard Index. These were chosen as the best out of a larger group of measures.

2.3 The Efficiency Track

The following discussion of the related work is taken from our published paper in ADCS 2010 [6].

Disk I/O involves reading query terms from a dictionary (a vocabulary of all terms in the collection) and the corresponding postings lists for the terms. The dictionary has a small size and can be loaded into memory at start-up. However, due to their large size, postings are usually compressed and stored on disk. A number of compression algorithms have been developed and compared [7,8]. Another way of reducing disk I/O is caching, either at application level or system level [9,10]. Since the advent of 64-bit machines with vast amounts of memory, it has become feasible to load both the dictionary and the compressed postings into main memory, thus eliminating all disk I/O. Reading both dictionary and postings lists into memory is the approach taken in our search engine.

The processing (decompression and similarity ranking) of postings and subsequent sorting of accumulators can be computationally expensive, especially when queries contain frequent terms. Processing of these frequent terms not only takes time, but also has little impact on the final ranking results. Postings pruning at query time is a method to eliminate unnecessary processing of postings and thus reduce the number of non-zero accumulators to be sorted. A number of pruning methods have been developed and proved to be efficient and effective [11,12,13,14,15,16]. In our previous work [16], the *topk* pruning algorithm partially sorts the static array of accumulators using an optimised version of quick sort [17] and statically prunes postings. In this paper, we present an improved *topk* pruning algorithm and an new pruning algorithm based on heap data structure.

Traditionally, term postings are stored in pairs of <document number, term frequency> pairs. However, postings should be impact ordered so that most important postings can be processed first and the less important ones can be pruned using pruning methods [18,14,15]. One approach is to store postings in order of term frequency and documents with the same term frequency are grouped together [18,14]. Each group stores the term frequency at the beginning of the group followed by the compressed differences of the document numbers. The format of a postings list for a term is a list of the groups in descending order of term frequencies. Another approach is to pre-compute similarity values and use these pre-computed impact values to group documents instead of term frequencies [15]. Pre-computed impact values are positive real numbers. In order to better compress these numbers, they are quantised into whole numbers [19,15]. Three forms of quantisation method have been proposed (*Left.Geom*, *Uniform.Geom*, *Right.Geom*) and each of the methods can better preserve certain range of the original numbers [15]. In our search engine, we use pre-computed

BM25 impact values to group documents and the differences of document numbers in each group are compressed using Variable Byte Coding by default. We choose to use the *Uniform.Geom* quantisation method for transformation of the impact values, because the *Uniform.Geom* quantisation method preserves the original distribution of the numbers, thus no decoding is required at query time. Each impact value is quantised into an 8-bit whole number.

Since only partial postings are processed in query pruning, there is no need to decompress the whole postings lists. Skipping [12] and blocking [20] allow pseudo-random access into encoded postings lists and only decompress the needed parts. Further research work [21,22] represent postings in fixed number of bits, thus allowing full random access. Our search engine partially decompress postings list based on the worst case of the static pruning. Since we know the parameter value of the static pruning and the biggest size of an uncompressed impact value (1 byte), we can add these together to find the cut point for decompression. We can simply hold decompression after that number of postings have been decompressed.

3 System Overview

3.1 Indexer

Memory management is a challenge for fast indexing. Efficient management of memory can substantially reduce indexing time. Our search engine has a memory management layer above the operating system. The layer pre-allocates large chunks of memory. When the search engine requires memory, the requests are served from the pre-allocated pool, instead of calling system memory allocation functions. The sacrifice is that some portion of pre-allocated memory might be wasted. The memory layer is used both in indexing and in query evaluation. As shown previously in [16], only a very small portion of memory is actually wasted.

The indexer uses hashing with a collision binary tree for maintaining terms. We tried several hashing functions including Hsieh's super fast hashing function. By default, the indexer uses a very simple hashing function, which only hashes the first four characters of a term and its length by referencing a pre-defined look-up table. A simple hashing function has less computational cost, but causes more collisions. Collisions are handled by a simple unbalanced binary tree. We will examine the advantages of various hashing and chaining algorithms in future work.

Postings lists can vary substantially in length. The indexer uses various sizes of memory blocks chained together. The initial block size is 8 bytes and the re-size factor is 1.5 for the subsequent blocks.

The indexer supports either storing term frequencies or pre-computed impact values. A modified BM25 is used for pre-computing the impact values. This variant does not result in negative IDF values and is defined thus:

$$RSV_d = \sum_{t \in q} log\left(\frac{N}{df_t}\right) \cdot \frac{(k_1+1)\, tf_{td}}{k_1\left((1-b) + b \times \left(\frac{L_d}{L_{avg}}\right)\right) + tf_{td}}$$

here, N is the total number of documents, and df_t and tf_{td} are the number of documents containing the term t and the frequency of the term in document d, and L_d and L_{avg} are the length of document d and the average length of all documents. The empirical parameters k_1 and b have been set to 0.9 and 0.4 respectively by training on the previous INEX Wikipedia collection.

In order to reduce the size of the inverted file, we always use 1 byte to store term frequencies and pre-computed impact values. This limits to a maximum value of 255. Term frequencies which have values larger than 255 are simply truncated. Truncating term frequencies could have an impact on long documents. But we assume long documents are rare in a collection and terms with high frequencies in a document are more likely to be common words. Pre-computed impact values are transformed using the *Uniform.Geom* quantisation method.

As shown in Figure 1, the index file has five levels of structure. In the top level, original documents in compressed format can be stored. Storing original documents is optional, but is required for focused retrieval.

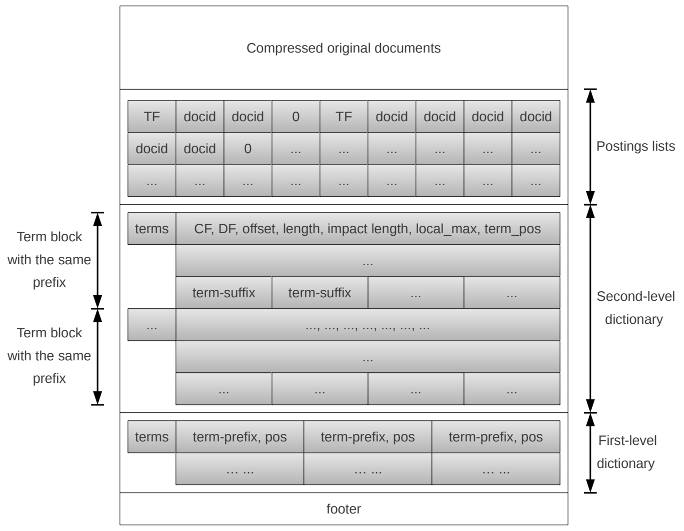

Compressed original documents
TF
docid
0
...
Postings lists
terms
CF, DF, offset, length, impact length, local_max, term_pos
term-suffix
..., ..., ..., ..., ..., ..., ...
Term block with the same prefix
Term block with the same prefix
Second-level dictionary
term-prefix, pos
... ...
First-level dictionary
footer

Fig. 1: The index structures.

Instead of using the pair of <document number, term frequency> for postings, we group documents with the same term frequency (or the impact value) together and store the term frequency (or the impact value) at the beginning of each group. By grouping and impacting order documents according to term frequencies (or impact values), during query evaluation we can easily process documents with potential high impacts first and prune the less important documents at the end of the postings list. The difference of document ids in each group are then stored in increasing order and each group ends with a zero. Postings are compressed with Variable Byte coding.

The dictionary of terms is split into two parts. Terms with the same prefix are grouped together in a term block. The common prefix (only the first four characters) is stored in the first level of the dictionary and the remaining are stored in the term block in the second level. The number of terms in the block is stored at the beginning of the block. The term block also stores the statistics for the terms, including collection frequency, document frequency, offset to locate the postings list, the length of the postings list stored on disk, the uncompressed length of the postings list, and the position to locate the term suffix which is stored at the end of the term block.

At the very end of the index file, the small footer stores the location of the first level dictionary and other values for the management of the index.

3.2 Query Evaluation

At start-up, only the the first-level dictionary is loaded into memory by default. To process a query term, two disk reads have to be issued; The first reads the second-level dictionary. Then the offset in that structure is used to locate postings. The search engine also supports a command line option which allows loading the whole index into memory, thus totally eliminating I/O at query time.

An array is used to store the accumulators. We used fixed point arithmetic on the accumulators because it is faster than the floating point.

For last year INEX, we developed the *topk* algorithm for fast sorting of the accumulators. It uses a special version of quick sort [17] which partially sorts the accumulators. A command line option (lower-k) is used to specify how many top documents to return.

Instead of explicit sorting of all the accumulators, we have developed an improved version of *topk*. During query evaluation, it keeps track of the current top documents and the minimum partial similarity score among the top documents. The improved *topk* uses an array of pointers to keep track of top documents. Two operations are required to maintain the top documents, i.e. *update* and *insert*. If a document is in the top documents and gets updated to a new score, the improved *topk* simply does nothing. If a document is not in the top k and gets updated to a new score which is larger than the minimum score, the document needs to be inserted into the *topk*. The insert operation is accomplished by two linear scans of the array of pointers; (1) the first scan locates the document which has the minimum score and swap the minimum document with the newly updated document, (2) the second finds the current minimum similarity score.

Based on the *topk* algorithm, we have further developed a new algorithm called *heapk*. It uses a minimum heap to keep track of the top documents. Instead of using the minimum similarity score, *heapk* uses bit strings to define if a document is among the top k. The heap structure is only built once which is when the number of top slots are fully filled. If a document is in the heap and gets updated to a new score, *heapk* first linearly scans the array to locate the document in the heap and then partially updates the structure. If a document is not in the heap and the newly updated score is larger than the minimum score (the first pointer) in the heap, *heapk* partially inserts the document into the heap.

The upper-K command line option is used for static pruning of postings. It specifies a value, which is the number of postings to be processed. Since only part of the postings lists is processed, there is no need to decompress the whole list. Our search engine partially decompress postings lists based on the worst cast. Since we know the parameter value of upper-K and the biggest size of an uncompressed impact value (1 byte), we can add these together to find the cut point for decompression.

4 The Link-The-Wiki Track

In this year's Link-the-Wiki track, the only corpus used was the *Te Ara Encyclopedia of New Zealand*. Wikipedia was abandoned as a corpus because it had become too easy for algorithms to score highly according to the metrics used by INEX. This is believed to be because of characteristics of Wikipedia that Te Ara does not possess. Te Ara is therefore of interest because it presents challenges that Wikipedia does not.

It is also of interest because its maintainers (New Zealand's Ministry of Culture and Heritage) have asked for links to be incorporated into the official, public version of their encyclopedia. This is an opportunity for these linking algorithms to be tested in a real-world application.

Our participation in the Link-the-Wiki track is detailed in the rest of this section. First, the differences between Wikipedia and Te Ara are outlined, as well as the possible ways to develop linking algorithms for Te Ara. Then, our own linking algorithm is explained, and its assessment results given. Finally, our contribution to the Link-the-Wiki assessment process is explained.

4.1 Differences between Wikipedia and Te Ara

The most important difference between Wikipedia and Te Ara is that Te Ara has no existing links. The Link-the-Wiki Track has always been to take a single "orphan" (a document whose incoming and outgoing links have been removed) and produce appropriate links to and from it, using the remainder of the corpus (including any links that do not involve the orphan) as input if desired. This meant that algorithms could statistically analyse the anchors and targets of the existing links in the corpus, using that information to decide what kind of links would be appropriate for the orphan document. Itakura's algorithm (described in Section 2) is an example of one that does so, and it has been consistently successful on Wikipedia.

In Te Ara this is not possible. The problem is not merely the lack of links, but that the encyclopedia was not written with links in mind. In any body of writing there are a number of different ways to refer to a given topic, but in a hypertext corpus such as Wikipedia, writers tend to use existing article titles as "canonical names" to refer to the topics of those articles. The absence of this in Te Ara renders an approach such as Geva's algorithm less effective.

Wikipedia and Te Ara are also organised in very different ways. Te Ara is primarily a record of New Zealand history, and the discussion of any given topic may be spread among several articles, each of which may discuss other topics as well. This is especially true of topics that are relevant to both the indigenous and colonial inhabitants of New Zealand; and also topics that have been relevant over a long period of time. In Wikipedia, even such wide-ranging topics are typically centred around a single article.

4.2 Adapting to the differences in Te Ara

Without the possibility of using previous years' best-performing algorithms directly on Te Ara, we were left with two options: we could either find a way to "map" Wikipedia documents to their closest Te Ara counterparts, and then translate Wikipedia links into Te Ara links; or we could devise a new linking algorithm that did not rely on existing links at all.

We chose the latter option because, as discussed above, Te Ara is organised very differently from Wikipedia, and finding a suitable mapping would have been difficult. The algorithm we used is described below.

4.3 Algorithm

The main premise behind our linking algorithm is that Te Ara documents are less "to-the-point" than Wikipedia documents (that is, a single Te Ara article tends to touch on numerous related topics in order to "tell a story" of some sort), and therefore it is important to take into account the immediate context of a candidate anchor or entry-point, as well as the more general content of the two documents being linked.

Three sets of files were created and indexed using our search engine (described in Section 3). In the first, each document was contained within a separate file. In the second, each section of each document was contained within a separate file. The third was the same, but only included the section headings rather than the body text of each section. In this way, we were able to vary the level of target-document context that was taken into account when searching for possible entry-points for a given link.

Within each source document, candidate anchors were generated. Every maximal sequence of consecutive words containing no stopwords or punctuation marks was considered as a candidate anchor. The purpose of this was to avoid using large portions of sentences as anchors merely because all the words appear in the target document.

For each candidate anchor, various levels of context around the anchor (document, paragraph, sentence, and clause) were extracted from the source document. Each anchor context, as well as the anchor text itself, was used to query for possible targets against whichever one of the three target file-sets provided the level of context closest in size to the source context. If a particular document (or section) appeared in the query results for the anchor text itself, and for at least one of the chosen contexts, it was used as a target for that anchor. The target was given a relevance score, which was a weighted average of the relevance scores given by BM25 for each of the different contexts' queries, based on our estimate of their importance.

24 runs were produced by varying the following 4 parameters:

– *Full-document anchor context* Whether or not the entire source document of an anchor was used as one of its contexts. If not, the largest level of context was the paragraph containing the anchor.
– *Relevance summation method* How the total relevance score for a link was added up. In one method, the relevance scores for a target, queried from all levels of context and from the anchor itself, were simply averaged using the predetermined weights. In the other method, the values averaged were the squared differences between the relevance scores for each context and from the anchor. The rationale for the second method was that if a target was much more relevant to the anchor context than the anchor, then a nearby anchor would probably be better than the current one.
– *Relevance score contribution* Whether all of the weights for the anchor contexts were non-zero, or just the weight for the largest context. When a context has a weight of zero, it still contributes to the choice of targets for an anchor, but not to their scores.
– *Target contexts* Which target contexts the anchor texts themselves were directly queried against (headings, sections or both).

4.4 Results

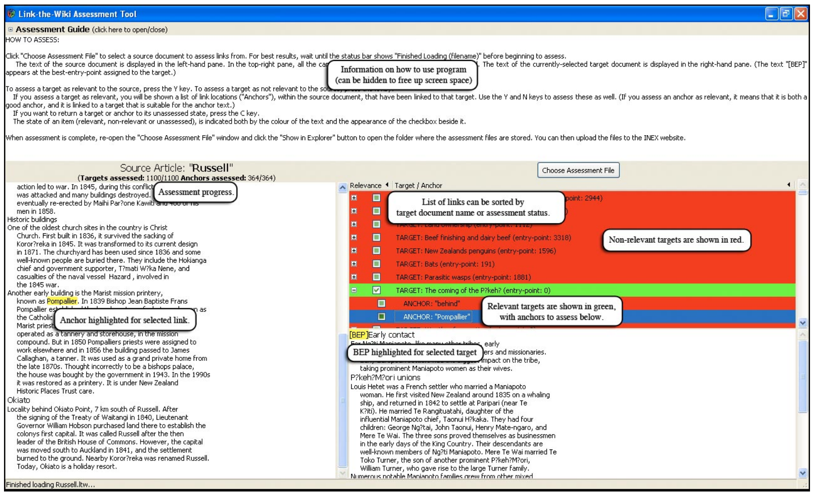
The results are incomplete because not all assessments have been received from the assessors.

4.5 Assessment Tool

Apart from submitting runs to Link-the-Wiki, we also took over the task of maintaining the assessment tool.

Improvements have been made to the assessment tool every year. However, it is crucial to the quality of our results that the manual assessment process is made as easy as possible — it is difficult for assessors to produce reliable results if they cannot understand what they are being asked, if they do not have readily available all the information that they need to make an assessment, if they need to perform unnecessarily repetitive tasks to make assessments, or if the tool responds too slowly. Therefore, we decided to make further improvements.

We rewrote the assessment tool from scratch in C++ using the cross-platform GUI library GTK+, with SQLite databases for storing assessment information.


Link-the-Wiki Assessment Tool
Information on how to use program (can be hidden to free up screen space)
Assessment progress.
List of links can be sorted by target document name or assessment status.
Non-relevant targets are shown in red.
Relevant targets are shown in green, with anchors to assess below.
Anchor highlighted for selected link.
BEP highlighted for selected target
Choose Assessment File
Relevance
Target / Anchor

Assessment Guide (click here to open/close)

HOW TO ASSESS:

Click "Choose Assessment File" to select a source document to assess links from. For best results, wait until the status bar shows "Finished Loading (filename)" before beginning to assess.
The text of the source document is displayed in the left-hand pane. In the top-right pane, all the ca . The text of the currently-selected target document is displayed in the right-hand pane. (The text "[BEP]" appears at the best-entry-point assigned to the target.)

To assess a target as relevant to the source, press the Y key. To assess a target as not relevant to the so
If you assess a target as relevant, you will be shown a list of link locations ("Anchors"), within the source document, that have been linked to that target. Use the Y and N keys to assess these as well. (If you assess an anchor as relevant, it means that it is both a good anchor, and it is linked to a target that is suitable for the anchor text.)
If you want to return a target or anchor to its unassessed state, press the C key.
The state of an item (relevant, non-relevant or unassessed), is indicated both by the colour of the text and the appearance of the checkbox beside it.

When assessment is complete, re-open the "Choose Assessment File" window and click the "Show in Explorer" button to open the folder where the assessment files are stored. You can then upload the files to the INEX website.

Source Article: "Russell"
(**Targets assessed:** 1100/1100 **Anchors assessed:** 364/364)

action led to war. In 1845, during this conflict
was attacked and many buildings destroyed.
eventually re-erected by Maihi Par?one Kawiti
men in 1858.
Historic buildings
One of the oldest church sites in the country is Christ
Church. First built in 1836, it survived the sacking of
Koror?reka in 1845. It was transformed to its current design
in 1871. The churchyard has been used since 1836 and some
well-known people are buried there. They include the Hokianga
chief and government supporter, T?mati W?ka Nene, and
casualties of the naval vessel Hazard , involved in
the 1845 war.
Another early building is the Marist mission printery,
known as Pompallier. In 1839 Bishop Jean Baptiste Frans
Pompallier es n as
the Catholic
Marist priest
operated as a tannery and storehouse, in the mission
compound. But in 1850 Pompalliers priests were assigned to
work elsewhere and in 1856 the building passed to James
Callaghan, a tanner. It was used as a grand private home from
the late 1870s. Thought incorrectly to be a bishops palace,
the house was bought by the government in 1943. In the 1990s
it was restored as a printery. It is under New Zealand
Historic Places Trust care.
Okiato
Locality behind Okiato Point, 7 km south of Russell. After
the signing of the Treaty of Waitangi in 1840, Lieutenant
Governor William Hobson purchased land there to establish the
colonys first capital. It was called Russell after the then
leader of the British House of Commons. However, the capital
was moved south to Auckland in 1841, and the settlement
burned to the ground. Nearby Koror?reka was renamed Russell.
Today, Okiato is a holiday resort.

point: 2944)
TARGET: Beef finishing and dairy beef (entry-point: 3318)
TARGET: New Zealands penguins (entry-point: 1596)
TARGET: Bats (entry-point: 191)
TARGET: Parasitic wasps (entry-point: 1881)
TARGET: The coming of the P?keh? (entry-point: 0)
ANCHOR: "behind"
ANCHOR: "Pompallier"

[BEP]Early contact
early
ers and missionaries.
mpact on the tribe,
taking prominent Maniapoto women as their wives.
P?keh?M?ori unions
Louis Hetet was a French settler who married a Maniapoto
woman. He first visited New Zealand around 1835 on a whaling
ship, and returned in 1842 to settle at Paripari (near Te
K?iti). He married Te Rangituatahi, daughter of the
influential Maniapoto chief, Taonui H?kaka. They had four
children: George Ng?tai, John Taonui, Henry Mate-ngaro, and
Mere Te Wai. The three sons proved themselves as businessmen
in the early days of the King Country. Their descendants are
well-known members of Ng?ti Maniapoto. Mere Te Wai married Te
Toko Turner, the son of another prominent P?keh?M?ori,
William Turner, who gave rise to the large Turner family.
Numerous notable Maniapoto families grew from other mixed

Finished loading Russell.ltw...

Fig. 2: An annotated screenshot of the 2010 assessment tool.

This has resulted in a tool that responds to the user's requests quickly, even for large documents containing many links to be assessed.

We also made some changes to the layout of the GUI. The previous GUI only showed information about one target document at a time, whereas the new one shows a list of the titles of all target documents to be assessed, and shows the contents of the selected target document. Rather than having every link assessed, as was done previously, we only ask the assessor to assess links whose BEPs they have deemed relevant (the assumption being that an anchor cannot be relevant if its BEP is not). Figure 2 shows a screenshot of the new GUI.

As well as improving the quality of assessments in 2010, we hope that our changes to the assessment tool will reveal further areas for improvement in 2011. Our assessment tool collects usage statistics, the analysis of which should help us improve the tool.

Even before analysing these statistics we have been able to identify one possible area for improvement. It became clear while doing the assessment that the process would have been greatly sped up if "hints" had been provided to the assessor about whether a target was likely to be relevant. As the assessment for a particular topic progressed, the assessor could build up a list of "relevant" and "non-relevant" words for that topic, which would be highlighted whenever they appeared in a candidate target document, just as the Ah Hoc tool does. The assessor could ignore this if necessary, but it would help in many cases. However, it would be very important to use such a feature carefully so as not to bias the assessment process.

5 The Ad Hoc Track

5.1 Learning Stemmers

We previously learnt suffix rewriting stemmers using Genetic Algorithms. The stemmer referred to as the Otago Stemmer is one created part way through this work. Here we use it to address one problem with using assessments to learn recall enhancing methods like stemming. Pooled collections rely on the result lists of the participants to restrict the list of documents to assess. When we later try to learn a recall enhancing method, finding documents which were not found by any participant cannot be rewarded by increases in Mean Average Precision. The goal of submitting runs with the Otago stemmer is to compare performance with the baselines where we can add documents to the pool.

Measure	Match this	Replace with this
0	shi	
2	ej	
4	ngen	
1	i	dops
4	nes	sy
0	ics	e
0	ii	sr
0	ito	ng
4	rs	tie
0	q	
4	al	
3	in	ar
0	ice	s
3	ic	
4	rs	tie
1	s	
1	f	uow
0	f	uow
0	q	
1	s	
2	que	sy
0	sl	anu
2	e	
1	f	
3	ague	dz
0	ean	

Table 2: The Otago Stemmer. Rule sections are separated by lines.

The rules of the Otago stemmer are shown in Table 2. Each rule of the stemmer uses a measure condition to ensure the length of the word is sufficient for a suffix to exist. This is taken from the Porter stemmer, and is an attempt to count the number of syllables. The measure of the word must be greater than or equal to the value for the rule. As an efficiency measure, any word to be stemmed must be longer than 3 characters. It also partitions the rules into sections. Only the first successful rule in a section is used. This was learnt on the INEX 2008 Wikipedia collection.

5.2 Refining Stemmers

We sought to improve the sets of terms that stemmers conflate. Additional terms found by the stemmer are only conflated if they are similar enough to the query term. We found a threshold value for several measures using an adaptive grid search on the INEX 2008 Wikipedia collection. Pointwise Mutual Information (PMI) and the Jaccard Index were found to aid performance, and we submitted runs using them to improve the Otago stemmer.

For both measures we used the term occurrences in documents as the probability distributions or sets to compare. For PMI, a threshold of 1.43 was found to give the best improvement. Only terms with similarity scores greater or equal to this were conflated. The PMI for two distributions x and y:

$$PMI(x,y) = log\frac{P(x,y)}{P(x)P(y)}$$

The Jaccard Index used a parameter of 0.00023 and is given between two sets of documents A and B by:

$$J(A,B) = \frac{A \cap B}{A \cup B}$$

5.3 Experimental Results

For the INEX 2010 Ad Hoc track we submitted 7 runs. Their performance is given in Table 3. These runs are combinations of stemmers and stemmer refinement.

The best run uses just the S Stripper. We find the Otago stemmer provides decent performance, and Porter to hurt performance a lot. Our baseline of no stemming occurs between the Otago and Porter stemmers.

Using PMI to improve the Otago stemmer proved successful. The Jaccard index on the same was less so. On the S stripper the Jaccard Index was found to harm performance excessively.

We forgot to submit one run, the PMI used on the S stripper. This run has been performed locally and gives a slight decrease in performance to just using the S stripper.

6 The Efficiency Track

6.1 Experiments

We conducted our experiments on a system with dual quad-core Intel Xeon E5410 2.3 GHz, DDR2 PC5300 8 GB main memory, Seagate 7200 RPM 500 GB hard drive, and running Linux with kernel version 2.6.30.

Rank	MAP	Run Name	Features
47	0.3012	v_sstem	S Stripper
54	0.2935	v_otago_w_pmi	Otago Stemmer with PMI refinement
58	0.2898	v_ostem_w_jts	Otago Stemmer with Jaccard Index refinement
59	0.2894	v_otago_stem_1	Otago Stemmer
61	0.2789	v_no_stem	No Stemming
74	0.2556	v_porter	Porter Stemmer
105	0.1102	v_sstem_w_jts	S Stripper with Jaccard Index refinement

Table 3: Stemming runs.

We conducted three sets of experiments, one for each of the *topk*, improved *topk,* and *heapk* algorithms. For the sets of experiments on the original *topk*, we used the same settings as our experiments conduced in INEX 2009. We want to compare the performance of the original *topk* with our improved *topk* and *heapk* algorithms.

The collection used in the INEX 2010 Efficiency Track is the INEX 2009 Wikipedia collection [23].

Collection Size	50.7 GB
Documents	2666190
Avg Document Length	880 words
Unique Words	11437080
Total Worlds	2347132312
Postings Size	1.2 GB
Dictionary Size	399 MB

(a)

Collection Size	50.7 GB
Documents	2666190
Avg Document Length	880 words
Unique Words	11186163
Total Worlds	2347132312
Postings Size	1.5 GB
Dictionary Size	390 MB

(b)

Table 4: (a) Summary of INEX 2009 Wikipedia Collection using term frequencies as impact values and without stemming. (b) Summary of INEX 2009 Wikipedia Collection using pre-computed BM25 as impact values and S-Striping for stemming.

The collection was indexed twice, one for the original *topk* and one for improved *topk* and *heapk*. For the original *topk*, term frequencies were used as impact values, no words were stopped and stemming was not used. For the improved topk and heapk, pre-computed BM25 similarity scores were used as impact values and S-String stemming was used. Table 4a and 4b show the summary of the document collection and statistics for the index file.

The Efficiency Track used 107 topics in INEX Ad Hoc 2010. Only title was used for each topic. All topics allow *focused*, *thorough* and *article* query evaluations. For the Efficiency Track, we only evaluated the topics for *article* Content-Only. During query evaluation, the terms for each topic were sorted in order of the maximum impact values of the terms.

For the sets of experiments on the improved *topk* and *heapk*, the whole index was loaded into memory, thus no I/O was involved at query evaluation time. For the original *topk*, only first-level dictionary was loaded into memory at start-up.

For the three sets of experiments, we specified lower-k parameter with k = 15, 150 and 1500 as required by the Efficiency Track. For each iteration of the lower-k, we specified the upper-K of 10, 100, 1 000, 10 000, 100 000, 1 000 000. In total we submitted 54 runs. The lists of run IDs and the associated lower-k and upper-K values are shown in Table 5. Officially we submitted the wrong runs for the *heapk*. The runs has been corrected and are used in this paper and the MAiP measures are generated using the official assessment tools.

Lower-k	Upper-K	Original topk	Improved Topk	Heapk
15	10	09topk-1	10topk-1	10heapk-1
15	100	09topk-2	10topk-2	10heapk-2
15	1000	09topk-3	10topk-3	10heapk-3
15	10000	09topk-4	10topk-4	10heapk-4
15	100000	09topk-5	10topk-5	10heapk-5
15	1000000	09topk-6	10topk-6	10heapk-6
150	10	09topk-7	10topk-7	10heapk-7
150	100	09topk-8	10topk-8	10heapk-8
150	1000	09topk-9	10topk-9	10heapk-9
150	10000	09topk-10	10topk-10	10heapk-10
150	100000	09topk-11	10topk-11	10heapk-11
150	1000000	09topk-12	10topk-12	10heapk-12
1500	10	09topk-13	10topk-13	10heapk-13
1500	100	09topk-14	10topk-14	10heapk-14
1500	1000	09topk-15	10topk-15	10heapk-15
1500	10000	09topk-16	10topk-16	10heapk-16
1500	100000	09topk-17	10topk-17	10heapk-17
1500	1000000	09topk-18	10topk-18	10heapk-18

Table 5: The lists of run IDs and the associated lower-k and upper-K values.

6.2 Results

This section talks about the evaluation and performance of our three sets of the runs, obtained from the official Efficiency Track (except for the *heapk*).

Figure 3 shows the MAiP measures for the original *topk*, improved *topk* and *heapk*. When upper-K has values of 150 and 1500, MAiP measures are much better than the upper-K 15. In terms of lower-k, MAiP measures approach constant at a value of 10 000. The best runs are 09topk-18 with a value of 0.2151, 10topk-18 with a value of 0.2304 and 10heapk-18 with a value of 0.2267 for the three algorithms respectively.

The MAiP measures are about the same for the improved *topk* and *heapk*. The subtle differences are when documents have the same similarity scores and the order of these documents can be different between the improved *topk* and *heapk*.

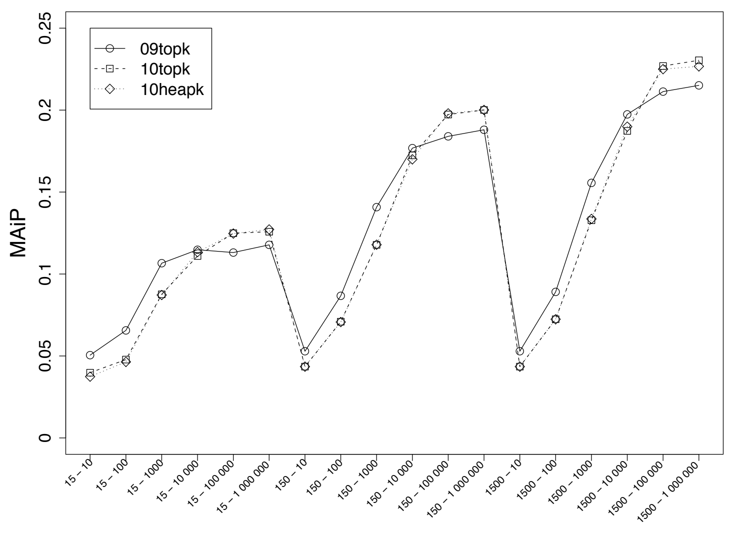


Fig. 3: MAiP measures for the original *topk*, improved *topk* and *heapk*.

The MAiP measures of the original *topk* are quite difference from the other two algorithms. Using term frequencies as impact values have better MAiP measures when the values of lower-k and uppoer-K are small while pre-computed BM25 impact values have better MAiP measures when upper-K has a value larger than 10 000.

To have a better picture of the time cost for the three sets of the runs, we plotted the total evaluation times (including both CPU and I/O times) of all runs in Figure 4. The total times of both the improved *topk* and *heapk* are simply the CPU costs since the index was load into memory.

For the original *topk*, the total times were dominated by the I/O times. Regardless of the values used for lower-k and upper-K, the same number of postings were retrieved from disk, thus causing all runs to have the same amount of disk I/O.

We also plotted the CPU times of the original *topk* since we want to compare it with the other algorithms in terms of CPU cost. The differences of the CPU times between the original *topk* and the other two algorithms are the times taken for decompression of the postings lists and sorting of the accumulators. First,

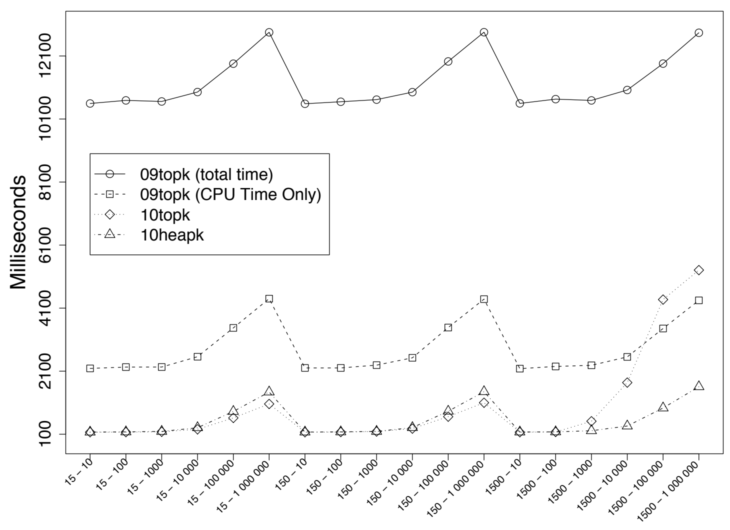


Fig. 4: Efficiency comparison.

partial decompression was used in improved *topk* and *heapk* while the original *topk* did not. Second, the original *topk* used a special version of quick sort to partially sort all accumulators while the improved *topk* and *heapk* only keep track of the top documents only the final top documents got sorted.

For the original *topk*, the value of lower-k has no effect on the CPU cost, and values of 10 000 or above for upper-K causes more CPU usage.

For the improved *topk*, it performs the best when lower-k has a value of 15 and 150. However, for the set of the runs where the value of lower-k is 1500, the performance of the improved *topk* grows exponentially. This is caused by the linearly scans of the array of pointers to insert a new document into the top k.

For the runs when lower-k has a value of 15 and 150, the *heapk* has a small overhead compared with the improved *topk*, especially when upper-K has a large value. Well, the *heapk* performs the best when both lower-k and upper-K have large values.

7 The Data Centric Track

The collection used in the INEX 2010 Efficiency Track is the 2010 IMDB collection. The collection was indexed twice. the first index used pre-computed BM25 similarity scores as the impact values and the second used pre-computer Divergence similarity scores [24] as the impact values. For both indexes, no words were stopped and S-String stemming was used. Table 6 shows the results.

Run ID	MAgP	MAiP	MAP
DC-BM25	0.2491	0.1550	0.3397
DC-DIVERGENCE	0.1561	0.1011	0.2103

Table 6: Effectiveness measure for the Data Centric Track

8 Conclusion and Future Work

8.1 The Link-the-Wiki Track

We have generated a number of runs for Te Ara. Given the inapplicability of Itakura and Geva's algorithms to Te Ara (see Section 4.1), we believe that this year's results are a step in the right direction towards a successful solution of what is still an unsolved problem: link recommendation in a corpus that has no existing links.

8.2 The Ad Hoc Track

We find that the S stripper is hard to beat. However it is possible to use machine learning to create a good stemmer. Furthermore such stemmers seem amenable to improvement using collection statistics. Of those PMI is a good measure to use. It was also found to be the best locally. This also confirms previous findings that Porter can have a variable effect on performance. Improvement using term similarity can also harm performance. We had seen this before when finding the parameters to use, so perhaps that might have been the consequence for the Jaccard Index. Of course, this refinement can only prevent terms from being stemmed together, so using it on such a weak stemmer would be expected to not do so well.

8.3 The Efficiency Track

We compared three of our query pruning algorithms. The original *topk* uses a special version of quick sort to sort all accumulators and return the top k documents. Instead of explicitly sorting all accumulators, the improved *topk* keeps tracks of the current top k documents and finally the top k documents are sorted and returned. Based on the improved *topk*, we have developed *heapk* which essentially is a minimum heap structure. The *heapk* algorithm has small overhead compared with the improved *topk* when the values of lower-k and upper-K are small. However, the *heapk* outperforms the improve *topk* for large values of lower-k and upper-K.

References

1. Huang, D., Xu, Y., Trotman, A., Geva, S.: Overview of inex 2007 link the wiki track. In Fuhr, N., Kamps, J., Lalmas, M., Trotman, A., eds.: Focused Access to

XML Documents. Volume 4862 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2008) 373–387
2. Geva, S.: Gpx: Ad-hoc queries and automated link discovery in the wikipedia. In Fuhr, N., Kamps, J., Lalmas, M., Trotman, A., eds.: Focused Access to XML Documents. Volume 4862 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2008) 404–416
3. Porter, M.: An algorithm for suffix stripping. Program **14**(3) (1980) 130–137
4. Spärck Jones, K.: Automatic Keyword Classification for Information Retrieval. Archon Books (1971)
5. Xu, J., Croft, W.B.: Corpus-based stemming using cooccurrence of word variants. ACM Trans. Inf. Syst. **16**(1) (1998) 61–81
6. Jia, X.F., Trotman, A., O'Keefe, R.: Efficient accumulator initialisation. In: Proceedings of the 15th Australasian Document Computing Symposium (ADCS2010), Melbourne, Australia (2010)
7. Trotman, A.: Compressing inverted files. Inf. Retr. **6**(1) (2003) 5–19
8. Anh, V.N., Moffat, A.: Inverted index compression using word-aligned binary codes. Inf. Retr. **8**(1) (2005) 151–166
9. Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Plachouras, V., Silvestri, F.: The impact of caching on search engines. In: SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (2007) 183–190
10. Jia, X.f., Trotman, A., O'Keefe, R., Huang, Z.: Application-specific disk I/O optimisation for a search engine. In: PDCAT '08: Proceedings of the 2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, Washington, DC, USA, IEEE Computer Society (2008) 399–404
11. Buckley, C., Lewit, A.F.: Optimization of inverted vector searches. (1985) 97–110
12. Moffat, A., Zobel, J.: Self-indexing inverted files for fast text retrieval. ACM Trans. Inf. Syst. **14**(4) (1996) 349–379
13. Tsegay, Y., Turpin, A., Zobel, J.: Dynamic index pruning for effective caching. (2007) 987–990
14. Persin, M., Zobel, J., Sacks-Davis, R.: Filtered document retrieval with frequency-sorted indexes. J. Am. Soc. Inf. Sci. **47**(10) (1996) 749–764
15. Anh, V.N., de Kretser, O., Moffat, A.: Vector-space ranking with effective early termination. (2001) 35–42
16. Trotman, A., Jia, X.F., Geva, S.: Fast and effective focused retrieval. In Geva, S., Kamps, J., Trotman, A., eds.: Focused Retrieval and Evaluation. Volume 6203 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2010) 229–241
17. Bentley, J.L., Mcilroy, M.D.: Engineering a sort function (1993)
18. Persin, M.: Document filtering for fast ranking. (1994) 339–348
19. Moffat, A., Zobel, J., Sacks-Davis, R.: Memory efficient ranking. Inf. Process. Manage. **30**(6) (1994) 733–744
20. Moffat, A., Zobel, J., Klein, S.T.: Improved inverted file processing for large text databases. (1995) 162–171
21. Anh, V.N., Moffat, A.: Random access compressed inverted files. Australian Computer Science Comm.: Proc. 9th Australasian Database Conf., ADC **20**(2) (February 1998) 1–12
22. Anh, V.N., Moffat, A.: Compressed inverted files with reduced decoding overheads. (1998) 290–297
23. Schenkel, R., Suchanek, F., Kasneci, G.: YAWN: A semantically annotated wikipedia xml corpus. (March 2007)

24. Amati, G., Van Rijsbergen, C.J.: Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans. Inf. Syst. **20**(4) (2002) 357–389

Overview of the 2010 QA Track: Preliminary results

Eric SanJuan[1], Patrice Bellot[1], Veronique Moriceau[2], and Xavier Tannier[2]

[1] LIA, Université d'Avignon et des Pays de Vaucluse (France)
{patrice.bellot,eric.sanjuan}@univ-avignon.fr
[2] LIMSI-CNRS, University Paris-Sud 11 (France)
{moriceau,xtannier}@limsi.fr

Abstract. The INEX QA track (QA@INEX) in 2009 - 2010 aims to evaluate a complex question-answering task using the Wikipedia. The set of questions is composed of factoid, precise questions that expect short answers, as well as more complex questions that can be answered by several sentences or by an aggregation of texts from different documents. This overview is centered on the long type answer QA@INEX sub track. The evaluation methodology based on word distribution divergence has allowed several summarization systems to participate. Lots of these systems generated a readable extract of sentences from top ranked documents by a state-of-the-art method. Some of the participants also tested several methods of question disambiguation. They have been evaluated on a pool of real questions from Nomao and Yahoo! Answers. Manual evaluation, as well as short type question task, are still running.

1 Introduction

The INEX QA 2009-2010 track [1] aims to compare the performance of QA, XML/passage retrieval and automatic summarization systems on the Wikipedia. Two types of questions are considered. The first type is factual questions which require a single precise answer to be found in the corpus if it exists. The second type consists of more complex questions whose answers require the aggregation of several passages. Passages might involve multi-document answer aggregation.

For both sets of questions, systems have to provide a ranked list of relevant passages. This overview is centered on results obtained on aggregated answers. Systems had to provide a document with a maximum of 500 words exclusively made of passages from the document collection (the 2008 annotated Wikipedia [2] used at INEX 2009-2010).

Systems have to make a selection of the most relevant information, the maximal length of the abstract being fixed. Focused IR systems can just return their top ranked passages meanwhile automatic summarization systems by sentence extraction need to be combined with a document IR engine.

The informative content of the resulting extracts has been evaluated by comparing their n-gram distributions with those from 4 relevant Wikipedia pages. Readability evaluation of passages by participants is undergoing.

The track is still open to short type answer runs.

2 Set of Questions

There is a total of 345 questions.

2.1 Encyclopedic *vs.* general questions

231 questions are related to 2009 INEX ad-hoc topics. The remaining questions come from commercial websites. The local search engine Nomao[3] provided us with a large pool of real questions submitted to their website. We have selected a subset of these questions such that there exists at least a partial answer in the Wikipedia 2008. Then we have mixed these questions with others from Yahoo! Answers website[4].

We considered three different types of questions: short_single, short_multiple and long.

2.2 Short type questions

Those labeled short_single or short_multiple are 195 and both require short answers, *i.e.* passages of a maximum of 50 words (strings of alphanumeric characters without spaces or punctuations) together with an offset indicating the position of the answer.

Short_single questions should have a single correct answer, *e.g.* question 216: *Who is the last olympic champion in sabre?*

whereas multiple type questions will admit multiple answers (question 209: *What are the main cloud computing service providers?*).

For both short types, participants should give their results as a ranked list of maximum 10 passages from the corpus together with an offset indicating the position of the answer. Passages have to be self-contained to decide if the answer is correct or not. Assessment for short questions takes into account the presence of the correct answer within this list and its rank.

Among the 195 short type questions, 151 are related to 2009 ad-hoc topics and 44 come from Nomao and Yahoo! Answers.

2.3 Long type questions

Long type questions require long answers up to 500 words that must be self-contained summaries made of passages extracted from the INEX 2009 corpus. Among the 150 long type questions, 80 are related to 2009 ad-hoc topics and the remaining 70 come from Nomao and Yahoo! Answers.

An example of a long type question is (#196): *What sort of health benefit has olive oil?*

[3] http://en.nomao.com/
[4] http://answers.yahoo.com/

3 Evaluation of long answers

3.1 Methodology

The informative content of the long type answers are evaluated by comparing the several n-gram distributions in participant extracts and in a set of relevant passages selected manually by organizers. We followed the experiment in [3] done on TAC 2008 automatic summarization evaluation data. This allows to evaluate directly summaries based on a selection of relevant passages.

Given a set R of relevant passages and a text T, let us denote by $p_X(w)$ the probability of finding an n-gram w from the Wikipedia in $X \in \{R, T\}$. We use standard Dirichlet smoothing with default $\mu = 2500$ to estimate these probabilities over the whole corpus. Word distributions are usually compared using one of these functions:

– Kullback Leibler (KL) divergence:

$$KL(p_T, p_R) = \sum_{w \in R \cup T} p_T(w) \times \log_2 \frac{p_T(w)}{p_R(w)}$$

– Jensen Shannon (JS) divergence:

$$JS(p_T, p_R) = \frac{1}{2}(KL(p_T, p_{T \cup R}) + KL(p_R, p_{T \cup R}))$$

In [3], the metric that obtained best correlation scores with ROUGE semi-automatic evaluations of abstracts used in DUC and TAC was JS. However, we have observed that JS is too sensitive to abstract size; therefore we finally used KL divergence to evaluate informative content extracts from participants.

We used the FRESA package[5] to compute both KL and JS divergences between n-grams ($1 \leq n \leq 4$). This package also allows to consider skip n-grams.

Evaluating informative content without evaluating readability does not make sense. It clearly appears that if readability is not considered then the best summarizer would be the random summarizer on n-grams which certainly minimizes KL divergence but produces incomprehensible texts.

The readability and coherence are evaluated according to "the last point of interest" in the answer which is the counterpart of the "best entry point" in INEX ad-hoc task. It requires a human evaluation where the assessor indicates where he misses the point of the answers because of highly incoherent grammatical structures, unsolved anaphora, or redundant passages. This evaluation is actually undergoing by participants. In the following results we evaluated general readability on pools of 10 summaries.

[5] http://lia.univ-avignon.fr/fileadmin/axes/TALNE/Ressources.html

3.2 Preliminary results

We received runs for long type questions from seven participants. All of these participants generate summaries by sentence extraction. This helps readability even if it does not ensure general coherence. Extracts made of long sentences without anaphora are often more coherent but have higher KL scores. To retrieve documents, all participants used the IR engine powered by Indri, available at track resources webpage[6].

The following preliminary results only take into account KL divergence on long-type questions from Nomao and Yahoo! Answers. For each question we have selected four highly relevant Wikipedia pages. The cumulative divergence is the sum of KL scores between participant extracts and selected pages.

ID	method	unigrams	bigrams	4 skip grams	average	readability
943	long sentences	1642.93	2252.28	2257.22	2050.81	good
857	question reformulation	1621.14	2234.57	2239.85	2031.85	average
98	focused IR	1599.29	2207.56	2212.49	2006.45	bad
92	MWT expansion	1617.35	2226.61	2231.56	2025.17	average
860	system combination	1617.6	2227.37	2232.43	2025.8	average
855	semantic expansion	1625.76	2235.21	2240.35	2033.77	average
557	JS minimization	1631.29	2237.61	2242.83	2037.24	average

Table 1. Cumulative KL divergence for best participant runs.

As expected, focused IR approach (98) using Indri minimizes KL divergence but the resulting readability is bad. This system is the same than the one used by the LIA in the Restricted Focused ad-hoc task. Meanwhile the system (943) having best readability gets highest divergence scores. The most sophisticated summary approach is the Cortex system (860) which reaches a compromise between KL divergence and readability. But query formulation to retrieve documents looks also important, the approach based on query enrichment with related MultiWord Terms (92) automatically extracted from top ranked documents, gets similar divergence scores. Surprisingly sentence JS minimization (557) does not seem to minimize overall KL divergence.

4 Status of the track and ongoing work

We expect participants to mark "the less point of interest" in a pool of abstracts. This will be done using a web interface. Simultaneously, participants will be invited to point most relevant sentences. We will then compute precise readability scores and alternative KL scores using only most relevant sentences.

The track remains open for short-type questions.

[6] http://qa.termwatch.es/

References

1. Moriceau, V., SanJuan, E., Tannier, X., Bellot, P.: Overview of the 2009 qa track: Towards a common task for qa, focused ir and automatic summarization systems. In Geva, S., Kamps, J., Trotman, A., eds.: INEX. Volume 6203 of Lecture Notes in Computer Science., Springer (2009) 355–365
2. Schenkel, R., Suchanek, F.M., Kasneci, G.: Yawn: A semantically annotated wikipedia xml corpus. In Kemper, A., Schöning, H., Rose, T., Jarke, M., Seidl, T., Quix, C., Brochhaus, C., eds.: BTW. Volume 103 of LNI., GI (2007) 277–291
3. Louis, A., Nenkova, A.: Performance confidence estimation for automatic summarization. In: EACL, The Association for Computer Linguistics (2009) 541–548

LIA at INEX 2010: Ad Hoc, Book and Question Answering Tracks

Romain Deveaud, Florian Boudin, Eric SanJuan and Patrice Bellot

Laboratoire Informatique d'Avignon - University of Avignon (CERI-LIA)
339, chemin des Meinajariès, F-84000 Avignon Cedex 9
firstname.lastname@univ-avignon.fr

Abstract. In this paper we describe our participation in the INEX 2010 Ad Hoc, Book and Question Answering tracks. In Ad Hoc and Book tracks, we experimented language modeling retrieval approaches [7] with different query expansion methods. We also propose a method for reducing the effect of word hyphenations errors on book retrieval and we experiment it on the Book track corpus.

1 Introduction

This year, LIA participated in three tracks: Ad Hoc, Question Answering (QA) and Book tracks. Our interest in the Ad Hoc track relies on the special IR context induced by Wikipedia. We wanted to check three points:

1. baseline IR system (state to the art systems used with default parameters) perform quite well on the wikipedia.
2. Query Expansion based on simple n-gram frequency (tf) improves the results on the wikipedia.
3. Ad Hoc restricted focus task can be handle using summarization methods also useful for QA.

To prove the last point we submitted one run to each task using the same system.

Our participation to Book track required much more effort. We wanted to try different Query Expansion methods on this corpus. For this purpose, we used Wikipedia as an external resource, and a state-of-the-art dependencies parser [1]. We experimented several term extraction methods for Query Expansion, and finally observed that the *entropy* method seemed to give better results on previous year *qrels*. We also observed that hidden hyphenations in texts were downgrading our results, we then set up a specific parser to solve them and produced a version of the corpus almost without these errors.

The rest of the paper is organized as follows. In Section 2, we present our first approch for the Ad Hoc and QA tracks. Then, we further develop the improvements we introduced and we conclude this section with a common focus Information Retrieval system used in both the Ad Hoc Restricted Focus and the QA Long Type Answers tasks. Finally, the Section 3 is devoted to the Book Track, starting by measuring the impact of word hyphenations correction over book retrieval and setting up two different Query Expansion processes.

2 Ad Hoc and Question Answering Tracks

This year we wanted to move away from probabilistic models in Information Retrieval (such as BM25 [9]) and experiment Language Modeling (LM) approaches. We also wanted to concentrate on Focused retrieval which is an interesting challenge. For these purposes, we chose Indri, which is part of the Lemur toolkit[1]. Indri is a search engine which combines inference network to language model [7], and also provides XML indexing.

We submitted 3 baseline runs for Document retrieval, 2 runs for the Relevance in Context and Restricted Relevance in Context tasks each and one run for the Restricted Focused task. These runs are very similar, hence we only present the best ones for each task.

2.1 Baseline

This run only performed on Document retrieval, not Focused retrieval. Queries are generated by combining the words from the `<title>` and `<phrasetitle>` fields of the topics. No weight repartition nor normalization are applied to the terms of the queries. Therefore, given a sequence of query words $Q = (q_1, ..., q_n)$, the scoring function of a document D is defined as follow :

$$s_Q(D) = \prod_{i=1}^{n} p(q_i|D)^{\frac{1}{n}} \qquad (1)$$

Where $p(q_i|D)$ is estimated using Dirichlet smoothing and $\mathcal{C}$ is the entire collection :

$$p(q_i|D) = \frac{tf_{q_i,D} + \mu \times p(q_i|\mathcal{C})}{|D| + \mu} \qquad (2)$$

We set the μ parameter to 2500, which is the default value proposed by Indri. Results for Document retrieval on the Efficiency task are reported in Table 1. For readability we will only present the N top results, where $N - 1$ is our position. In other words, we present the 3 top systems when we are ranked second and the 4 top systems when we are third.

We can see that the approach of the Peking University is by far more effective than the others. However, our run has roughly the same *MAP* score than the University of Otago's one. The run submitted by the LIP6 also shows comparable results.

2.2 Query Expansion with Pseudo-Relevance Feedback

This year we decided to concentrate on Focused Retrieval. Hence, we modified the prior baseline and we applied a simple passage selection strategy : we only select the first `<section>` field of each retrieved document. Indeed, we observed

[1] `www.lemurproject.com`

Table 1. Document retrieval results on the Efficiency task in terms of Mean Average Precision (MAP).

Institute	**Runs**	MAP
Peking University	`38P167`	0.3385
University of Otago	`OTAGO-2010-10topk-18`	0.2912
LIA - University of Avignon	`I10LIA4ElBas`	**0.2906**
University Pierre et Marie Curie - LIP6	`LIP6-OWPCRefRunTh`	0.2801

that the first `<section>` field correspond to the first paragraph of the Wikipedia page. This paragraph usually provides a general description of the topic of the Wikipedia page, and is highly informative. This is why we think they can always be considered as important passages inside relevant documents.

Apart from the passage selection, we also expand the queries with relevant terms selected by the well-known Pseudo-Relevance Feedback technique [2]. We retrieve the 10 top documents with a baseline query (see Section 2.1) and extract the 50 most frequent unigrams, the 20 most frequent 2-grams and the 10 most frequent 3-grams from the aggregated documents.

We keep the same notation as in (1), with $Q = (q_1, ..., q_n)$ as the sequence of baseline query terms, and $T_{uni} = (t_{uni,1}, ..., t_{uni,50})$, $T_{bi} = (t_{bi,1}, ..., t_{bi,20})$ and $T_{tri} = (t_{tri,1}, ..., t_{tri,10})$ the sequences of extracted terms. The scoring function is then defined as :

$$\begin{aligned} s_Q(D) = \left(\prod_{i=1}^{n} p(q_i|D)^{\frac{1}{n}}\right)^{\frac{X}{X+Y}} & \times \left(\left(\prod_{i=1}^{50} p(t_{uni,i}|D)^{\frac{1}{i}}\right)\right. \\ & \times \left(\prod_{j=1}^{20} p(t_{bi,j}|D)^{\frac{1}{j}}\right) \\ & \left.\times \left(\prod_{k=1}^{10} p(t_{tri,k}|D)^{\frac{1}{k}}\right)\right)^{\frac{Y}{X+Y}} \end{aligned} \tag{3}$$

Where $p(\cdot|D)$ are estimated using (2), X is the weight of the baseline query and Y is the weight of its expansion. For the following runs we set the weights to $X = 3$ and $Y = 2$.

Results for Focused retrieval on the Relevance in Context and Restricted Relevance in Context tasks are reported in Table 2 and Table 3.

The aim of the Relevance in Context task is to retrieve relevant passages, no matter the length of these passages. We can see that our approach ranks third, with scores comparable to the Peking University and the Queensland University of Technology. The ENSM-SE's system ranks first and shows a large improvement over the other systems.

The aim of the Restricted Relevance in Context task is the same as the Relevance in Context task, except that the passage lengths are limited to 500

Table 2. Focused retrieval results on the Relevance in Context task in terms of Mean Average Precision (MAP).

Institute	**Runs**	MAP
ENSM-SE	`Emse303R`	0.1977
Peking University	`32p167`	0.1615
LIA - University of Avignon	`I10LIA1FTri`	**0.1589**
Queensland University of Technology	`Reference`	0.1522

Table 3. Focused retrieval results on the Restricted Relevance in Context task in terms of Mean Average Precision (MAP).

Institute	**Runs**	MAP
Peking University	`32p167`	0.1580
LIA - University of Avignon	`I10LIA2FTri`	**0.1542**
Queensland University of Technology	`Reference`	0.1509

characters. For these runs, we only select the 500 first characters of the first `<section>` field of the retrieved documents. If the field's text size below than 500 characters, the system returns the entire field. We can see in Table 3 that this approach performed well and is ranked second. It also has comparable scores with the Peking University's system which is ranked first.

2.3 A summarization system for restricted focus and QA with long answers

Enertex system applies statistical physics to Natural Language Processing. The first step consists of retrieving the top 100 Wikipedia pages by using the Baseline we introduced in Section 2.1. Then, these pages are segmented into sentences with Tree-Tagger[2]. A first subset of sentences is defined by selecting the sentences that contains at least one term from the query. A second subset is defined by selecting the sentences that contains at least one term from a sentences belonging to the first subset. The union of these two subsets forms a set of relevant sentences which are ranked by an *entropy* measure. Considering a sentence $S = (w_1, ..., w_n)$, the *entropy* measure we use is defined as follow:

$$E(S) = -\sum_{i=1}^{n} p(w_i)\log_2(p(w_i)) \tag{4}$$

Where the $p(w_i)$ are computed within the whole set of selected sentences.

For the Question Answering track, the system returns this ranked list of sentences until the allowed limit number of words is reached. The preliminary results of the QA track show that this approach obtained the scores in term of information content, but the readability of the generated extracts needs to be improved.

[2] `www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`

For the Ad Hoc Restricted Focused task, we built extraction patterns using our set of sentences. Each sentence was considered as a word sequence with possible insertions, resulting in a pattern. Every passage in the collection matching at least one of the constructed patterns was retrieved using Indri and a Language Modeling approach. The results show that this approach achieves the best precision overall at low recall, but its effectiveness drops very quickly when the recall increases. Results are presented in Table 4.

Table 4. Focused retrieval results on the Restricted Focused task in terms of Mean Average Precision (MAP).

Institute	**Runs**	MAP
University Pierre et Marie Curie - LIP6	`LIP6-OWPCparentFo`	0.4125
Doshisha University	`DURF10SIXF`	0.3884
LIA - University of Avignon	`LIAenertexTopic`	**0.3434**
Peking University	`40p167`	0.3370

3 Book Track

This is our first participation in one of the INEX Book tracks. Several studies in Book Search tend to show that indexing specific parts (e.g. headers, titles or table of contents) is nearly as effective as indexing the entire content of books [6, 10]. We hypothesized that difficulty for retrieval models to cope with these documents could be partially due to the Optical Character Recognition (OCR) process used to generate the machine-encoded text. Errors are generally introduced in this process, increasing the difficulty for retrieval models to deal with these documents.

Hyphenated words are one source of errors. They are introduced to control line wrapping in the physical books, but they will be interpreted, if no correction is applied, as two different words at the indexing step. For example, the hyphenated word "*inter-*`[line_break]`*polation*", will be indexed as "*inter-*" and "*polation*".

First, we investigated the effects of word hyphenation correction on the INEX 2009 Book Track test collection. Second, we present different baselines and also more complex models involving a query expansion process using Wikipedia as an external resource following the work by Koolen *et al.* [4]. As for the Ad Hoc, all the runs we submitted use a language modeling (LM) approach to IR.

3.1 Word Hyphenation Correction

Although we observed a small amount of OCR errors in the corpus, there is a large number of hyphenations. To tackle this problem, we decided to reconstruct hyphenated words using a large lexicon made of 118,221 unique words extracted from the English Gigaword corpus[3]. We evaluated the correction impact with the official *qrels* and topics from the INEX 2009 Book Track, issued from the relevance judgements collected by 9 assessors [3].

The correction algorithm iterates through each couple of successive lines and generates word candidates from the last substring of the first line and the first substring of the second line. The candidate word is then corrected if it occurs in the lexicon.

The collection contains 613,107,923 lines, in which 37,551,834 (6,125%) were corrected by our method. To measure how much book retrieval is impacted by these corrections, we tested with three retrieval models. We use the embedded stopword list along with the standard Porter stemmer. All these methods use a LM approach to IR with different Dirichlet prior smoothing (μ) values. Queries are generated from the `<title>` fields of the INEX 2009 Book Track topics, and the number of retrieved books is set to 100. Results are reported in Table 5.

Despite the sizeable number of corrected words, the improvement is relatively low even though it could be statistically significant if it was measured on twice the number of queries. The large length of books (118,000 words on average) also tends to reduce the errors introduced by some misspelled words. We used the corrected version for all the runs we submitted.

Table 5. Book retrieval results on both initial and corrected INEX 2009 Book Track corpus in terms of Mean Average Precision (MAP) and precision at 10 (P@10).

Model	**Uncorrected data**		**Corrected data**	
	MAP	P@10	MAP	P@10
LM, $\mu = 2500$	0.302	0.486	0.304	0.507
LM, $\mu = 1000$	0.299	0.493	0.302	0.507
LM, $\mu = 0$	0.244	0.443	0.243	0.450

3.2 Baselines

We submitted two simple models as baselines. Queries are treated as a bag of words and retrieval is performed using a LM approach with μ set to 2000. The first baseline (namely **baseline_1**) uses the content of the `<query>` fields of the topics, while the second one (**baseline_2**) uses the content of the `<fact>` fields.

3.3 Query Expansion using Wikipedia

Several studies previously investigated the use of Wikipedia as an external corpus for Query Expansion [5]. In their approach, Koolen *et al.* [4] extract useful terms from Wikipedia pages to expand queries, and use them for Book Retrieval.

[3] LDC Catalog No. LDC2007T07, Available at `www.ldc.upenn.edu`

Indeed, a page is selected by querying Wikipedia with the original query and getting the page that match the query, or the best result. The well-known *tf.idf* measure is then computed for each word of the selected Wikipedia page, and the expanded query is formed by adding the top-ranked N words to the original query. The *idf* values are computed within the whole test collection. They employ a simple term weighting method: the original query terms are weighted N times more than the N added terms.

We started by expanding the baseline models described in Section 3.2. We used the `<wikiurl>` field, when available, to get a Wikipedia page for each topic. Otherwise we queried the Wikipedia search engine with the `<query>` of the topic and we kept its best result. Terms contained in the upper part of the Wikipedia page (small introductory summary) are selected to expand the queries. Two runs, one for each baseline, were submitted: **baseline_1_wikifact**, **baseline_2_wikifact**.

In the following runs, we used the `<query>` field as the original query. We also noticed that the `<fact>` field was most of the time a *cut-and-paste* sentence from a book, therefore we used it as a first query expansion. Therefore, given a sequence of query terms $Q = (q_1, ..., q_k)$, a sequence of fact terms $F = (f_1, ..., f_m)$ and a list of weighted terms $T_Q = \{(t_1, w_1), ..., (t_n, w_n)\}$ extracted from related Wikipedia pages, we shall rank books according to the following scoring function $\Delta_Q(D)$:

$$\begin{aligned}\Delta_Q(D) = &\left(\prod_{i=1}^{k} p_D(q_i)^{\frac{1}{k}}\right)^{\frac{X}{X+Y+Z}} \times \\ &\left(\prod_{i=1}^{m} p_D(f_i)^{\frac{1}{m}}\right)^{\frac{Y}{X+Y+Z}} \times \\ &\left(\prod_{i=1}^{n} p_D(t_i)^{\frac{w_i}{\sum_{j=1}^{n} w_j}}\right)^{\frac{Z}{X+Y+Z}}\end{aligned}$$

where $p_D(q)$ is estimated using Dirichlet smoothing (2), with $\mu = 2500$.

Here, we couldn't learn an appropriate weighting scheme for the X, Y and Z weights, so they were set empirically. We gave the same weight to the `<title>` and the `<fact>` fields ($X = Y = 4$), whereas the expansion terms were weighted half ($Z = 2$). We use these weights for all the runs featuring Wikipedia query expansion.

In the following runs, we split the Wikipedia pages into chunks with TreeTagger. Therefore, a term can be composed of one or many words.

Fact_query_tfwiki Run In this run, the terms from the associated Wikipedia page are ranked by tf, and the top 10 ones are selected for the expansion. Their scores are also normalized inside the expansion in order to weight appropriately the important words.

Fact_query_tfidfwiki Run This run is practically the same as above, except that we rank the terms by $tf.idf$, where the *idf* is computed whithin the whole collection. The scores are also normalized and used in the expansion.

Fact_query_entropy Run This run is similar to the `fact_query_tfidfwiki` run but the term selection measure is only computed within the associated Wikipedia page. We use the *entropy* measure defined in the Section 2.3 (4).

Fact_query_10bestswiki Run This run is a sort of query expansion baseline. Indeed we didn't normalized the scores inside the expansion and the selected terms are words and not chunks from Tree-Tagger. As for the prior runs, the 10 most frequent words are selected for the expansion.

The results for the Book Track are not currently available. We however experimented our approach with the topics and the *qrels* from the INEX Book Track 2009, and the *entropy* term selection measure seems to lead to better results.

3.4 Using the Stanford Parser

In these models, we consider multiword phrases. It is clear that finding the exact phrase "*New York*" is a much stronger indicator of relevance than just finding "*New*" and "*York*" scattered within a document. We use Metzler and Croft's Markov Random Field model [8] to integrate that. In this model three features are considered: single term features (standard unigram language model features), exact phrase features (words appearing in sequence) and unordered window features (require words to be close together, but not necessarily in an exact sequence order). Features weights are set according to the authors's recommendation. Multiword phrases are detected using the Stanford parser [1]. In this work, we use the typed dependency representation of the `<fact>` fields to extract complex noun phrases (e.g. "*london daily mail*", "*sioux north american shields*" or "*symphony no 3*"). The submitted run is named **fact_stanford_deps**.

4 Conclusions

In this paper we experimented Focused retrieval with a language modeling approach and traditional query expansion with Pseudo-Relevance Feedback. This year, this approach proved its efficiency and robustness, and it encourages us to further explore this path. We also proposed to enhance Book Search performance by correcting word hyphenations. Although we cannot see a significant improvement on a Book Retrieval task, we expect that this approach can perform better in focused search tasks such as page or extent retrieval. We plan to extend this study when the *qrels* of the INEX 2010 Book track will be available.

References

1. M.C. De Marneffe, B. MacCartney, and C.D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC'06 conference*, 2006.
2. D. Harman. Relevance feedback revisited. In *Proceedings of the SIGIR'92 conference*, pages 1–10, 1992.
3. G. Kazai, A. Doucet, M. Koolen, and M. Landoni. Overview of the INEX 2009 Book Track. *Focused Retrieval and Evaluation*, pages 145–159, 2010.
4. M. Koolen, G. Kazai, and N. Craswell. Wikipedia pages as entry points for book search. In *Proceedings of the WSDM'09 conference*, pages 44–53, 2009.
5. Y. Li, W.P.R. Luk, K.S.E. Ho, and F.L.K. Chung. Improving weak ad-hoc queries using wikipedia as external corpus. In *Proceedings of the SIGIR'07 conference*, pages 797–798, 2007.
6. W. Magdy and K. Darwish. Book search: indexing the valuable parts. In *Proceeding of the BooksOnline'08 workshop*, pages 53–56, 2008.
7. D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Inf. Process. Manage.*, 40:735–750, September 2004.
8. D. Metzler and W.B. Croft. A Markov random field model for term dependencies. In *Proceedings of SIGIR'05 conference*, pages 472–479, 2005.
9. S. Robertson and K. Sparck Jones. *Relevance weighting of search terms*, pages 143–160. Taylor Graham Publishing, London, UK, UK, 1988.
10. H. Wu, G. Kazai, and M. Taylor. Book search experiments: Investigating IR methods for the indexing and retrieval of books. *Advances in Information Retrieval*, pages 234–245, 2008.

The GIL-UNAM-3 summarizer: an experiment in the track QA@INEX'10

Edmundo-Pavel Soriano-Morales, Alfonso Medina-Urrea, Gerardo Sierra and Carlos-Francisco Méndez-Cruz

Instituto de Ingeniería
Universidad Nacional Autónoma de México, Mexico.
{esorianom,amedinau,gsierram,cmendezc}@iingen.unam.mx}
http://www.iling.unam.mx

Abstract. In this paper we describe the GIL-UNAM-3 summarizer, which extracts utterances from a set of documents retrieved by means of synonym modified queries. That is, we modify each query by obtaining from the WordNet database word synonyms for each of its words. The queries are provided by the INEX@QA 2010 task. The results of the experiment are evaluated automatically by means of the FRESA system.

Key words: INEX, Automatic summarization system, Question-Answering system, WordNet synonyms, GIL-UNAM-3 summarizer, FRESA system.

1 Introduction

A wide variety of text extraction techniques for summarizing documents exists (see, for instance, [4, 5]). Our experience with summarization systems has included mainly word information content, sentence or utterance position, and standard deviation of utterance position measurements ([1–3]). In this paper, we explore the implementation of a summarizer based on a) query enrichment by means of word synonyms and b) token occurrence in both the query and the document summarized. We call this system the GIL-UNAM-3 summarizer.

This system is evaluated in the context of the question-answering task of the INEX@QA 2010 track,[1] which provides questions either with short answers or with complex answers dealing with one or more utterances, possibly very long ones. Our summarizer is proposed as a tool for dealing with an experiment on this second challenge.

The corpus used contains all English texts of Wikipedia. The idea is to retrieve relevant documents by means of synonym enriched queries. The documents are retrieved by the search engine INDRI.[2] Then, by using the GIL-UNAM-3 summarizer, we provide answers which are extracts from these retrieved documents. Each extract has no more than 500 words. We then evaluate the answers automatically by means of the FRESA system ([6–8]).

[1] http://www.inex.otago.ac.nz/
[2] http://www.lemurproject.org/indri/

The organization of this paper is as follows: in Section 2 we give details on the methodology behind the summarizer proposed; in Section 3 we present the algorithm used in order to modify queries; the experimental settings and results are presented in Section 4; and, finally, in Section 5, we briefly present our conclusions and future work.

2 The GIL-UNAM-3 summarizer

The **GIL-UNAM-3** summarizer is an utterance extraction system for single-documents. The method proposed deals with two key aspects: a document representation scheme and an utterance weighting procedure. The former is accomplished by vector representation and the latter by means of a greedy optimization algorithm. The summary is build by concatenating utterances exhibiting the highest weightings according to the algorithm. In order to accomplish this, the GIL-UNAM-3 summarizer is composed of the following stages: document transformation to vector representation, calculation of utterance weightings, and summary generation.

In the first stage, a lemmatizer (*stemmer*) and a stop-list, both tailor-made for the English language, are used in order to filter the document tokens so that same-root forms are conflated and function words are eliminated.

In the second stage, the optimization algorithm is applied to build a matrix and to estimate utterance weightings. These weightings are calculated by means of the following two values:

1. Cosine measurements of the angles related between each utterance vector and the modified query vector. These values are normalized to fall in the interval [0,1].
2. Normalized word frequencies in both the document and the modified query (also in [0,1]).

Furthermore, each utterance weighting is simply the arithmetic mean of both of these values calculated for each utterance, which is also a value in the interval [0,1].

In the last stage, the relevant utterances, those with the highest weightings, are extracted and concatenated in order to build the summary.

3 Query Modification

To modify the queries, all content words in the original query were replaced by those synonyms found in the synsets of the lexical database WordNet.[3] In fact, since reliance on WordNet for obtaining synonyms is questionable, we are actually dealing with candidate synonyms rather than real synonyms.

The algorithm for replacing each word of the question with the first different word found in the synsets obtained from WordNet was implemented with Python

[3] `http://wordnet.princeton.edu/`

2.7,[4] using the Natural Language Toolkit (NLTK)[5] suite of libraries. This allows fast tokenization and easy access to the WordNet database.

The steps of the algorithm are:

1. For each query:
    (a) Tokenize the original query via regular expressions keeping the order of appearance of each token.
    (b) Remove all tokens that contain non alphabetic characters. Also remove tokens contained in a stoplist.
    (c) For each remaining token:
        i. Search in WordNet the token and keep the first different word found in the synsets. this word will be the new token to be exchanged with the original one.
    (d) Replace the old tokens from the original query with the new ones, keeping the same order of appearance.

This can be also described in pseudocode:

```
Input: List of original queries $OQ$
Output: List of modified queries $MQ$
foreach $originalquery \in OQ$ do
    $OriginalTokens \leftarrow$ Tokenize($originalquery$) ;
    $NotFunctional \leftarrow$ RemoveFunctionalTokens($OriginalTokens$);
    foreach $token \in NotFunctional$ do
        $Synonym \leftarrow$ GetSynonymFromWordNet($token$);
        $NewTokens$ += $Synonym$;
    end
    $ModifiedQuery \leftarrow$ ReplaceTokens($OriginalTokens, NewTokens$);
    $NewQueries$ += $ModifiedQuery$;
end
$OQ \leftarrow NewQueries$;
return $OQ$;
```

Algorithm 1: ModifyOriginalQueries(OQ)

The most interesting method, **GetSynonymFromWordNet**, is described also:

[4] `http://www.python.org/about/`
[5] `http://www.nltk.org/`

```
**Input**: token $T$
**Output**: synonym $S$
$Result \leftarrow LookupWordInWordNet(T)$;
$S \leftarrow T$;
**if** *Result* ***is not*** *empty* **then**
    **foreach** $synset \in Result$ **do**
        **foreach** $word \in synset$ **do**
            **if** $word \neq T$ **then**
                $S \leftarrow word$;
                **return** $S$;
            **end**
        **end**
    **end**
**else**
    **return** $S$;
```

Algorithm 2: GetSynonymFromWordNet(T)

4 Experiments Settings and Results

As mentioned above, in order to evaluate the performance of the GIL-UNAM-3 summarizer, applied on the INEX@QA corpus, we used the FRESA system, which does not rely on human produced summaries or human validation. The results of the experiment can be observed in Table 1. Values represent divergence of the summaries with respect to the original documents. On the one hand, the random summaries (unigram and 5-gram) exhibit smaller divergence values than our summaries. However, these summaries are unintelligible (given their randomness).

Table 1. FRESA results for modified query number 2009071

Distribution type	**unigram**	**bigram**	**with 2-gap**	**Average**
Baseline summary	14.46978	22.27342	22.19415	19.64579
Empty baseline	19.32973	27.98158	27.87586	25.06239
Random unigram	11.94329	20.80448	20.57957	17.77578
Random 5-gram	10.777	18.08401	18.30523	15.72208
Submitted summary	13.92082	21.7732	21.7724	19.15548

On the other hand, it can be observed that all divergence values of the summaries generated by our summarizer (the submitted ones) are somehow smaller than the values of the baseline and noticeably smaller than those of the empty baseline summaries (which consists of the words in the original documents not included in our summaries).

5 Conclusions

In this brief paper, we have presented an experiment on the document sets made available during the Initiative for the Evaluation of XML Retrieval (INEX) 2010[6], in particular on the INEX 2010 QA Track (QA@INEX) `http://www.inex.otago.ac.nz/tracks/qa/qa.asp`.

We have described the GIL-UNAM-3 summarizer, which extracts utterances from a set of documents retrieved by means of synonym modified queries. That is, we obtain from the WordNet database word synonyms for each word of said query. The system applies several utterance selection metrics in order to extract those most likely to summarize a document; namely, normalized cosine measurements and normalized word frequencies. As mentioned before, the queries are provided by the INEX@QA 2010 task. The results of the experiment are evaluated automatically by means of the FRESA system. Interestingly, our system performs somehow better than the baselines.

Many adjustments can be made to this experiment. For instance, as future work, it would be interesting to modify the queries by adding synonyms, rather than replacing the query words by them, like it was done here.

References

1. Méndez C, Carlos F and Medina U, Alfonso, "Extractive Summarization Based on Word Information and Sentence Position", *Computational Linguistics and Intelligent Text Processing*, CICLing 2005, 653–656, Springer, *Lecture Notes in Computer Science*, 3406.
2. Gutiérrez Vasques, M Ximena, *Sistema de resumen extractivo automático*, Facultad de Ingeniería, UNAM, Mexico, 2010.
3. Medina U, Alfonso, "De la palabra gráfica al texto: sobre la extracción de enunciados para el resumen automático", In: Vázquez Laslop, M E and Zimmermann, Klaus and Segovia, Francisco, eds., *De la lengua por sólo la extrañeza: estudios de lexicología, norma lingüística, historia y literatura en homenaje a Luis Fernando Lara*, El Colegio de México, Mexico, forthcoming.
4. Mani, Inderjeet and Maybury, Mark T, *Automatic Text Summarization*, The MIT Press, 1999.
5. Weiss, Sh M and Indurkhya, N and Zhang, T and Damerau, F, *Text Mining. Predictive Methods for Analizing Unstructured Information*, Springer, 2005.
6. Saggion, H and Torres-Moreno, J M and da Cunha, I and SanJuan, E, and Velásquez-Morales, P, *Multilingual Summarization Evaluation without Human Models*, Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010, Beijing, 2010.
7. Torres-Moreno, J M and Saggion, H and da Cunha, I and SanJuan, E, *Summary Evaluation with and without References*, Polibits Research Journal on Computer Science and Computer Engineering and Applications, 42, 2010.
8. Torres-Moreno, J M and Saggion, H and da Cunha, I and Velázquez-Morales, P and SanJuan, E, *Évaluation automatique de résumés avec et sans référence*, 17e Conférence sur le Traitement Automatique des Langues Naturelles, TALN, Montreal, 2010.

[6] `http://www.inex.otago.ac.nz/`

The Cortex automatic summarization system at the QA@INEX track 2010

Juan-Manuel Torres-Moreno and Michel Gagnon

École Polytechnique de Montréal - Département de génie informatique
CP 6079 Succ. Centre Ville H3C 3A7 Montréal (Québec), Canada.
`juan-manuel.torres@univ-avignon.fr,michel.gagnon@polymtl.ca`

Abstract. The Cortex system is constructed of several different sentence selection metrics and a decision module. Our experiments have shown that the Cortex decision on the metrics always scores better than each system alone. In the INEX@QA 2010 task of Long Questions, Cortex strategy system obtained very good results in the automatic evaluations FRESA.

Key words: INEX, Automatic summarization system, Question-Answering system, Cortex.

1 Introduction

Automatic summarization is indispensable to cope with ever increasing volumes of valuable information. An abstract is by far the most concrete and most recognized kind of text condensation [1]. We adopted a simpler method, usually called *extraction*, that allow to generate summaries by extraction of pertinence sentences [2, 3]. Essentially, extracting aims at producing a shorter version of the text by selecting the most relevant sentences of the original text, which we juxtapose without any modification. The vector space model [4, 5] has been used in information extraction, information retrieval, question-answering, and it may also be used in text summarization. CORTEX[1] is an automatic summarization system, recently developed [6] which combines several statistical methods with an optimal decision algorithm, to choose the most relevant sentences.

An open domain Question-Answering system (QA) has to precisely answer a question expressed in natural language. QA systems are confronted with a fine and difficult task because they are expected to supply specific information and not whole documents. At present there exists a strong demand for this kind of text processing systems on the Internet. A QA system comprises, *a priori*, the following stages [7]:

– Transform the questions into queries, then associate them to a set of documents;

[1] *COndenss et Rsums de TEXte* (Text Condensation and Summarization).

- Filter and sort these documents to calculate various degrees of similarity;
- Identify the sentences which might contain the answers, then extract text fragments from them which constitute the answers. In this phase an analysis using Named Entities (NE) is essential to find the expected answers.

Most research efforts in summarization emphasize generic summarization [8–10]. User query terms are commonly used in information retrieval tasks. However, there are few papers in literature that propose to employ this approach in summarization systems [11–13]. In the systems described in [11], a learning approach is used (performed). A document set is used to train a classifier that estimates the probability that a given sentence is included in the extract. In [12], several features (document title, location of a sentence in the document, cluster of significant words and occurrence of terms present in the query) are applied to score the sentences. In [13] learning and feature approches are combined in a two step system: a training system and a generator system. Score features include short length sentence, sentence position in the document, sentence position in the paragraph, and tf.idf metrics. Our generic summarization system includes a set of ten independent metrics combined by a Decision Algorithm. Query-based summaries can be generated by our system using a modification of the scoring method. In both cases, no training phase is necessary in our system.

This paper is organized as follows. In Section 2 we explain the methodology of our work. Experimental settings are results are presented in Section 3. Section 4 exposes the conclusions of the paper and the future work.

2 The CORTEX system

CO*ndensation et* ***R****ésumés de* ***T****extes* (CORTEX) [14, 15] is a single-document extract summarization system using an optimal decision algorithm that combines several metrics. These metrics result from processing statistical and informational algorithms on the document vector space representation.

The INEX 2010 Query Task evaluation is a real-world complex question (called long query) answering, in which the answer is a summary constructed from a set of relevant documents. The documents are parsed to create a corpus composed of the query and the the multi-document retrieved by Indri.

The idea is to represent the text in an appropriate vectorial space and apply numeric treatments to it. In order to reduce complexity, a preprocessing is performed to the topic and the document: words are filtered, lemmatized and stemmed.

The CORTEX system can use up to $\Gamma = 11$ metrics [16] to evaluate the sentence's relevance.

- The frequency of words (F).
- The overlap between the words of query (R).
- The entropy the words (E).
- The shape of text (Z).
- The angle between the topic and the sentence (A).

– The sum of Hamming weights of words per segment times the number of different words in a sentence.
– The sum of Hamming weights of the words multiplied by word frequencies.
– ...

The specific similarity measure [17] between the query and the corpus allows us to re-scale the sentence scores according to the relevance of the document from which they are extracted. This measure is the normalized scalar product of the Tf-Idf vectorial representations ($\boldsymbol{v_d}$, $\boldsymbol{w_t}$) of the document d and the topic t.

$$Similarity(t,d) = \frac{\sum \boldsymbol{v_d}.\boldsymbol{w_t}}{\sqrt{\sum \boldsymbol{v_d}^2 + \sum \boldsymbol{w_t}^2}}$$

The overlap assigns a higher ranking for the sentences containing topic words and makes selected sentences more relevant. The overlap is defined as the normalized cardinality of the intersection between the topic word set T and the sentence word set S.

$$Overlap(T,S) = \frac{card(S \cap T)}{card(T)}$$

The system scores each sentence with a decision algorithm which relies on the normalized metrics. Two averages are calculated, a positive $\lambda_s > 0.5$ and a negative $\lambda_s < 0.5$ tendency (the case $\lambda_s = 0.5$ is ignored). The following algorithm combines the vote of each metric:

$$\sum^{s} \alpha = \sum_{v=1}^{\mathit{\Gamma}} (||\lambda_s^v|| - 0.5); \quad ||\lambda_s^v|| > 0.5$$
$$\sum^{s} \beta = \sum_{v=1}^{\mathit{\Gamma}} (0.5 - ||\lambda_s^v||); \quad ||\lambda_s^v|| < 0.5$$

$\mathit{\Gamma}$ is the number of metrics and v is the index of the metrics. The value given to each sentence s is calculated with:

$$if(\sum^{s} \alpha > \sum^{s} \beta)$$

then $Score_s^{cortex} = 0.5 + \sum^{s} \alpha/\mathit{\Gamma}$: retain s
else $Score_s^{cortex} = 0.5 - \sum^{s} \beta/\mathit{\Gamma}$: not retain s

CORTEX will be used as a user-oriented multi-document summarization system by using two parameters : the topic-document similarity and the topic-sentence overlap. The CORTEX system is applied to each document of a topic set and the summary is generated by concatenating higher score sentences.

The final score of a sentence s from a document d and a topic t is the following:

$$\begin{aligned} Score = \alpha_1 \; Score^{cortex}_{s,d} + \alpha_2 \; Overlap_{s,t} \\ + \alpha_3 \; Similarity_{d,t}; \end{aligned}$$

$\sum_i \alpha_i = 1$

3 Experiments Settings and Results

In this study, we used the document sets made available during the Initiative for the Evaluation of XML retrieval (INEX) 2010[2], in particular on the INEX 2010 QA Track (QA@INEX) `http://www.inex.otago.ac.nz/tracks/qa/qa.asp`.

To evaluate the efficacity of Cortex on INEX@QA corpus, we have used the FRESA package [18–20]. FRESA package is disponible at web site: `http://lia.univ-avignon.fr/fileadmin/axes/TALNE/downloads/index_fresa.html`.

INEX queries None pre-processing or modification was applied on queries set. Cortex used the query as a title of a big document retrieved by Indri. Table 1 shows an example of the results obtained by Cortex systema using 50 documents as input. The query that the summary should answer in this case was the number 2010111:

What is considered a beautiful body shape?.

This table presents Cortex results in comparison with an the INEX baseline (Baseline summary), and three baselines, that is, summaries including random n-grams (Random unigram) and 5-grams (Random 5-gram) and empty baseline. We observe that our system is always better than Baseline summary and empty baseline.

Table 1. Example of Cortex Summarization results.

Summary type	1-gram	2-gram	SU2-gram	FRESA Average
Baseline summary:	26.67949	34.10809	34.21768	31.66842
Empty baseline:	31.71499	39.45237	39.53447	36.90061
Random unigram:	25.06349	32.82227	32.85178	30.24585
Random 5-gram:	23.16818	30.64380	30.83843	28.21680
Cortex summary:	**26.42140**	33.93461	34.02986	**31.46196**

[2] `http://www.inex.otago.ac.nz/`

4 Conclusions

We have presented the Cortex summarization system that is based on the fusion process of several different sentence selection metrics. The decision algorithm obtains good scores on INEX-2010, indicating that the decision process is a good strategy for preventing overfitting on the training corpus. In the INEX-2010 corpus, Cortex system obtained very good results in the automatic FRESA evaluations.

References

1. ANSI. American National Standard for Writing Abstracts. Technical report, American National Standards Institute, Inc., New York, NY, 1979. (ANSI Z39.14.1979).
2. H. P. Luhn. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, 2(2):159, 1958.
3. H. P. Edmundson. New Methods in Automatic Extracting. *Journal of the ACM (JACM)*, 16(2):264–285, 1969.
4. Gregory Salton. *The SMART Retrieval System - Experiments un Automatic Document Processing*. Englewood Cliffs, 1971.
5. Gregory Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
6. Juan-Manuel Torres-Moreno, Patricia Velazquez-Morales, and Jean-Guy Meunier. Condenss automatiques de textes. *Lexicometrica. L'analyse de donnes textuelles : De l'enqute aux corpus littraires*, Special(www.cavi.univ-paris3.fr/lexicometrica), 2004.
7. C. Jacquemin and P. Zweigenbaum. Traitement automatique des langues pour l'accs au contenu des documents. *Le document en sciences du traitement de l'information*, 4:71–109, 2000.
8. Jose Abracos and Gabriel Pereira Lopes. Statistical Methods for Retrieving Most Significant Paragraphs in Newspaper Articles. In Inderjeet Mani and Mark T. Maybury, editors, *ACL/EACL97-WS*, Madrid, Spain, July 11 1997.
9. Simone Teufel and Marc Moens. Sentence Extraction as a Classification Task. In Inderjeet Mani and Mark T. Maybury, editors, *ACL/EACL97-WS*, Madrid, Spain, 1997.
10. Eduard Hovy and Chin Yew Lin. Automated Text Summarization in SUMMARIST. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 81–94. The MIT Press, 1999.
11. Julian Kupiec, Jan O. Pedersen, and Francine Chen. A Trainable Document Summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 68–73, 1995.
12. Anastasios Tombros, Mark Sanderson, and Phil Gray. Advantages of Query Biased Summaries in Information Retrieval. In Eduard Hovy and Dragomir R. Radev, editors, *AAAI98-S*, pages 34–43, Stanford, California, USA, March 23–25 1998. The AAAI Press.
13. Judith D. Schlesinger, Deborah J. Backer, and Robert L. Donway. Using Document Features and Statistical Modeling to Improve Query-Based Summarization. In *DUC'01*, New Orleans, LA, 2001.

14. J.M. Torres-Moreno, P. Velazquez-Moralez, and J. Meunier. *CORTEX, un algorithme pour la condensation automatique de textes*. In *ARCo*, volume 2, page 365, 2005.
15. Juan Manuel Torres-Moreno, Pier-Luc St-Onge, Michel Gagnon, Marc El-Bèze, and Patrice Bellot. Automatic summarization system coupled with a question-answering system (qaas). *in CoRR*, abs/0905.2990, 2009.
16. J.M. Torres-Moreno, P. Velazquez-Morales, and J.G. Meunier. *Condensés de textes par des méthodes numériques. JADT*, 2:723–734, 2002.
17. G. Salton. *Automatic text processing*, chapter 9. Addison-Wesley Longman Publishing Co., Inc., 1989.
18. J.-M. Torres-Moreno, H. Saggion, I. da Cunha, P. Velázquez-Morales, and E. SanJuan. Evaluation automatique de rsums avec et sans rfrence. In *17e Confrence sur le Traitement Automatique des Langues Naturelles (TALN)*, 2010.
19. J-M. Torres-Moreno, H. Saggion, I. da Cunha, E. SanJuan, and P. Velázquez-Morales. Summary evaluation with and without references. *Polibitis: Research journal on Computer science and computer engineering with applications*, 42:13–19, 2010.
20. H. Saggion, J-M. Torres-Moreno, I. da Cunha, E. SanJuan, and P. Velázquez-Morales. Multilingual summarization evaluation without human models. In *23rd International Conference on Computational Linguistics (COLING 2010), Pekin*, 2010.

Using Textual Energy (Enertex) at QA@INEX track 2010

Andrea Carneiro Linhares[1] and Patricia Velázquez[2]

[1] Universidade Federal do Ceara
Fortaleza, Brasil
[2] VM Labs
France
andreaclinhares@gmail.com,patricia_velazquez@yahoo.com

Abstract. In this paper we present a Neural Network approach, inspired by statistical physics of magnetic systems, applied to NLP problems. We obtained good results on the application of this method to automatic summarization. In the INEX@QA 2010 task of Long Questions, Enertex strategy system obtained good results in the automatic evaluations FRESA.

Key words: INEX, Automatic summarization system, Question-Answering system, Enertex.

1 Introduction

Hopfield [1, 2] took as a starting point physical systems like the magnetic Ising model (formalism resulting from statistical physics describing a system composed of units with two possible states named spins) to build a Neural Network (NN) with abilities of learning and recovery of patterns. The capacities and limitations of this Network, called associative memory, were well established in a theoretical frame in several studies [1, 2]: the patterns must be not correlated to obtain free error recovery, the system saturates quickly and only a little fraction of the patterns can be stored correctly. As soon as their number exceeds $\approx 0,14N$, any pattern is recognized. This situation strongly restricts the practical applications of Hopfield Network. However, in NLP, it is possible to exploit this behavior. Vector Space Model (VSM) [3] represents the sentences of a document into vectors. These vectors can be studied as Hopfield NN. With a vocabulary of N terms of a document, it is possible to represent a sentence as a chain of N neurons actives (words are presents) or inactives (words are absents). A document with P sentences is formed of P chains in the vector space Ξ of dimension N. These vectors are correlated according to the shared words. If thematics are close, it is raisonable to suppose that the degree of correlation will be very high. That is a problem if we want to store and recover these representations from a Hopfield NN. However, our interest does not relate with recovery, but to study the interactions between the terms and the sentences. From these interactions [4, 5] have defined the Textual Energy of a document. It can be to score sentences in order to obtain a summary of a document.

This paper is organized as follows. In Section 2 we explain the methodology of our work. Experimental settings are results are presented in Section 3. Section 4 exposes the conclusions of the paper and the future work.

2 The Enertex system

The Textual Energy [4, 5] can be expressed:

$$E = -\frac{1}{2} S \times J \times S^T \; ; \; E_{\mu,\nu} \in E_{[P \times P]} \tag{1}$$

2.1 Textual Energy: a new similarity measure

We are going to explain the nature of the links between sentences that Textual Energy infers. To do that, we use some elementary notions of the graph theory. The interpretation that we are going to do, is based on the fact that the matrix (1) can be written:

$$E = -\frac{1}{2} S \times (S^T \times S) \times S^T = -\frac{1}{2} (S \times S^T)^2 \tag{2}$$

Textual Energy can be used as a similarity measure in NLP applications. In an intuitive way, this similarity can be used in order to score the sentences of a document and thus separate those which are relevant from those which are not. This leads immediately to a strategy for automatic summarization by extraction of sentences.

The summarization algorithm includes three modules. The first one makes the vectorial transformation of the text with filtering, lemmatisation/stemming and standardization processes. The second module applies the spins model and makes the calculation of the matrix of textual energy (2). We obtain the weighting of a sentence ν by using its absolute energy values, by sorting according to $\sum_\mu |\boldsymbol{E_{\mu,\nu}}|$. So, the relevant sentences will be selected as having the biggest absolute energy. Finally, the third module generates summaries by displaying and concatenating of the relevant sentences. The two first modules are based on the Cortex system[1].

Enertex system has been used in automatic mono and multidocument in three languages: english, french and spanish.

3 Experiments Settings and Results

In this study, we used the document sets made available during the Initiative for the Evaluation of XML retrieval (INEX) 2010[2], in particular on the INEX 2010 QA Track (QA@INEX) `http://www.inex.otago.ac.nz/tracks/qa/qa.asp`.

[1] The Cortex system [6–8] is an unsupervised summarizer of relevant sentences using several metrics controlled by an algorithm of decision.

[2] `http://www.inex.otago.ac.nz/`

To evaluate the efficacity of Enertex on INEX@QA corpus, we have used the FRESA package [9–11] [3].

3.1 INEX queries

A simple modification was applied on queries set. We have manually enriched the original query with the synonyms of a term. Enertex used the query as a title of a big document retrieved by Indri. Table 1 and 2 show two examples of the results obtained by Enertex systema using five documents as input.

The queries that the summary should answer in these cases was the number 2009006 :

What are the similarities and differences between mean average precision and reciprocal rank used in Information Retrieval?

and 2009009:

Who are the people, chemists, physicists and even alchemists, who studied elements and the periodic table?.

These tables present Enertex results in comparison with an the INEX baseline (Baseline summary), and three baselines, that is, summaries including random n-grams (Random unigram) and 5-grams (Random 5-gram) and empty baseline. We observe that summaries produced by Enertex system are close of Baseline summary.

Table 1. Example of Enertex Summarization results on 2009006 question.

Summary type	1-gram	2-gram	SU2-gram	FRESA Average
Baseline summary:	15.92107	22.67403	23.02247	20.53919
Empty baseline:	25.51794	33.38139	33.58377	30.82770
Random unigram:	11.91236	19.89169	19.96834	17.25747
Random 5-gram:	13.41264	20.26874	20.82560	18.16899
Enertex summary:	17.93214	25.54367	25.76446	23.08009

Table 2. Example of Enertex Summarization results on 2009009 question.

Summary type	1-gram	2-gram	SU2-gram	FRESA Average
Baseline summary:	24.24380	32.77566	32.77625	29.93190
Empty baseline:	35.38031	44.44000	44.55194	41.45742
Random unigram:	23.14971	32.24334	32.17335	29.18880
Random 5-gram:	20.80877	29.14371	29.31407	26.42218
Enertex summary:	27.06117	35.78855	35.84767	32.89913

[3] FRESA package is disponible at web site: `http://lia.univ-avignon.fr/fileadmin/axes/TALNE/downloads/index_fresa.html`

4 Conclusions

We have presented the Enertex summarization system on INEX-2010 Question-Answering task. In the INEX-2010 corpus, Enertex system obtained good results. However, by technical limitations (memory, CPU time), we used only the first five documents retrieved by Indri. We think that, a bigger number of documents may outperform the present results.

References

1. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *National Academy of Sciences*, 9:2554–2558, 1982.
2. J. Hertz, A. Krogh, and G. Palmer. *Introduction to the theorie of Neural Computation.* Addison Wesley, Redwood City, CA, 1991.
3. G. Salton and M.J. McGill. *Introduction to modern information retrieval.* Computer Science Series McGraw Hill Publishing Company, 1983.
4. Silvia Fernandez, Eric SanJuan, and Juan-Manuel Torres-Moreno. Energie textuelle des mémoires associatives. In *TALN'07*, volume 1, pages 25–34, 2007.
5. Silvia Fernandez, Eric SanJuan, and Juan-Manuel Torres-Moreno. Textual Energy of Associative Memories: performants applications of Enertex algorithm in text summarization and topic segmentation. In *MICAI'07*, pages 861–871, 2007.
6. J.M. Torres-Moreno, P. Velazquez-Moralez, and J. Meunier. *CORTEX, un algorithme pour la condensation automatique de textes.* In *ARCo*, volume 2, page 365, 2005.
7. J.M. Torres-Moreno, P. Velazquez-Morales, and J.G. Meunier. *Condensés de textes par des méthodes numériques. JADT*, 2:723–734, 2002.
8. Juan Manuel Torres-Moreno, Pier-Luc St-Onge, Michel Gagnon, Marc El-Bèze, and Patrice Bellot. Automatic summarization system coupled with a question-answering system (qaas). *in CoRR*, abs/0905.2990, 2009.
9. J.-M. Torres-Moreno, H. Saggion, I. da Cunha, P. Velázquez-Morales, and E. SanJuan. Evaluation automatique de rsums avec et sans rfrence. In *17e Confrence sur le Traitement Automatique des Langues Naturelles (TALN)*, 2010.
10. J-M. Torres-Moreno, H. Saggion, I. da Cunha, E. SanJuan, and P. Velázquez-Morales. Summary evaluation with and without references. *Polibitis: Research journal on Computer science and computer engineering with applications*, 42:13–19, 2010.
11. H. Saggion, J-M. Torres-Moreno, I. da Cunha, E. SanJuan, and P. Velázquez-Morales. Multilingual summarization evaluation without human models. In *23rd International Conference on Computational Linguistics (COLING 2010), Pekin*, 2010.

The REG summarization system at QA@INEX track 2010

Jorge Vivaldi[1], Iria da Cunha[1] and Javier Ramírez[2]

[1] Instituto Universitario de Linguistica Aplicada - UPF
Barcelona
[2] Universidad Autonoma Metropolitana-Azcapotzalco
Mexico
{iria.dacunha,jorge.vivaldi}@upf.edu;jararo@correo.azc.uam.mx
http://www.iula.upf.edu

Abstract. In this paper we present REG, a graph approach to study a fundamental problem of Natural Language Processing: the automatic summarization of documents. The algorithm models a document as a graph, to obtain weighted sentences. We applied this approach to the INEX@QA 2010 task (question-answering). To do it, we have extracted the terms from the queries, in order to obtain a list of terms related with the main topic of the question. Using this strategy, REG obtained good results with the automatic evaluation system FRESA.

Key words: INEX, Automatic Summarization System, Question-Answering System, REG.

1 Introduction

Nowadays automatic summarization is a very prominent research topic. We can define summary as “a condensed version of a source document having a recognizable genre and a very specific purpose: to give the reader an exact and concise idea of the contents of the source” (Saggion and Lapalme, 2002: 497). Summaries can be divided into “extracts”, if they contain the most important sentences extracted from the original text (ex. Edmunson, 1969; Nanba and Okumura, 2000; Gaizauskas et al., 2001; Lal and Reger, 2002; Torres-Moreno et al., 2002) and “abstracts”, if these sentences are re-written or paraphrased, generating a new text (ex. Ono et al., 1994; Paice, 1990; Radev, 1999). Most of the automatic summarization systems are extractive. These systems are useful in several domains: medical (ex. Johnson et al., 2002 Afantenos et al., 2005; da Cunha et al., 2007; Vivaldi et al., 2010), legal (ex. Farzindar et al., 2004), journalistic (ex. Abracos and Lopes, 1997; Fuentes et al., 2004), etc. One of the tasks where these extractive summarization systems could help is question-answering. The objective of the INEX@QA 2010 track is to evaluate a difficult question-answering task, where questions are very precise (expecting short answers) or very complex (expecting long answers, including several sentences). In this second task is where automatic summarization systems could help. The used corpus in this track contains all the texts included into the English Wikipedia. The expected answers are automatic summaries of less than 500 words exclusively made of aggregated passages extracted from the Wikipedia corpus. The evaluation of the answers will be automatic, using the automatic evaluation system FRESA (Torres-Moreno et al., 2010a, 2010b, Saggion et al., 2010), and manual (evaluating syntactic incoherence, unsolved anaphora, redundancy, etc.). To carry out this task, we have decided to use REG (Torres-Moreno and Ramirez, 2010; Torres-Moreno et al., 2010), an automatic summarization system based on graphs. We have performed some expansions of the official INEX@QA 2010 queries, detecting the terms they contain automatically, in order to obtain a list of terms related with the main topic of all the questions.

This paper is organized as follows. In Section 2 we show REG, the summarization system we have used for our experiments. In Section 3 we explain how we have carried out the terms extraction of the queries. In Section 4 we present the experimental settings and results. Finally, in Section 5, we expose some conclusions.

2 The REG system

REG (Torres-Moreno and Ramirez, 2010; Torres-Moreno et al. 2010) is an Enhanced Graph summarizer (REG) for extract summarization, using a graph approach. The strategy of this system has two main stages: a) to carry out an adequate representation of the document and b) to give a weight to each sentence of the document. In the first stage, the system makes a vectorial representation of the document. In the second stage, the system uses a greedy optimization algorithm. The summary generation is done with the concatenation of the most relevant sentences (previously scored in the optimization stage).

REG algorithm contains three modules. The first one carries out the vectorial transformation of the text with filtering, lemmatization/stemming and normalization processes. The second one applies the greedy algorithm and calculates the adjacency matrix. We obtain the score of the sentences directly from the algorithm. Therefore, sentences with more score will be selected as the most relevant. Finally, the third module generates the summary, selecting and concatenating the relevant sentences. The first and second modules use CORTEX (Torres-Moreno et al., 2002), a system that carries out an unsupervised extraction of the relevant sentences of a document using several numerical measures and a decision algorithm.

3 Terms extraction

The first procedure for obtaining the query terms has been to found the main topic of the questions. This has been obtained by finding the terms candidate present in every query. Terms are usually defined as lexical units to designate concepts in a domain. The detection of these units is a complex task mainly because terms adopt all word formation rules in a given language [22]. Also, as mentioned in the term definition itself, it is necessary to confirm that a given lexical unit belong to the domain of interest. Due to the difficulties to verify this condition it is usual to refer the results obtained by an extractor as term candidates instead of just "terms". In this context we have used the basic procedure for obtaining term candidates in the field of term extraction. Such candidates are typically obtained by using the morphosyntactic terminological patterns for any given language (see [23, 24]), English in this case.

As the queries do not belong to any specific domain it is not possible determine the termhood of the retrieved candidates.

Considering that questions are very short, only a few candidates are obtained by such procedure; therefore, they have a high probability to be the main topic of the question.

For example, for the query "How does GLSL unify vertex and fragment processing in a single instruction set?", we consider the terms "glsl", "vertex processing", "fragment processing" and "single instruction set". But for the query "Who is Eiffel?", there are not any term, only the proper name "Eiffel?".

4 Experiments Settings and Results

In this study, we used the document sets made available during the Initiative for the Evaluation of XML retrieval (INEX) 2010[1], in particular on the INEX 2010 QA Track (QA@INEX). These sets of documents where provided by the search engine Indri.[2] REG has produced multidocument summaries using sets of 30, 40 and 50 of the documents provided by Indri using all the queries of the track.

To evaluate the efficiency of REG over the INEX@QA corpus, we have used the FRESA package.

Table 1 shows an example of the results obtained by REG using 50 documents as input. The query that the summary should answer in this case was the number 2009006. This table presents REG results in comparison with an intelligent baseline (Baseline summary), and two simple baselines, that is, summaries including random n-grams (Random unigram) and 5-grams (Random 5-gram). We observe that our system is always better than these two simple baselines, but in comparison with the first one the performance is variable.

5 Conclusions

We have presented the REG summarization system, an extractive summarization algorithm that models a document as a graph, to obtain weighted sentences. We applied this approach to the INEX@QA 2010 task, extracting the terms from

[1] `http://www.inex.otago.ac.nz/`

[2] Indri is a search engine from the Lemur project, a cooperative work between the University of Massachusetts and Carnegie Mellon University in order to build language modelling information retrieval tools: `http://www.lemurproject.org/indri/`

Table 1. Example of REG results using 50 documents as input.

Distribution type	unigram	bigram	with 2-gap	Average
Baseline summary	22.64989	31.70850	32.07926	28.81255
Random unigram	18.18043	28.25213	28.44528	24.95928
Random 5-gram	17.47178	26.33253	27.03882	23.61437
Submitted summary	22.77755	32.06325	32.53706	29.12595

the queries, in order to obtain a list of terms related with the main topic of the question.

Our experiments have shown that the system is always better than the two simple baselines, but in comparison with the first one the performance is variable. We think this is due to the fact that some queries are long and they have several terms we could extract, but there are some queries that are very short and the term extraction is not possible or very limited. Nevertheless, we consider that, over the INEX-2010 corpus, REG obtained good results in the automatic evaluations, but now it is necessary to wait for the human evaluation and the evaluation of other systems to compare with.

References

1. Abracos, J.; Lopes, G. (1997). *Statistical methods for retrieving most significant paragraphs in newspaper articles.* In Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization. Madrid. 51-57.
2. Afantenos, S.; Karkaletsis, V.; Stamatopoulos, P. (2005). *Summarization of medical documents: A survey.* Artificial Intelligence in Medicine 33 (2). 157-177.
3. da Cunha, I.; Wanner, L.; Cabré, M.T. (2007). *Summarization of specialized discourse: The case of medical articles in Spanish.* Terminology 13 (2). 249-286.
4. Edmunson, H. P. (1969). *New Methods in Automatic Extraction.* Journal of the Association for Computing Machinery 16. 264-285.
5. Farzindar, A.; Lapalme, G.; Desclés, J.-P. (2004). *Résumé de textes juridiques par identification de leur structure thématique.* Traitement automatique des langues 45 (1). 39-64.
6. Fuentes, M.; Gonzalez, E.; Rodriguez, H. (2004). *Resumidor de noticies en catala del projecte Hermes.* In Proceedings of II Congrés d'Enginyeria en Llengua Catalana (CELC'04). Andorra. 102-102.
7. Gaizauskas, R.; Herring, P.; Oakes, M.; Beaulieu, M.; Willett, P.; Fowkes, H.; Jonsson, A. (2001). *Intelligent access to text: Integrating information extraction technology into text browsers.* En Proceedings of the Human Language Technology Conference. San Diego. 189-193.
8. Johnson, D.B.; Zou, Q.; Dionisio, J.D.; Liu, V, Z.; Chu, W.W. (2002). *Modeling medical content for automated summarization.* Annals of the New York Academy of Sciences 980. 247-258.
9. Lal, P.; Reger, S. (2002). *Extract-based Summarization with Simplication.* In Proceedings of the 2nd Document Understanding Conference at the 40th Meeting of the Association for Computational Linguistics. 90-96.

10. Nanba, H.; Okumura, M. (2000). *Producing More Readable Extracts by Revising Them*. In Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000). Saarbrucken. 1071-1075.
11. Ono, K.; Sumita, K.; Miike, S. (1994). *Abstract generation based on rhetorical structure extraction*. In Proceedings of the International Conference on Computational Linguistics. Kyoto. 344-348.
12. Paice, C. D. (1990). *Constructing literature abstracts by computer: Techniques and prospects*. Information Processing and Management 26. 171-186.
13. Radev, D. (1999). Language Reuse and Regeneration: Generating Natural Language Summaries from Multiple On-Line Sources. New York, Columbia University. [PhD Thesis]
14. Saggion, H.; Lapalme, G. (2002). *Generating Indicative-Informative Summaries with SumUM*. Computational Linguistics 28 (4). 497-526.
15. Torres-Moreno, J-M.; Saggion, H. da Cunha, I. SanJuan, E. Velázquez-Morales, P. SanJuan, E.(2010a). *Summary Evaluation With and Without References*. Polibitis: Research journal on Computer science and computer engineering with applications 42.
16. Saggion, H.; Torres-Moreno, J-M.; da Cunha, I.; SanJuan, E.; Velázquez-Morales, P.; SanJuan, E. (2010b). *Multilingual Summarization Evaluation without Human Models*. In Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010). Pekin.
17. Torres-Moreno, J.-M.; Saggion, H.; da Cunha, I.; Velázquez-Morales, P.; SanJuan, E. (2010b). *Ealuation automatique de résumés avec et sans référence*. In Proceedings of the 17e Conférence sur le Traitement Automatique des Langues Naturelles (TALN). Université de Montréal et Ecole Polytechnique de Montréal: Montreal (Canada).
18. Torres-Moreno, J-M.; Ramírez, J. (2010). *REG : un algorithme glouton appliqué au résumé automatique de texte*. JADT 2010. Roma, Italia.
19. Torres-Moreno, J-M.; Ramírez, J.; da Cunha, I. (2010). *Un resumeur a base de graphes, indépendant de la langue*. Workshop African HLT 2010. Djibouti.
20. Torres-Moreno, J. M.; Velázquez-Morales, P.; Meunier, J. G. (2002). *Condensés de textes par des méthodes numériques*. En Proceedings of the 6th International Conference on the Statistical Analysis of Textual Data (JADT). St. Malo. 723-734.
21. Vivaldi, J.; da Cunha, I.; Torres-Moreno, J.M.; Velázquez, P. (2010). "Automatic Summarization Using Terminological and Semantic Resources". En actas del 7th International Conference on Language Resources and Evaluation (LREC 2010). Valletta, Malta.
22. Pearson J. (1998). Terms in context. John Benjamin. Amsterdam.
23. Cabré, M.T.; R. Estopà; Vivaldi, J. (2001). *Automatic term detection: a review of current systems*. In Bourigault, D., C. Jacquemin and M.C. L'Homme (eds.). Recent Advances in Computational Terminology. 53-87. Amsterdam: John Benjamins.
24. Pazienza, M.T.; Pennacchiotti, M.; Zanzotto, F.M. (2005). *Terminology Extraction: An Analysis of Linguistic and Statistical Approaches*. In: Studies in Fuzziness and Soft Computing. Volume 185/2005. 255-279.

Focused Relevance Feedback @ INEX 2010

Shlomo Geva
Computer Science, QUT
2 George St
Brisbane Q4001 Australia
+617 3138 1920
s.geva@qut.edu.au

Timothy Chappell
Computer Science, QUT
2 George St
Brisbane Q4001 Australia
+617 3138 1920
timothy.chappell@qut.edu.au

ABSTRACT

The INEX 2010 Focused Relevance Feedback track offered a refined approach to the evaluation of Focused Relevance Feedback algorithms through simulated exhaustive user feedback. As in traditional approaches we simulated a user-in-the loop by re-using the assessments of ad-hoc retrieval obtained from real users who assess *focused* ad-hoc retrieval submissions. The evaluation was extended in several ways: the use of *exhaustive* relevance feedback over entire runs; the evaluation of *focused retrieval* where both the retrieval results and the feedback are *focused*; the evaluation was performed over a closed set of documents and complete focused assessments; the evaluation was performed over executable implementations of relevance feedback algorithms; and finally, the entire evaluation platform is reusable. We present the evaluation methodology, its implementation, and experimental results obtained for 9 submissions from 3 participating organizations.

Categories and Subject Descriptors

Focused Relevance Feedback, Relevance Feedback, Information Retrieval, IR, RF, User Simulation, Search Engine, Evaluation, INEX http://www.inex.otago.ac.nz/

General Terms

Algorithms, Measurement, Performance, Benchmark.

Keywords

Relevance feedback evaluation.

1. INTRODUCTION

Information retrieval systems are most effective when used by skilled operators who are capable of forming queries appropriate for retrieving relevant documents. The vast majority of users of information retrieval systems are unlikely to be skilled users. It is a trivial observation that user will sooner reformulate a query than they would scan the initial result list to any depth beyond the first page of results. As query reformulation may be a difficult and time-consuming task, machine-assisted query reformulation based on the requirements of the user is an important part of information retrieval. An early and rather effective mechanism for improving the effectiveness of search interfaces is known as *relevance feedback* where by query reformulation is automated. We are concerned with the *evaluation* of this approach. This paper describes an extension of the *Incremental Relevance Feedback* approach, described by IJsbrand Jan Aalbersberg[1], to Focused Relevance Feedback and to the evaluation of executable implementations under uniform setting.

1.1 Relevance Feedback Evaluation

This wealth of research and reported results on relevance feedback leads to an obvious problem – most of the earlier work is difficult to reproduce reliably, and certainly not without great difficulty in implementation of systems described by others. Of great importance to the task of comparing different methods of ranking and retrieval is having a standard, systematic way of evaluating the results so that it can be empirically validated, in a methodologically sound manner, that for a given test collection one particular method is better than another.

Ruthven and Lalmas[2] review alternate evaluation methods suited to relevance feedback: *Freezing, Residual ranking*, and *Test and control,* all intended to counter the effect where the documents marked as 'relevant' by the user are pushed to the top of the document ranking, artificially raising the mean precision of the results. *Freezing* is where the initially top-ranked documents are frozen in place and the relevance feedback system used to re-rank the remaining documents, and the precision/recall evaluation conducted on the entire document set. *Residual ranking* is where the top-ranked documents, used to train the relevance feedback system, are removed from the document set before evaluation. *Test and control groups*; where the document set is partitioned into two equal groups, the first used to train the relevance feedback system and the second used to evaluate the system. Aalbersberg[1] describes an incremental approach which we extend in this paper, where one document at a time is evaluated by the user until a given depth of results list is inspected.

1.2 Focused Relevance Feedback evaluation

In the INEX 2010 we adopted a refined approach to the evaluation of Relevance Feedback algorithms through simulated *exhaustive* incremental user feedback. The approach extends evaluation in several ways, relative to traditional evaluation. First, it facilitates the evaluation of retrieval where both the retrieval results and the feedback are ***focused***. This means that both the search results and the feedback are specified as passages, or as XML elements, in documents - rather than as whole documents. Second, the evaluation is performed over a closed set of documents and assessments, and hence the evaluation is exhaustive, reliable and less dependent on the specific search engine in use. By reusing the relatively small topic assessment pools, having only several hundred documents per topic, the search engine quality can largely be taken out of the equation. Third, the evaluation is performed over ***executable*** implementations of relevance feedback algorithms rather than being performed over result submissions. Finally, the entire evaluation platform is reusable and over time can be used to measure progress in focused relevance feedback in an independent, reproducible, verifiable, uniform, and methodologically sound manner.

2. EVALUATION APPROACH

This approach is concerned with the simulation of a user in loop, in the evaluation of relevance feedback systems. This approach can be used to compare systems in an evaluation forum setting, or simply to evaluate improvements of variations to existing relevance feedback algorithms in the development process.

2.1 Use Case

The use-case of this track is similar to Aalbersberg[1] - a single user searching with a particular query in an information retrieval system that supports relevance feedback. Our user views and ***highlights relevant passages*** of text in a returned document (if exist) and provides this feedback to the information retrieval system. The IR system re-ranks the remainder of the unseen results list to provide the next ***assumed*** most relevant result to the user. The exact manner in which this is implemented is not of concern in this evaluation; here we test the ability of the system to use focused relevance feedback to improve the ranking of previously unseen results. Importantly, we extend Aalbersberg's approach to compare the improvement, if exists, which focused relevance feedback (FRF) offers over whole document feedback. This includes structured IR (e.g. XML documents).

2.2 Test Collection

The relevance feedback track re-used the INEX Wikipedia XML collection. Evaluation was based on the focused relevance assessments, which are gathered by the INEX Ad-Hoc track through the GPXrai assessment tool, where assessors highlight relevant passages in documents. The INEX Wikipedia test collection is semantically marked up. This facilitates the evaluation of FRF algorithms implementations, which take advantage not only of the (often) passage-sized feedback, but also the semantic mark-up of the relevant text.

2.3 Task

Participants were asked to create one or more Relevance Feedback Modules (RFMs) intended to rank a collection of documents with a query while incrementally responding to explicit user feedback on the relevance of the results presented to the user. These RFMs were implemented as dynamically linkable modules that implement a standard defined interface. The Evaluation Platform (EP) interacts with the RFMs directly, simulating a user search session. The EP instantiates an RFM object and provide it with a set of XML documents and a query. The RFM responds by ranking the documents (without feedback) and returning the ranking to the EP. This is so that the difference in quality between the rankings before and after feedback can be compared to determine the extent of the effect the relevance feedback has on the results. The EP is then asked for the next most relevant document in the collection (that has not yet been presented to the user). On subsequent calls the EP passes relevance feedback (in the form of passage offsets and lengths) about the last document presented by the RFM. This feedback is taken from the qrels of the respective topic, as provided by the Ad-Hoc track assessors. The simulated user feedback may then be used by the RFM to re-rank the remaining unseen documents and return the next most relevant document. The EP makes repeated calls to the RFM until all relevant documents in the collection have been returned.

The EP retains the presentation order of documents as generated by the RFM. This order can then be evaluated as a submission to the ad-hoc track in the usual manner and with the standard retrieval evaluation metrics. It is expected that an effective dynamic relevance feedback method will produce a higher score than a static ranking method (i.e. the initial baseline rank ordering). Evaluation is performed over all topics and systems are ranked by the averaged performance over the entire set of topics, using standard INEX and TREC metrics.

Each topic consists of a set of documents (the topic pool) and a complete and exhaustive set of manual focused assessments against a query. Hence, we effectively have a "classical" Cranfield experiment over each topic pool as a small collection with complete assessments for a single query. The small collection size allows participants without an efficient implementation of a search engine to handle the task without the complexities of scale that the full collection presents.

Participants in the track were provided with a model solution as a baseline. Figure 1 depicts the performance improvement evaluation as obtained by using Rocchio with the Lucene search engine, when evaluated by trec_eval. Rocchio-based relevance feedback engine results in an improved mean average precision. The third line shown (the middle) is the best performing submission at INEX 2008, modified to conform to the trec_eval input format. It performs best out of the three in early precision, but precision suffers later and it has a lower average precision than the Lucene engine when using relevance feedback.

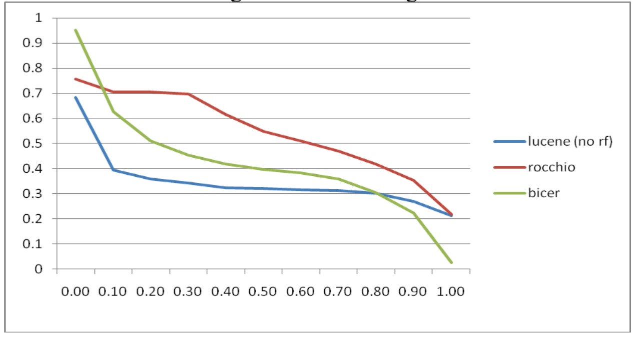


Figure 1. Evaluation with trec_eval, document retrieval.

The approach provides an interactive user session simulation in a focused relevance feedback setting. The evaluation provides a level playing field for the ***independent and completely reproducible*** evaluation of RF implementations in a standard setting and with a standard pool of documents for each topic.
The approach supports the accurate evaluation of any benefits that may (or may not) arise from the use of Focused IR, as opposed to document IR, be it passage based or XML Element based.

3. Results

In 2010 there were 3 participating organisations: Indian Statistical Institute, Peking University, and QUT. There were 9 submissions in total and the baseline submission was provided by the Lucene search engine with default parameters out of the box.

Figures 2 and 3 depict preliminary outcomes. Of course all FRF submissions were collected before the INEX 2010 assessments were made available and so this test is not biased by training. Further analysis and details will be provided in the final proceedings paper.

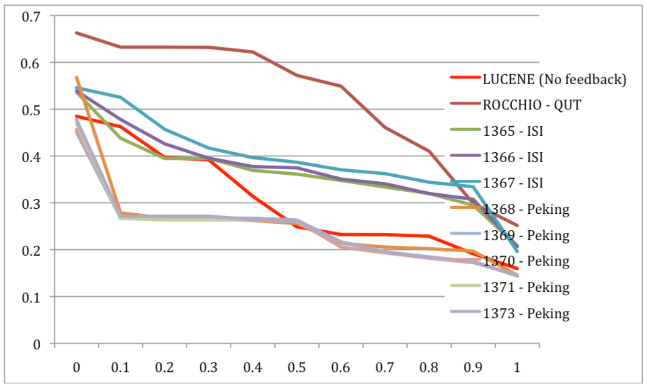


Figure 2. INEX 2010 topics evaluated with trec_eval, document retrieval

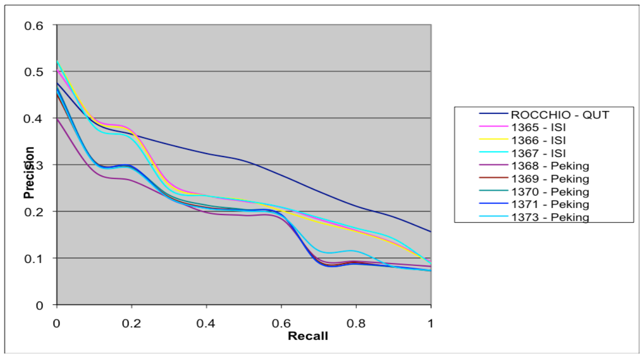


Figure 3. INEX 2010 topics evaluated with inex_eval, best in context metric

4. References

1. IJsbrand Jan Aalbersberg, Incremental Relevance Feedback, Proceedings of SIGIR 1992, pp 11-22.
2. I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.,* 18(2):95-145, 2003

DCU and ISI@INEX 2010: Adhoc Data-Centric and Feedback tracks

Debasis Ganguly[1], Johannes Leveling[1], Gareth J. F. Jones[1], , and Sauparna PalChowdhury[2]

[1]CNGL, School of Computing, Dublin City University, Dublin, Ireland
[2]CVPR Unit, Indian Statistical Institute, Kolkata, India
{dganguly, jleveling, gjones}@computing.dcu.ie

Abstract. We describe the participation of Dublin City University (DCU) and the Indian Statistical Institute (ISI) in INEX 2010. We propose a Hieararchical Language Model (HLM) implemented in SMART for the adhoc task to retrieve the XML elements where an XML element is scored against the combined probability of generating the given query from itself and its parent article. We submitted three element level runs for each Focussed and Restricted subtask. For the newly introduced "Data-Centric" track we submitted 10 runs. exploring the effects of Blind Relevance Feedback (BRF) using query expansion terms and Expectation Maximization (EM). For the newly introduced Relevance Feedback track, we submitted an implementation which is based on extracting the most similar non-overlapping fixed length word windows from relevant passages and then re-retrieving with a modified query which additionally contains the most frequent terms extracted from these word windows.

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 XML elements. A retrieved passage can be a sequence of textual content either from within an element or spanning a range of elements. INEX-2010 saw the introduction of the restricted versions of the "Focused" and the "Relevant In Context" tasks which have been designed particularly for displaying results on a mobile device with limited graphics resources. The tasks for the Adhoc track are outlined as follows: a) the "Restricted Focused" task which asks systems to return a ranked list of elements or passages to the user; b) the (un)restricted "Relevant in Context" tasks which asks systems to return relevant elements or passages grouped by article, a limit of atmost 500 characters being imposed on the restricted version; and c) the "Efficiency" task which expects systems to return thorough article level runs.

The participants were free to use either of the two query variants: Content-Only (CO) and Content-And-Structure (CAS) queries. In the CO variant, the user poses the query in free text (resembling web search queries) 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.

We also participated in two new tracks introduced in 2010: a) the "Data Centric track" which is similar to adhoc retrieval of elements or passages on a domain specific collection of IMDB movie pages; b) the "Feedback track" which attempts to simulate user-interaction by the communication of true relevance information between a *Controller module*, which has access to the *qrels* file, and a user implemented *Feedback module*.

In INEX-2010 we submitted 9 adhoc focused runs, (3 for each Focussed task) and 3 Thorough runs for the Adhoc track. In addition we submitted 10 runs for the Data Centric task and 3 relevance feedback modules (as .jar files) for the Feedback track. Retrievals for the adhoc and the Data Centric tasks were done by using the SMART[1] system whereas the Java classes used for the Feedback track use the Lucene API[2] for indexing and retrieval.

The remainder of the paper is organized as follows: Section 2 elaborates on the approaches to indexing and retrieval of whole documents followed by Section 3 which describes the strategy for measuring the similarities of the individual XML elements to the query. In Sections 4, 5, 6 we give details of our approaches for our participation in the various tracks. Section 7 reports the official results and Section 8 concludes the paper with directions for future work.

2 Document Retrieval

This Section describes the approaches undertaken towards whole document retrieval.

2.1 Preprocessing

Similar to INEX-2009, for extracting useful parts of documents, we shortlisted about thirty tags that contain useful information: <*p*>, <*p1*>, <*st*>, <*section*> etc. [6]. 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]. Words listed in the standard stop-word list included within SMART were removed from both documents

[1] `ftp://ftp.cs.cornell.edu/pub/smart/`
[2] `http://www.apache.org/dyn/closer.cgi/lucene/java/`

and queries. Words were stemmed using the default stemmer implementation of SMART which is a variation of the Lovin's stemmer. Frequently occurring word bi-grams (loosely referred to as phrases) were also used as indexing units. We used the N-gram Statistics Package (NSP)[3] on the English Wikipedia text corpus and selected the 100,000 most frequent word bi-grams as the list of candidate phrases.

2.2 Language Model (LM) Term weighting

Our retrieval method was based on the Language Modeling approach proposed by Hiemstra [4]. In this Subsection we summarize the Language Modeling method to IR used by us for document retrieval. In LM based IR, a document d is ranked by the estimated probability $P(q|d)$ of generating a query q from the document model underlying the document d. The document is modelled to choose $q = \{t_1, t_2 \ldots t_n\}$ as a sequence of independent words as proposed by Hiemstra [4].

$$P(q|d) = P(d) \log P(d) \prod_{i=1}^{n} \lambda_i P(t_i|d) + (1 - \lambda_i) P(t_i) \tag{1}$$

The term weighting equation can be derived from Equation 1 by dividing it with $(1 - \lambda_i)P(t_i)$ and taking log on both sides to convert the product to summation.

$$\log P(q|d) = \log P(d) + \sum_{i=1}^{n} \log(1 + \frac{\lambda_i}{1 - \lambda_i} \frac{P(t_i|d)}{P(t_i)}) \tag{2}$$

We used SMART to index each query vector $\boldsymbol{q}$ as $q_k = tf(t_k)$ and each document vector $\boldsymbol{d}$ as $d_k = log(1 + \frac{P(t_k|d)}{P(t_k)} \frac{\lambda_k}{1-\lambda_k})$, so that the dot product $\boldsymbol{d} \cdot \boldsymbol{q}$ gives the likelihood of generating $\boldsymbol{q}$ from $\boldsymbol{d}$ and hence can be used as the similarity score to rank the documents.

3 Element Retrieval

For the element-level retrieval, we adopted a 2-pass strategy. In the first pass, we retrieved 1500 documents (since the thorough runs for INEX are required to report atmost 1500 documents per topic) for each query using the LM retrieval method as described in the previous section 2.2, without using query expansion from BRF the reason being usage of BRF showed a decrease in the MAiP on the 2009 topics.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

[3] http://www.d.umn.edu/∼tpederse/nsp.html

retrieved elements. We adopted two strategies for element scoring: a) pivoted length normalization b) a generalization of LM with two levels of smoothing [a)] The following sections provide details of these methods. The preprocessing steps are similar to 2.1.

3.1 Pivoted Length Normalization

Earlier year, we revisited length normalization (see [5] [6] for more details) and rewritten it as:

$$normalization = 1 + \frac{slope}{(1 - slope)} * \frac{\#unique\ terms}{pivot}$$

where *length* is given by #unique terms in the document. Instead of varying both pivot and slope separately we considered the combined term $\frac{slope}{(1-slope)*pivot}$. For the sake of simplicity, we chose $pivot = 1$ which reduces the factor to $\frac{slope}{(1-slope)}$. We call it the *pivot-slope factor* and used the optimal value of 0.00073 as trained on 2009 topics.

3.2 Hierarchical Language Model

An extension of LM for Field Search was proposed by Hiemstra [4] which involves scoring a document according to the probability of generation of the query terms either from the document itself as a whole, or from a particular field of it (e.g title) or from the collection. Thus Equation 1 can be extended as

$$P(q|d) = \log P(d) \prod_{i=1}^{n} \mu_i P(t_i|d, f) + \lambda_i P(t_i|d) + (1 - \lambda_i - \mu_i) P(t_i) \quad (3)$$

where f is the field to search from and μ_i is the probability of the additional event of choosing the term only from field f. We use almost the same principle as in Equation 1 for scoring the individual elements of the Wikipedia articles. The only difference with Field Language Model (FLM) is that it uses the evidence from a constituent field to assign a score to the container document, whereas we assign a score to the constituent element itself from the parent article evidence. We use Equation 4 to score an element.

$$P(q|e) = \log P(d) \prod_{i=1}^{n} \mu_i P(t_i|e) + \lambda_i P(t_i|d) + (1 - \lambda_i - \mu_i) P(t_i) \quad (4)$$

The parameter λ_i denotes the probability of choosing t_i from the parent article of the element e, whereas μ_i denotes the probability of choosing t_i from the element text. The residual event involves choosing t_i from the collection. Thus even if a query term t_i is not present in the element some non zero probability of generation is contributed to the product. Two levels of this smoothing are employed in this case. The following interpretations can be drawn from Equation 4:

a) An element e_1 which has a query term t only in itself but not anywhere else in the top level article, would score lower than an element e_2 which has the term present both in itself and somewhere else in the article. Thus the model would favour elements with some pre-defined contextual information about the query terms over individual snippets of information which do not have any associated context.
b) An element without any or only a few of the given query terms might still be retrieved if the missing terms are abundant in the article. This is particularly helpful for assigning high scores to elements (sections or paragraphs) which densely discusses a single sub-topic (the sub-topic typically being one facet of the user information need). For example if the user issues a query "fish drift net" and he wants information on the environmental hazards of "drift netting" but only from documents which are about "fishing", this way of scoring the elements ensures that a section with abundance of the words "drift" and "net" have higher likelihood of retrieval if the parent article is abundant with the term "fish".

We call this method Hierarchical Language Model (HLM) since it can be generalized upto $d + 1$ levels of smoothing, i.e. one from each of the parent levels $0, \ldots d - 1$ (d being the depth of the XML element in the DOM tree) and one from the collection. For the time being to keep things simple, we restrict our choice of smoothing only to the root article element.

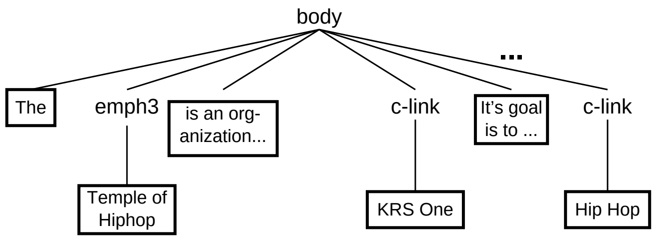


Fig. 1. Parse tree for a fragment of a Wikipedia document

4 Adhoc Track

4.1 Thorough task

We submitted three element level runs for each of the four tasks thus making 12 runs in total. Each element level run was converted to the *trec++* format reporting the FOLs instead of the element names using the conversion utility SUB2FOL.jar. All the initial article level runs were done using LM retrieved as described in Equation 1. We assign $\lambda_i = \lambda \quad \forall i = 1 \ldots n$ and also assigned uniform prior probabilities to the documents. To find the optimal value of λ

we varied its value in the range $[0.2, 0.5]$ in steps of 0.1 assuming that 0.1 is the smallest level of granularity of changes in the λ values for observing any significant differences in retrieval results. The results are reported in Table 1. We discovered an error that we used the file *gpxrairel.table.serialized* instead of *gpxrai50rel.table.serialized* for converting the element submission to FOL submission. Usage of the smaller version of the serialized FOLs (which only contains the mapping for the true relevant elements for 2009 topics) resulted in false boosting of retrieval effectiveness scores. In Table 1 we report both the evaluations - one with *gpxrairel.table.serialized* and the other with *gpxrai50rel.table.serialized*. Table 1 shows that we obtain the best retrieval results with λ set to 0.4.

Table 1. Effect of varying λ for INEX 2009 topics

λ	Relevant FOLs				All FOLs			
	iP[0.01]	iP[0.05]	iP[0.10]	MAiP	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
0.2	0.4762	0.4498	0.4249	0.2309	0.3503	0.3229	0.2735	0.1134
0.3	0.4887	0.4712	0.4403	0.2515	0.3690	0.3442	0.2949	0.1210
0.4	0.4996	0.4824	0.4513	**0.2572**	0.3720	0.3456	0.2991	**0.1228**
0.5	0.4858	0.4638	0.4471	0.2505	0.3635	0.3367	0.2938	0.1167

4.2 Focussed Task

We tried out two element level retrieval approaches as outlined in Section 3 on our generated best performing article level run. We also performed HLM element retrieval on the reference BM25 run provided by the INEX organizers. For the HLM experiments to simplify the model we used $\lambda_i = \lambda \wedge \mu_i = \mu \quad \forall i = 1 \ldots n$. A higher value of μ as compared to λ would attach too much importance on the presence of the query terms in the elements to get a higher likelihood of retrieval. While this might be good for queries with highly correlated terms, typically user queries are faceted, each term representing one such facet. It is highly unlikely that a single section or paragraph would cover all the facets. The more likely situation is that a small paragraph would cover one facet of the user's information need whereas the other facets are covered somewhere else in the document. A value of μ lower than λ ensures retrieval of elements highly focussed on one subtopic covering a single aspect of the user's information need from the parent article covering some other facets as well. Following the above line of argument, we chose $\lambda = 0.25$ and $\mu = 0.15$ for our element retrieval experiments.

A critical issue to explore in the model of Equation 4 is the issue of assigning prior probabilities to the elements. Singhal [7] analyzes the likelihood of relevance against the length of TREC documents and reports that longer documents have a higher probability of relevance. While this scheme of assigning document prior probabilities proportional to their lengths suits the traditional adhoc retrieval of documents (the retrieval units being whole documents) from the news genre, for a more flexible retrieval scenario such as the Restricted Focussed INEX task where retrieval units can be arbitrary passages and shorter passages are favoured over longer ones, it might be worth trying to assign prior probabilities to elements inversely proportional to their lengths. We performed HLM retrieval of XML elements on the best performing article level run (λ being set to 0.4 as reported in Table 1). As a baseline we chose to use standard LM scoring of the elements which is a special case of HLM obtained by setting $\lambda = 0$. To verify our hypothesis that λ should be higher than μ, we ran two versions of HLM one with $\lambda < \mu$ and μ, λ. Table 2 reports the measured retrieval effectiveness of the different cases and also shows the effect on precision for the three different modes of element priors - uniform, proportional and inversely proportional for the case $\mu < \lambda$. Table 2 provides empirical evidence to the hypothesis that an element

Table 2. HLM for element retrieval for INEX 2009 topics

λ	μ	Element Prior	Retrieval Effectiveness			
		probability	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
0.0	0.15	Uniform	0.2639	0.1863	0.1335	0.0448
0.15	0.25	Uniform	0.4082	0.2648	0.1894	0.0566
0.25	0.15	Uniform	**0.5256**	**0.3595**	0.2700	0.0991
0.25	0.15	Shorter favored	0.3459	0.1682	0.0901	0.0314
0.25	0.15	Longer favored	0.4424	0.3582	**0.2787**	**0.1064**

scored with some contextual information from the article performs better. The first row of the table reports the case where elements are LM weighted without any parent article information. It can be seen that the first row yields the least $iP[0.01]$ value. The table also justifies the hypothesis of assigning $\mu < \lambda$ since $iP[0.01]$ of the third and fifth rows are higher than that of the second row.

4.3 Relevant In Context

For the Relevant In Context task we undertook an approach of regrouping the returned elements by the parent article. We employed two methods for reranking the elements described as follows:

Group by the reading order In this ordering scheme the elements are grouped together by the rank of the top ranked element from an article. In this ordering scheme, an element e_i gets a lower rank (best is rank 1) as compared to e_j if the best ranked element from the article to which e_i belongs is retrieved at a lower rank when compared to the best ranked element from the article containing e_j. Thus,

$$rank(e_i) < rank(e_j) \Leftrightarrow rank(first(p(e_i))) < rank(first(p(e_j))$$

where $p(x)$ refers to the parent article of element x and $first(y)$ denotes the best ranked element of article y.

Group by the aggregated similarities In this ordering scheme, the articles are reranked by a weighted sum of the similarity values of the individual elements of the articles. Thus the elements of the article which has the maximum aggregated similarity values of its constituent elements are returned first and so on. An element e_i gets a lower rank as compared to e_j if the aggregated similarity values of the elements from the parent article of e_i is higher than those for e_j. Thus,

$$rank(e_i) < rank(e_j) \Leftrightarrow \sum_{e \in A_i} sim(e) < \sum_{e \in A_j} sim(e) \tag{5}$$

where A_i refers to the set of all elements of the parent article of e_i and $sim(x)$ is the similarity of element x. The elements which are further down the ranked list can be penalized by down-weighting their contributions. Thus, we used a modified version of Equation 5

$$rank(e_i) < rank(e_j) \Leftrightarrow \sum_{e \in A_i} \frac{N - rank(e)}{N} sim(e) < \sum_{e \in A_j} \frac{N - rank(e)}{N} sim(e) \tag{6}$$

In Equation 6 $rank(e)$ refers to the rank of an element e and downweights the similarities of elements at higher ranks.

We tried out these approaches on the HLM element retrievals involving uniform and longer element priors. The results are summarized in Table 3. Results are reported for both the unrestricted and restricted (to 500 characters) versions. The reported 500 characters for the restricted version often cover less than 5 documents as a result of which the fixed point cut-off metrics for 5 or higher number of documents e.g. $gP[5]$, $gP[10]$ etc. yield the same value. Hence for the restricted version, we report only the MAgP scores. Table 3 shows that

Table 3. Relevant In Context runs generated from HLM element retrievals for INEX 2009 topics

λ	μ	Element Prior	RIC	Unrestricted					Restricted
		probability	method	gP[5]	gP[10]	gP[25]	gP[50]	MAgP	MAgP
0.25	0.15	Uniform	ro	**0.2872**	**0.2182**	0.1547	0.1138	**0.0912**	**0.0231**
0.25	0.15	Uniform	aggr	0.1992	0.1613	0.1294	0.1095	0.0722	0.0114
0.25	0.15	Uniform	waggr	0.2516	0.1918	**0.1579**	**0.1222**	0.0869	0.0163
0.25	0.15	Longer favored	ro	0.2175	0.1752	0.1251	0.0880	0.0628	0.0309
0.25	0.15	Longer favored	aggr	0.1304	0.1124	0.0916	0.0707	0.0396	0.0110
0.25	0.15	Longer favored	waggr	0.1625	0.1491	0.1106	0.0842	0.0496	0.0129

weighted aggregation method of Equation 6 performs better than its unweighted counterpart. The HLM element retrieval runs without length bias of the elements do better. Although *waggr* performs worse as compared to *ro*, it yields higher fixed point cut-off precisions for 25 and 50 documents as can be seen from the third row.

5 Data Centric Track

6 Feedback Track

6.1 Motivation

A major problem in IR is the mismatch between query terms and terms in relevant documents in the collection which satisfy the user's information need. Query expansion (QE) is a popular technique used to bridge this vocabulary gap. Query expansion techniques work by adding terms to the user's original query so as to enrich it to more fully describe the information need either by including alternative terms which might have been used in the relevant documents or which augment the terms in the original query. If good expansion terms are selected then the retrieval system can fetch additional relevant documents or increase the retrieved rank of items already retrieved. The query expansion techniques aim to predict the most suitable candidate words to be added to the query so as to increase retrieval effectiveness.

One of the problems in BRF is that all terms which are not related to the query, but meet the selection criterion for feedback terms are used for QE (e.g. semantically unrelated, but high frequency terms from long pseudo-relevant documents). Using text passages for feedback (sentences instead of full documents) might be more successful because long documents can contain a wider range of discourse and noisy terms would be added to the original query, causing a topic shift. As a result of this wide range of discourse for long documents the relevant portion of such a document may be quite small and feedback terms should be extracted from relevant portions only. These problems have led to the idea of using smaller textual units (passages[4]) for query expansion, which dates back to the experiments on the TIPSTER collections [8, 9]. This approach raises questions of how to create the passages, how to select the relevant passages, how passage size influences performance, and how to extract feedback terms from the passages. This track provides the opportunity to explore how true relevant passages can be used for feedback.

Xu and Croft [10] proposed Local Context Analysis (LCA) which involves decomposing the feedback documents into fixed length word windows so as to overcome the problem of choosing terms from the unrelated portions of a long document and then ranking the terms by a scoring function which depends on the co-occurrence of a word with the query term, the co-occurrence being computed within the fixed word length windows.

[4] We employ the term passage in its most general sense, denoting phrases, sentences, paragraphs, and other small text units.

The motivation behind our method is the assumption that even large *true* relevant text might contain unuseful or harmful terms for query expansion. We assume that terms in close proximity to the query terms are good candidates for expansion. Thus, in our method, we restrict the choice of feedback terms from word windows maximally similar to the query, thus achieving the same effect of filtering out potentially irrelevant parts of a longer document as LCA. The difference with LCA is that we do not compute the co-occurrences explicitly nor do we use the idf scores.

6.2 Algorithm

The newly introduced feedback track provides opportunity to utilize true relevance feedback information as opposed to BRF paradigm of traditional IR. The incremental reporting of relevant portions from full documents allows development of a feedback algorithm with which to choose a variable number of terms directly or inversely proportional to the lengths of the relevant portions reported. Both the directly and inversely proportional approaches can be justified in their own ways. One might want to choose more terms from a smaller chunk of a relevant section in the hope that it has little or no noisy terms, whereas it might be worth to choosing more terms from a larger chunk of relevant section on the assumption that longer the relevant section higher is the greater likelihood of finding useful expansion terms. For the feedback track, we propose the following basic algorithm and its variations achieved by how t_i is chosen in Step 6 of the algorithm.

1. For the i^{th} request of the next document to return, repeat Steps 2-7.
2. Let R be the accumulated string of relevant passages from the last document returned.
3. Break up R into fixed length windows of m words after applying stopword removal and stemming.
4. For each window $\boldsymbol{w} = (w_1, \ldots w_n)$, where $w_i = tf(t_i)^{\frac{1}{2}} \log idf(t_i)$ ($tf(t_i)$ is the term frequency of the i^{th} term in the window and $idf(t_i)$ is the inverse document frequency of t_i), compute the cosine-similarity of $\boldsymbol{w}$ with $\boldsymbol{q} = (q_1, \ldots q_n)$, where $q_i = tf(t_i)$.
5. Choose a subset of p windows having the top similarity values.
6. Extract out the most frequent t_i terms from these windows and add them to the query.
7. Re-retrieve with the expanded query and return the topmost yet unreturned document of the new ranked list.

The three variants for choosing t_i in Step 6 are as follows:

1. $t_i = t$, where t is a constant $\forall i$.
2. $t_i = \frac{(L_i - r_i)}{L_i} t$, where t is a constant, L_i is the length of the i^{th} document and r_i is the length of the relevant section of the i^{th} document.
3. $t_i = \frac{r_i}{L_i} t$, with t, L_i and r_i defined as before.

The first variant chooses a constant number of terms regardless of the segment length, whereas the second variant is used to choose a greater number of terms from shorter relevant segments, and the third variant is used to choose a smaller number of terms from shorter segments.

6.3 Training the system

The parameters as outlined in the feedback algorithm are the window length m, the number p of most similar windows to restrict the expansion terms to, and the constant t which guides the choice of the number of feedback terms. After performing a range of experiments to choose the optimal settings, the following values of the parameters were found to work well in practise: $m = 30$, $p = 10$, and $t = 5$. The results of the experiments with the above settings are outlined in Table 4. RF$_{const}$ is the relevance feedback run with a constant number of terms added from each marked relevant section, whereas RF$_{invrsl}$ and RF$_{rsl}$ use number of terms inversely and directly proportional to the length of the relevant section respectively. As a baseline we use standard Rochhio's feedback which was packaged as a default feedback module implementation by the INEX organizers. The default Rochhio's feedback implementation uses $\alpha = 1$, $\beta = 0.75$ and $\gamma = 0$ and uses 20 terms for query expansion. A notable difference between the baseline module and our implementation is that the former adds expansion terms to the original query at each iteration, whereas in case of the later for the i^{th} iteration we add expansion terms to the expanded query obtained during the $(i-1)^{th}$ iteration. Thus our query expansion is cumulative in nature in contrast to the baseline method.

Table 4. Best MAPs obtained by the three term selection methods for QE.

Topics	RF$_{Rochhio}$	RF$_{const}$	RF$_{invrsl}$	RF$_{rsl}$
Training set	0.4356	0.4892	**0.4939**	0.4838

Figure 2 shows the interpolated precsion-recall curves for the four approaches as reported in Table 4. The graph reveals some interesting characteristics of the two feedback methods RF_{rsl} and RF_{invrsl}. While it can be seen that RF_{rsl} yields low precision for lower levels of recall, it outperforms RF_{invrsl} for higher levels of recall which suggests that it might be worthy to try out a combination of the above two as a part of our future work.

7 Results

In this section we report the official results as reported in the INEX website.

Table 5. Official evaluation of the thorough runs

Run Id	# docs retrieved	MAiP
ISI2010_thorough.1500	1500	0.0846
ISI2010_thorough.150	150	0.0826
ISI2010_thorough.15	15	0.0714
12P167 (Best run)		**0.2354**

Table 6. Official evaluation of the Focussed runs

Run Id	Methodology	char precision
ISI2010_rfcs_ref	HLM element retrieval on article level reference run	0.2451
ISI2010_rfcs_flm	HLM element retrieval on article level LM run	0.2151
ISI2010_rfcs_vsm	Pivoted normalized element retrieval on article level LM run	0.1289
LIP6-OWPCparentFo (Best run)		**0.4125**

Table 7. Official evaluation of the Relevant In Context runs

Run Id	Methodology	Restricted	MAgP
ISI2010_ric_ro	Reading order	No	0.0645
ISI2010_ric_aggr	Weighted aggregation	No	0.0245
ISI2010_rric_ro	Reading order	Yes	0.0485
ISI2010_rric_aggr	Weighted aggregation	Yes	0.0482
ENSM-SE (Best run)		No	**0.1977**
32p167 (Best run)		Yes	**0.1580**

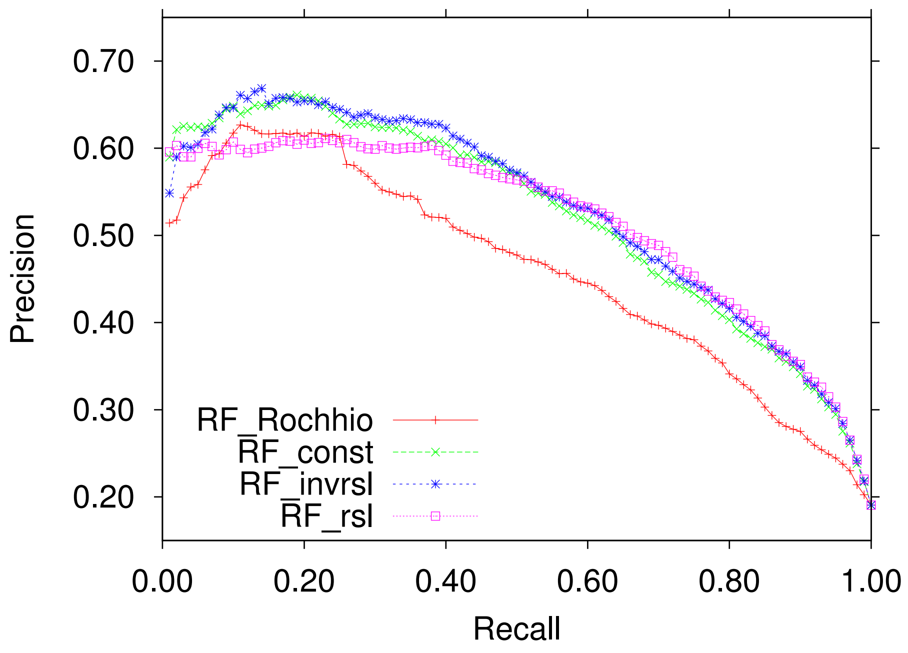


Fig. 2. Interpolated Precision-Recall curve

7.1 Adhoc task

Table 6 shows that the HLM based element retrieval on the reference run yields the highest character-precision among our runs. Table 7 shows that the reading order grouping by method works better than the weighted aggregation method.

8 Conclusion

Through our participation in INEX-2010, we wanted to explore an extension of LM for IR which we call HLM for element retrieval method. Trial experiments on INEX-2009 topics show that it outperforms the baseline LM element retrieval. Official restricted focussed runs show that HLM element retrieval outperforms the pivoted normalized VSM element retrieval. A new method for grouping retrieved elements by articles utilizing the weighted aggregated retrieval scores was investigated. But unfortunately it was not able to outperform the method of grouping by reading order on both INEX 2009 and 2010 topics. We proposed a new feedback method which restricts the choice of feedback terms to maximally similar pseudo-sentences extracted from reported relevant sections. The proposed method outperforms the baseline Rochhio feedback method on the training topics.

Acknowledgments

This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (CNGL) project.

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. Hiemstra, D.: Using language models for information retrieval. PhD thesis, University of Twente (2001)
5. 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
6. Pal, S., Mitra, M., Ganguly, D.: Parameter tuning in pivoted normalization for xml retrieval: Isi@inex09 adhoc focused task. In: INEX. (2009) 112–121
7. Singhal, A.: Term Weighting Revisited. PhD thesis, Cornell University (1996)
8. Callan, J.P.: Passage-level evidence in document retrieval. In: SIGIR 1994, ACM/Springer (1994) 302–310
9. Allan, J.: Relevance feedback with too much data. In: SIGIR 1995, ACM Press (1995) 337–343
10. Xu, J., Croft, W.B.: Query expansion using local and global document analysis. In: SIGIR 1996, ACM (1996) 4–11

Combining Strategy for XML Retrieval

Ning Gao, Zhi-Hong Deng, Jia-jian Jiang, Sheng-Long Lv, Yu Hang

Key Laboratory of Machine Perception (Ministry of Education)
School of Electronic Engineering and Computer Science, Peking University
nanacream@gmail.com;zhdeng@cis.pku.edu.cn;jjjoke@163.com;
davidfracs@gmail.com; pkucthh@gmail.com

Abstract. This paper describes Peking University's approaches to the Ad Hoc, Data Centric and Relevance Feedback track. In Ad Hoc track, results for four tasks were submitted: Efficiency, Restricted Focused, Relevance In Context and Restricted Relevance In Context. To evaluate the relevance between documents and a given query, multiple strategies, such as BM25, Two-Layer retrieval, MAXLCA semantic query model, distribution measurements and learn-to-optimize method are combined to form a more effective search engine. In Data Centric track, to get a set of "closely related" nodes that are "collectively relevant" to a given keyword query, we promoted three factors, correlativeness, explicitnesses and distinctiveness, to evaluate the importance of an attribute or a set of nodes from result documents. In Relevance Feedback track, our implementation employs two techniques, a revised Rocchio algorithm and criterion weight adjustment, to obtain useful information from feedbacks.

Keywords: INEX, Ad Hoc, Data Centric, Relevance Feedback

1 Introduction

INEX Ad Hoc Track [1] aims to evaluate performance in retrieving relevant results (e.g. XML elements or documents) to a certain query. In INEX Ad Hoc 2010, four different tasks are addressed: (1) Efficiency task requires a thorough run estimating the relevance of document components. (2) Relevant in Context task requires a ranked list of articles, and for each article a non-overlap ranked list results covering the relevant material in the article. (3) Restricted Focused task limits results (elements or passages) ranked in relevance order up to a maximal length of 1,000 characters per topic. (4) Restricted Relevant in Context tasks requires a ranked list of documents, and for each document a ranked list results covering its non-overlapping relevant material. Per result document can contain maximal 500 characters.

Initially our search engine is implemented to optimize the effectiveness of retrieval, regardless of the efficiency and restricted length. Five different querying strategies, BM25 [2], Two-Layer retrieval, Maximal Lowest Common Ancestor (MAXLCA) semantic query model, distribution measurements and learn-to-optimize method, are combined to form a more efficient search engine. Detailed definitions of these technologies will be introduced in section 2.The thorough retrieval results are submitted to efficiency task. Furthermore, the results for other three tasks are variants of these thorough runs. In Restricted Focused task, each topic limits the answers up to a maximal length of 1,000 characters. Hence, we only return the result elements that with top relevance-length ratio, in which relevance score for each results are computed by the aforementioned combined strategy module. For Relevance in Context task, the process scans the thorough runs and integrates the elements belong to the same document. The orders for these integrated covers in ranked list are determined by the elements with maximal relevance score in each set. To obtain the results of Restricted Relevance in Context task, each result in the Relevance in Context are pruned to maximal 500 characters. Similar to Restricted Focused task, only passages with top relevance-length ratio are reserved.

Data Centric track aims to provide a common forum for researchers or users to compare different retrieval techniques on data-centric XML, where the structure is very rich and carries important information about objects and their relationships. In Data Centric track, the task is to get a set of "closely related" nodes that are "collectively relevant" to a given keyword query. In XML keyword search, most XML documents can be seen as description of an entity (such as a person, a website or a company etc.), and most content of the document are about attributes of the entity (e.g., name or job of a person). In order to generate a good snippet [1][2] of an XML document, it is critical to pick up the most descriptive or representative attributes together with their values. In this paper, we present three main principles to judge the importance of attributes. First, whether the attribute is distinguishable or not. Second, whether the attribute is explicit or implicit. Third, whether the attribute describes the entity directly or indirectly.

Relevance feedback is a new track in INEX 2010. IR system with relevancefeedback permits interactivities between the users and the system. Users provide relevance (irrelevance) information of search result to IR system, which will be utilized by IR system to return more effective result. Relevance feedback track in INEX2010 simulates a single user searching with a particular query in an IR system that supports relevance feedback. The user highlights relevant passages of text and provides this feedback to the IR system. The IR system re-ranks the remainder of the unseen results list to provide more relevant results to the user. The Relevance Feedback track mainly focuses on the improvement of search result before and after implementing relevance feedback. As a consequence, our team pays more attention to acquiring more information through feedback rather than optimize results as in Ad hoc track.

In section 2, we explicitly introduce the five strategies used in Ad Hoc track and reveal the corresponding evaluation results. Section 3 describes our work on Data Centric track. In section 4, we show the methods applied for Relevance Feedback track. Section 5 is the conclusion.

2 Ad Hoc Track

In Ad Hoc track, the search engine combines five strategies: Two-Layer Retrieval, BM25, MAXLCA semantic query model, Distribution measurements and Learn-to-optimize method. Within these five strategies, Two-Layer retrieval and BM25 are two technologies that widely used and proven to be effective.

- **Two-Layer Retrieval**: Different from HTML, the querying atom of XML retrieval is elements rather than the whole documents. Thus, the core idea of Two-Layer retrieval is splitting the searching process into two layers. The first level starts from the traditional article retrieval. Then taking thetop returned relevant articles as querying database, the second layer further processes extracting the relevant elements. Finally, the extracted elements are ranked and returned in a result list form.
- **BM25**: Based onlots of research and comparative experiments, BM25 is confirmed to be an effective ranking method.It takes both text and structure information into consideration. Plus, evaluation results of Ad Hoc Track show thatBM25 performs better than some other frequently cited ranking models, such as TF*IDF [3] and so on. Motivated by BM25's excellent performance, we implant it into the search engine as a basic ranking method.

In the rest part of the section, we will propose the other three technologies applied in the search engine. MAXLCA is a semantic model defining which elements are relevant so that can be returned as results. Distribution measurements are ranking criterions used to evaluate the relevance between elements and queries according to the distribution of the keyword matches. Learn-to-optimize method is devised to tune the weights of different ranking methods in the final decision.

2.1 Maximal Lowest Common Ancestor (MAXLCA)

Due to that the returned results of XML retrieval can be elements, semantic query model are used to define which elements are relevant so that should be returned. Several approaches have been proposed for identifying relevant results, such as XRANK [4], SLCA [5] and XSeek [6] and so on. In this paper, we define a new semantic query model call MAXLCA [12], and compare it with another model All Common Ancestor (ACA). The formal definitions are as follow:

DEFINITION 2.1.1. Given a keyword query $Q = \{ k_1, \ldots, k_m\}$, an XML tree X_{tree}, we assume that V_i ($1 \leq i \leq m$) is the set of nodes that directly contains keyword k_i in X_{tree}. $LCA(Q, X_{tree})$is defined as follows:

$LCA(Q, X_{tree}) = \{n \mid \exists\ (v_1 \in V_1, \ldots, v_m \in V_m)$, n is the lowest common ancestors of $v_1, \ldots, v_m\}$

DEFINITION 2.1.2. Given a keyword query $Q = \{ k_1, \ldots, k_m\}$, an XML tree$X_{tree}$, we define $MAXLCA(Q, X_{tree})$ as follows:

$MAXLCA(Q, X_{tree}) = \{n \mid n \in LCA(Q, X_{tree}) \wedge \neg(\exists\ n' \in LCA(Q, X_{tree}),\ n' \succ n)\}$, where $n' \succ n$meansn' is an ancestor of n.

DEFINITION 2.1.3. Given a keyword query $Q = \{ k_1, \ldots, k_m\}$, an XML tree$X_{tree}$, we define $ACA(Q, X_{tree})$ as follows:

$ACA(Q, X_{tree}) = \{n \mid \exists\ (v_1 \in V_1 ,\ldots, v_m \in V_m)$, n is the common ancestors of $v_1 ,\ldots, v_m\}$.

To sum up, in ACA semantic query model, elements containing all matches of keywords are returned as relevant results. In MAXLCA, only the maximal LCA element of a document is returned.

2.2 Distribution Measurements

By observing a large amount of over 2000 query-result pairs, we interestingly discover that the distribution of the keyword matches in results plays a crucial role in picturing the theme of the passage. Moreover, four detailed statistical characteristics based on distribution are considered that presenting advantage capability on distinguishing relevant and irrelevant passages [13].

- **Distance Among Keywords (DAK)**. The minimum distance among the keywords is calculated. A passage with close matches of keywords in query will be more relevant.
- **Distance Among Keyword Classes (DAKC)**.The matches in passage of a certain keyword are firstly categorized into several subsets. The closer the keywords subsets are, the more relevant a passage is.
- **Degree of Integration Among Keywords (DIAK)**.The passage with high degree of integration will be more concentrating on one certain theme and should be given higher priority in the returned list.
- **Quantity Variance of Keywords (QVK)**. The passages whose numbers of different keywords vary significantly should be penalized.

The four distribution measurements evaluate the relevant between an element and a given query from different point of view. The weights of these features in final ranking module can be learned by the learn-to-optimize method introduced in next section. After all, the distribution measurements strategy belongs to ranking technology, modifying and completing the classic ranking function from the perspective of distribution of keyword matches in the passage.

2.3 Learn-to-optimize

Learn-to-optimize method [14] is proposed to tune the weights of the four features in distribution measurements. The Wiki English collection, queries and assessments of INEX 2009 Ad Hoc track are used as training samples.

In training, there is a set of query Q = {q^1,q^2,…,q^m} extracted from the INEX 2009 Ad Hoc track. Each query q^i is associated with a list of candidate elements E^i= (e^i_1,e^i_2, … ,$e^i_{n(i)}$), wheree^i_j denotes the the j-th candidate element to query q^i and n(i) is the size of E^i. The candidate elements are defined as MAXLCA or ACA elements. Moreover, each candidate elements list E^i is associated with a ground-truth list G^i= (g^i_1,g^i_2, … ,$g^i_{n(i)}$), indicating the relevance score of each elements in E^i.

Furthermore, for each query q^i, we use the distribution criterions defined in section 2.2 to get the predict relevant scores of each candidate element, recorded in R^i= ($r^i_1 r^i_2$,…,$r^i_{n(i)}$). In formula (1), S_{DAK}, S_{DAKC}, S_{DIAK} and S_{QVK} are the predicted scores for element j according to distance among keywords, distance among keyword classes, degree of integration among keywords and quantity variance of keywords respectively.

$$r^i_j = \alpha S_{DAK} + \beta S_{DAKC} + \gamma S_{DIAK} + \delta S_{QVK}$$

Then each ground-truth score list Gi and predicted score list Ri form a “instance”. The loss function L is defined as the Euclidean distance between standard results lists Di and search results lists Ri.

In each training epoch, the four criterions were used to compute the predicted score R^i. Then the learning module replaced the current weights with the new weights tuned according to the derivative of the loss between G^i and R^i. Finally the process stops either while reaching the limit cycle index or when the parameters do not change.

2.4 Comparison Results

In Ad Hoc track, the results submitted are obtained from five different combinations of the aforementioned strategies, illustrated in table 1.

Table 1. Results Submitted to Ad Hoc Track

	MAXLCA	ACA	Two-Layer	BM25	Distribution
AcaBM25		√		√	
MaxBM25	√			√	
RefMaxDis	√		√		√
RefMaxBM25	√		√	√	
RefMaxBM25Dis	√		√	√	√

Figure 1,2,3 present the evaluation results of Efficiency task, Relevance In Context task and Restricted Relevance In Context task respectively under measure as focused retrieval. Table 2 describes the evaluation results under document retrieval. As can be concluded: (1) Two-Layer search performs better than simple one layer element search; (2) MAXLCA semantic query model is more suitable for XML retrieval than ACA; (3) Rather than completely replacing BM25, distribution measurement is suitable for improving and modifying the drawbacks of it; (4) The method using Two-

Layer as retrieving strategy, MAXLCA as semantic query model, BM25 and distribution measurement as ranking functions shows the best performance.

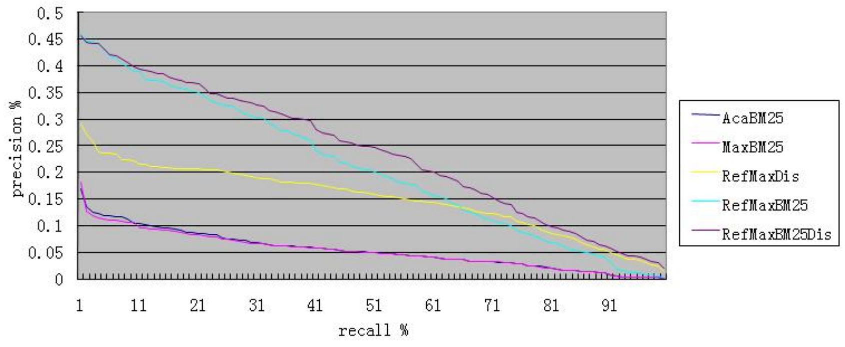


Fig.1. Evaluation Results of Efficiency Task

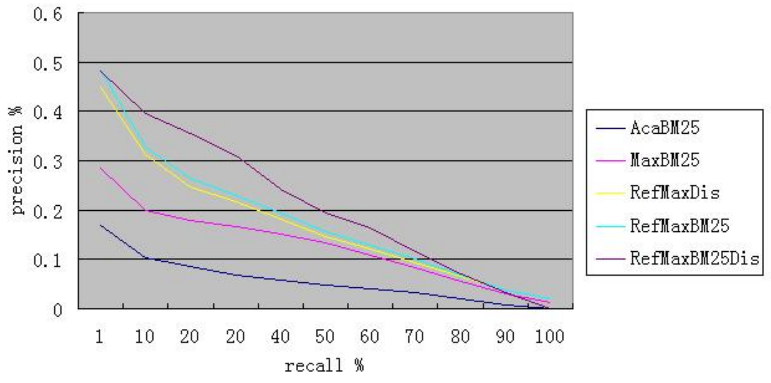


Fig.2. Evaluation Results of Relevance in Context Task

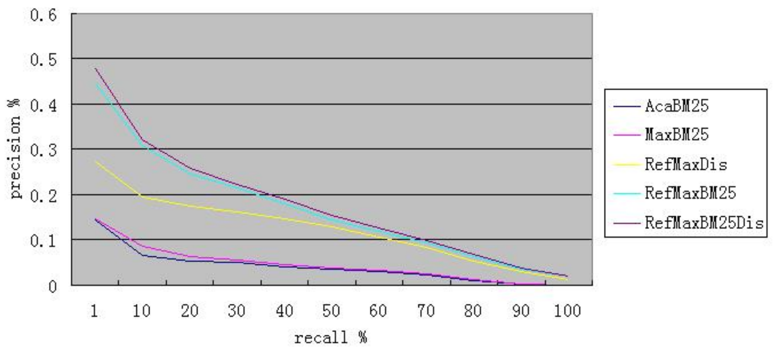


Fig.3. Evaluation Results of Restricted Relevance in Context Task

Table 2. Evaluation Results Under Measure as Document Retrieval

MAiP	AcaBM25	MaxBM25	RefMaxDis	RefMaxBM25	RefMaxBM25Dis
Efficiency	0.0538	0.0538	0.2404	0.3160	0.3047
Relevance In Context	0.0538	0.0538	0.2404	0.3160	0.3385
Restricted Relevance In Context	0.0102	0.0102	0.2404	0.3160	0.3202

3 Data Centric Track

In XML keyword search, there can be usually quite a number of results returned, only by the snippetof each result can the users judge whether it is useful to them, therefore it is important to generate a good snippetfor each result. [6][8] Pointed that a good XML result snippet should be a self-contained information unit of a bounded size that effectively summarizes the query result and differentiates itself from the others, meanwhile the authors have also accomplished a snippet generation system called eXtract [9].

The limited size of a snippet requests the search engine to extract the most important information (information here refers to attributes of an entity) from the result document, thus the main problem is to define the importance of attributes. In this paper, we present a semantic model MRepA (Most Representative Attribute) to evaluate the importance of an attribute to its corresponding entity, and afford three main principles to judge the importance of attributes. First, whether the attribute is distinguishable or not. Second, whether the attribute is explicit or implicit. Third, whether the attribute describes the entity directly or indirectly.

To be distinguishable, an attribute should meet the following two conditions. First, the attribute appears in all of the corresponding entities. Second, the attributes in different entities have different values. For example, as everyone has his or her name, and mostly different person has different name (though sometimes it is possible that two different person share the same name), so we can use the name to distinguish different person, however, we can’t use the nationality or driving license ID to distinguish different person.

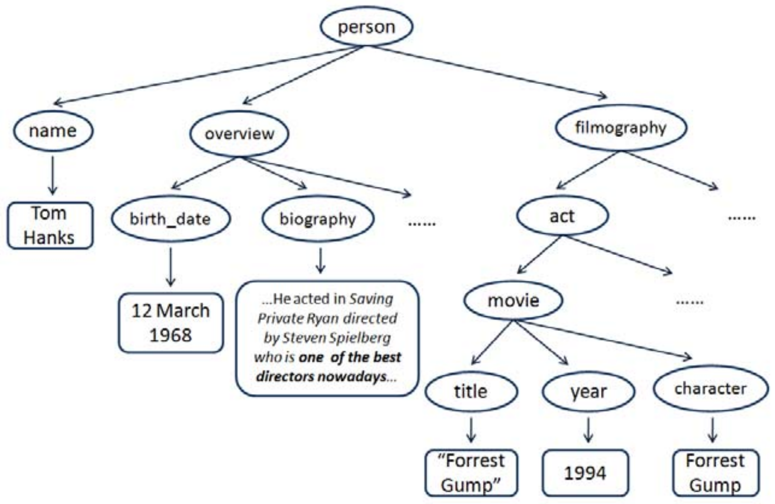


Fig.4.IMDB data segment

An attribute is explicit if all the content of the attribute value describes its corresponding entity only and not any other entities. For example, in figure 4 an XML document about *Tom Hanks* has two attributes *birth_date* and *biography*, and *birth_date* is an explicit attribute while *biography* is not, since *birth_date* is all about *Tom Hanks's* informationwhile *biography* contains the information about others (here, *"one of the best directors nowadays"*describes *Steven Spielberg* not *Tom Hanks*).

An attribute describes an entity directly means the attribute describes the entity itself but not any part of the entity, while an attribute describes an entity indirectly means the attribute describes some of the attributes of the entity or entities that are related to the entity (e.g., entity that is a son of the entity). For example, in figure 1 *Tom Hanks* is the name of the actor and we say it describes the entity (the actor) directly while *Forrest Gump* is the name of a movie that he acts in, so we can say that it describes the entity indirectly.

3.1 Extracting the Most Representative Attributes

In this section, we will discuss about how to integrate the three factors, which evaluate the importance of an attribute to an entity, into our semantic model MRepA [15].

3.1.1 Correlativeness Between Entities and Attributes

To evaluate if an attribute describes an entity directly or not, we have to analyze the relative position between the attribute and the entity in an XML document tree. In this section we define the entity-attribute path and use the number of entities on the path to measure the *correlativeness* between an attribute and its corresponding entity.

Definition 3.1.1.1: an entity-attribute path is a set of nodes those are on the path from the entity to the attribute (including the entity and the attribute).

Besides the number of entities on the entity-attribute path, the number of entities in the same level can be another factor that affects the correlativeness between an entity and an attribute. For example, for an actor named *Tom Hanks* in figure 1, as we discussed in section 1 *Forrest Gump* describes the actor indirectly, however, if *Forrest Gump* is the only movie that *Tom Hanks* acts in, then *Forrest Gump* can be very representative for actor *Tom Hanks*, while in the other hand, if *Forrest Gump* is just the one of a large number of movies that *Tom Hanks* acts in, then it may not be that much representative for actor *Tom Hanks*. So we integrate the two factors to evaluate the correlativeness between entities and attributes.

Definition 3.1.1.2: we define the correlativeness between entity e and attribute a R(e, a) as follows:

$$R(e,a) = k^{length(e,a)} \cdot \prod_{i=1}^{n} \frac{1}{m_i}$$

Where n=length(e, a) refers to the number of entities between entity e and attribute a, and m_i refers to the number of the entities of i-th category in the path.

For example, in figure 1, suppose there are 10 *movie* nodes, the entity-attribute path from node *person* to *title-Forrest Gump* is {person, filmography, act, movie, title}, and there are two entities *person* and *movie* on the path, and the number of person is 1 while the number of the movie is 10. If we set k as 1/2

3.1.2 Explicitnesses of attributes

In XML keyword search, length of the value of an attribute is usually associated with the explicitness of the attribute, and long text tends to be more descriptive but less explicit, while short text tends to be more explicit and clear. Thus we use the length (or the number of words) of the text to judge the explicitness of an attribute roughly.

Definition 3.1.2.1: we judge the explicitness of an attribute by the complicacy (we use the length here) of its value, and we sign the explicitness of attribute a as E(a).

3.1.3 Distinctiveness of attributes

As discussed in section 1, a distinguishable attributes should match the following two conditions: (1) The attribute appears in all of the entities; (2) The value of the attribute should be different in different entities.

In XML keyword search, there may not exist this kind of attributes. For example, we tend to use someone's name to differentiate one person from another, however, it is possible that two different person share the same name. Therefore in this section we afford the formula to calculate how much an attribute meet the demands.

Definition 3.3: we use distinctiveness of attributes to evaluate the degree of how much an attribute is able to distinguish different entities, and we sign the distinctiveness of attribute a as W_a.

$$W_a = \exp(p_a) \cdot H(a)$$

$$H(a) = -\sum_{i=1}^{n} p(a_i) \cdot \log[p(a_i)]$$

In the above formulas, $p_a$refers to the percentage of the correlative entities that attribute a appears, while H(a) is the chaos of attribute a, which estimates the variety of attribute a. The attribute's weightiness just evaluates the information an attribute itself can provide. To weigh how much an attribute contributes to an entity, we need the relationship between the attribute and the entity in addition.

4 Relevance Feedback Track

In relevance feedback track, our implementation employs two techniques: a revised Rocchio algorithm and criterion weight adjustment. In section 2.1, we will briefly introduce Rocchio algorithm and in section 2.2 we will show how we revise it. At last, we will we will show how we adjust criterion weights in section 2.3.

4.1 Rocchio Algorithm

Rocchio algorithm operates on vector space model [9], in which a document is represented by a vector of n weights like $\mathrm{d} = (t_1, t_2, t_3, t_4, \dots \dots, t_n)$, where n is the number of unique terms in document collections and t_i is the weight of the ith term in document d if d contains the term, else t_i equals to 0. And query is also represented as a vector just like a particular document only contains keywords.

The general idea of Rocchio is that the initial query may not express the purpose of a IR system user completely and effectively. Using the relevant or irrelevant document vector, RF could bring the query vector to express what the user needs more accurately. Rocchio's goal is to define an optimal query that maximizes the difference between the average vector of the relevant documents and the average vector of the irrelevant documents.

To achieve this, Rocchio add new query terms and reweight query terms in the query vector, making it more discriminative in choosing relevant documents from documents collection. The following formula shows the basic Rocchio algorithm:

$$Q_t = Q_0 + \frac{1}{n_1}\sum_{i=0}^{n_1} R_i - \frac{1}{n_2}\sum_{i=0}^{n_2} S_i$$

whereQ_0 is the initial query vector and $Q_t$is the revised query vector, $n_1$is the number of relevant documents and n_2 is the number of irrelevant documents, R_i is the vector of a relevant document and S_i is a irrelevant document vector.

After modification , the terms only appear in relevant document get a high positive weight and those only in irrelevant documents get a high negative weight, while the terms got relatively low weight if they appear in both relevant and irrelevant documents and have less discriminative power. It makes revised query vector contain more information about the difference of relevant and irrelevant documents.

Some researchers have modified and extended the formula such as assign different weight to original query terms [10] and added query terms or make constraints of number of documents used in the modification [11]. We won't introduce them here because they are basically the same.

4.2 Revised Rocchio Algorithm

Due to relevant information about each part of a relevant document is accessible in INEX2010 we divided a document into several paragraphs and represent each paragraph as a vector in our implementation. We treat a paragraph as a document in the searching process and give each paragraph a score. The score of a document is the weighted sum of its paragraphs' scores.

In our implementation, we redefine the formula in section 2.i as:

$$Q_t = Q_0 + \frac{1}{n}\sum_{i=0}^{n} P_i$$

Where P_i donates the vector of term weights calculated from a paragraph. Unlike original method to compute term weight , as how we define R_i and S_i in formula in section 2.1, we define P_i as follows:

For term $t_j$in P_i, its weight

$$w_j = \begin{cases} score_{P_i} & if\ P_i\ is\ a\ relevant\ paragraph\ in\ relevant\ document \\ 0 - score_{P_i - w_j} & if\ score_{P_i} is\ a\ paragraph\ in\ irrelevant\ document \\ 0 & otherwise \end{cases}$$

Where $score_{P_i}$ denotes the score of paragraph P_i , and $score_{P_i - w_j}$ denotes the score of paragraph after removing all t_j from it.

We will use an example to illustrate why we compute P_i like this. In INEX2009, the first topic of Ad hoc track is "Noble Prize". A user queries this for needing to prepare a presentation about the Nobel Prize. Therefore, he wants to collect information about it as much as possible. Assume there is a document with a simple paragraph as a section title, "Ig Nobel Prize". Apparently, the paragraph is not relevant because the

Ig Nobel Prize is an American parody of the Nobel Prizes organized by the scientific humor magazine *Annals of Improbable Research.* However, the score of this paragraph is relatively high for the reason that it only contains three words and two of them are keywords. Intuitively, we can figure out that the term "Ig" is a largely bad word for this topic for it turns a high-score paragraph to a irrelevant one. In addition, term "Nobel" or "Prize" seems has no responsibility for the irrelevance of this paragraph. However, if we use the formula in section 2.1, no difference between "Ig" and "Nobel" is reflected in the values of $R_i$or S_i. While in the revised model, the weights of "Ig" and "Nobel" are significantly different.

In the revised model, we focus on the contribution of a term to relevance or irrelevance of the paragraph it is belong to.

4.3 Criterion Weight Adjustment

To calculate score of a paragraph, we make three criterions. They are the frequency entropy, the mixing entropy and the weighted term distance between paragraph vector and query vector. The frequency entropy scales difference of terms' appearance frequency. It assigns a high score if all keywords appear the same number of times in a paragraph. The mixing entropy scales whether keywords appears alternatively. It assigns low score to a paragraph if it talks about one of the keyword at beginning while talks about another keyword at end without mixture of them. Each criterion makes contrition to the final score of a paragraph.

However, we can't decide which criterion is of greater importance to a specific topic. So we try to get this information from the feedback data. In the searching process, we keep the score history of every criterion and every keyword. Then at criterion weight update time, the discriminative power of each criterion and each keyword are computed. The discriminative power is computed as follows:

$$\mathrm{DP} = \frac{(\mu_r - \mu_{ir})^2}{d_r^2 - d_{ir}^2}$$

Where μ_r is the mean contribution of this criterion or keyword to relevant paragraphs and μ_{ir} is the mean contribution of this criterion or keyword to irrelevant paragraphs, d_r is the standard deviation of contribution of this criterion or keyword to relevant paragraphs and d_{ir} is the standard deviation of contribution of this criterion or keyword to relevant paragraphs

High DP value means strong discriminative power in current topic, so we raise its weight to let it make bigger contribution to scoring paragraph. While low DP value donates a criterion of keyword is not a suitable criterion for current topic.

For example, in our experiment, in the topic "Nobel Prize", these two keywords are assigned the same criterion weight, 0.5. However after all the document are returned. The criterion weight of "Nobel" rises to 0.89 but the "Prize" is only 0.11. The result is easy to understand. "Prize" is relatively a more widely word because there are a lot of prizes such as Fields Medal Prize, Turing Prize.

5 ACKNOWLEDGMENTS

This work was supported by the National High-Tech Research and Development Plan of China under Grant No.2009AA01Z136.

6 Reference

[1] http://www.inex.otago.ac.nz/.

[2] D. Carmel, Y.S. Maarek, M. Mandelbrod, et al. Searching XML documents via XML fragments. In SIGIR, pages 151—158, 2003.

[3] M. Theobald, R. Schenkel, G. Wiekum. An Efficient and Versatile Query Engine for TopX Search. In VLDB, pages 625—636, 2005.

[4] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In SIGMOD, 2003.

[5] Y. Xu and Y. Papakonstantinou.Efficient Keyword Search for Shortest LCAs in XML Databases.In SIGMOD, 2005.

[6] Z. Liu and Y. Chen.Identifying Meaningful Return Information for XML Keyword Search.In SIGMOD, 2007.

[7] Z. Liu, J. Walker, Y. Chen: XSeek: A Semantic XML Search Engine Using Keywords. In VLDB 2007: 1330-1333.

[8] Yu Huang, Ziyang Liu, Yi Chen. eXtract: A Snippet Generation System for XML Search. In VLDB 2008, Pages 1392-1395.

[9] Ian Ruthven and MouniaLalmas, A survey on the use of relevance feedbackfor information access systems. In The Knowledge Engineering Review (2003)

[10] E Ide. *New experiments in relevance feedback.*The SMART retrieval system experiments inautomatic document processing. (G. Salton ed). Chapter 16.pp 337-354. 1971.

[11] E. Ide and G. Salton.*Interactive search strategies and dynamic file organization ininformation retrieval*. The SMART retrieval system - experiments in automatic document processing.(G. Salton ed). Chapter 18.pp 373-393. 1971.

[12] Ning Gao, Zhi-Hong Deng, Jia-Jian Jiang, Yong-Qing Xiang, Hang Yu, MAXLCA A Semantic XML Search Model Using Keywords. Technical Report.

[13] Ning Gao, Zhi-Hong Deng, Hang Yu, Jia-Jian Jiang, ListOPT: A Learning to Optimize Method for XML Ranking. Technical Report.

[14] Ning Gao, Zhi-Hong Deng, Shenglong, Lv, Hang Yu, Jia-Jian Jiang, XDist A New XML Ranking Model Based on Keywords Distribution in Results. Technical Report.

[15] Jiajian Jiang, Zhihong Deng, Ning Gao, Shenglong Lv, Hang Yu. MRepA: Extracting the Most Representative Attributes in XML Keyword Search. Technical Report.

Web Service Discovery Track Overview

James A. Thom[1] and Chen Wu[2]

[1] RMIT University, Melbourne, Australia
james.thom@rmit.edu.au
[2] Curtin University, Perth, Australia
Chen.Wu@cbs.curtin.edu.au

Abstract

An efficient and effective Web services discovery mechanism is important in many computing paradigms including Pervasive Computing, Service-Oriented Computing, and the most recent Cloud Computing, in which Web services constitute the chief building blocks. The Web Service Discovery track aims to investigate techniques for discovery of Web services based on searching service descriptions provided in Web Services Description Language (WSDL). Participating groups have contribute to topic development and will contribute to evaluation, which will then allow them to compare the effectiveness of their XML retrieval techniques for the discovery of Web services. This will lead to the development of a test collection that will allow participating groups to undertake future comparative experiments. This track would not been possible without the support of the INEX organisers in providing software support, as it makes extensive use of the existing INEX infrastructure. The results for the Web Services Discovery track are not yet available.

Semantics-based Web Service Discovery Using Information Retrieval Techniques

J.Hou, J.Zhang, R.Nayak, A.Bose

Faculty of Information Technology, Queensland University of Technology,
2 George Street, GPO Box 2434, Brisbane, QLD 4001 Australia
{jun.hou, jinglan.zhang, r.nayak, a.bose}@qut.edu.au

Abstract. This paper demonstrates an experimental study that examines the accuracy of various information retrieval techniques for Web service discovery according to users' queries. The evaluation is comprehensively benchmarked using more than 1,700 real-world WSDL documents from INEX 2010 Web Service Discovery Track dataset. For automatically search, we successfully use Latent Semantic Analysis and BM25 to perform Web service discovery. Moreover, we provide linking analysis to automatically link possible atomic Web services for complex requirements of users. After that, the fusion engine integrates results and recommends a final result to users. Based on evaluation, linking analysis can provide flexible combinations of Web services based on users' preferences. Search results show that the combination of Latent Semantic Analysis and linking analysis can improve the overall performance of Web service discovery.

Keywords: Web service discovery, Semantics, Latent Semantic Analysis, Linking Analysis

1 Introduction

With the popularity of Service Oriented Architecture (SOA), many enterprises offer their distributed Web services as interfaces for their core business systems. Web Services are embracing an unprecedented attention from the computer world. Web services can be discovered by matchmaking requirements of service requesters with service specifications from service providers. Web service discovery plays a key role in finding appropriate Web services. Since a number of Web services are provided by different organizations and there are still no standards for Web service design and provision. It is a challenging and interesting task to identify accurate Web services.

Web Service Description Language (WSDL), the standard description language, and Universal Description Discovery and Integration (UDDI) for advertising Web services, are introduced to discover and invoke existing Web services. Web services Requesters and providers then communicate with each other by SOAP message, a XML format communication language based on HTTP. However, WSDL and UDDI search mechanism utilizes syntactic search based on keyword. Semantic search is needed to enhance the search performance. In addition, since different organizations design services in enclosed circumstance, atomic Web services cannot satisfy different users' requirements [8]. Therefore, a set of Web services need to be composed to fulfil the given tasks.

The main goal of this research is to evaluate algorithms for semantic Web service discovery using WSDL1.1–compliant documents. Two methods are used in this paper, namely Latent Semantic Analysis supported by Wikipedia corpus and BM25 supported by WordNet. Wikipedia corpus is used to create Latent Semantic Kernel, while WordNet is introduced to improve the search performance of BM25. On top of that, we propose a linking analysis, which can automatically compose possible atomic Web services to conduct user-preferred tasks. In the fusion engine, a new result with atomic and composite Web services is recommended to users. There are six submissions created by each method and their combinations with linking analysis.

2 Related Work

This section summarizes some previous work in semantic Web service discovery.

Due to the lack of semantics in WSDL, many semantic Web service description languages such as OWL-S, WSMO and WSDL-S have emerged to explicitly annotate WSDL with semantic information. OWL-S and WSMO demonstrate Web services semantics at a distinct level [7]. OWL-S is more concentrated on the "Upper ontology" (not domain-specific ontology) for describing Web services [2]. Compared to OWL-S, WSMO is more focused on producing a reference implementation of an execution environment, the Web Service modelling execution environment and specifying mediators [9]. Mediators are not the significant consideration in OWL-S conceptual and implementation [12]. However, the discovery mechanism in WSMX is based on keyword and simple semantic description [12]. Compared to OWL-S, WSDL-S has several advantages over OWL-S. First, details of both the semantics and operations can be described in WSDL. In addition, the semantic domain models are detailed externally, which offers Web service developers an opportunity to select preferred ontology language. On top of that, the existing tooling can be updated relatively easy. The objectives of WSDL-S are to be of compatibility with OWL-S with emphasises on a more lightweight and incremental approach [9]. Although more lightweight and flexible (supporting different ontologies) ontology languages are emerging, there is still no standard ontology and the maintenance cost is very high with low scalability.

Many researchers make use of traditional Information Retrieval techniques. They parse WSDL files (documents) into bags of words and create terms-documents. Then Webs services are ranked by TF-IDF (term frequency –inverse document frequency) or LSA (Latent Semantic Analysis)/LSI (Latent Semantic Indexing) according to the term (search query) frequency in each document. A binning & merging-based Latent Semantic Kernel [1] is proposed to enhance the semantics of LSA. Experiment result shows that LSA approach can be acceptable both in scalability and complexity [13]. A method using surface parsing of sentences to add structural relations [3] are proposed to improve the performance on single sentences in LSA. However, there still are some issues related to LSA. When the pre-process (stop word removal and stemming) reduces common terms and outliers, WSDL structure is broken at the same time. Nayak & Iryadi [10] and Hao & Zhang [5] propose Schema matching approaches in WSDL-based Web service discovery. Such approaches try to find not only text but also structure information for comparing WSDL documents. To effectively investigate semantics in text, a Wikipedia-based structural relationship-enhanced concept thesaurus [6] is introduced. This approach concentrates on improving the semantic relationships between important terms by applying text clustering.

Researchers are devoted to dig more semantic information from current Web resources. Ding, Lei, Jia, Bin, & Lun [4] propose a discovery method based on Tag. Tags are widely used in images, bookmarks, blogs and videos to annotate the content of them. This approach suffers the same problem of above ontology language. It is limited by the scope of comment on Web services and variety between different comment styles. Semantic Web Services Clustering (SWSC) [11] makes use of preconditions and effects from OWL-S to improve the accuracy of Web service discovery. Using translation tools, more context information such as preconditions and effects after invocation can be collected thereby increasing the consistency of Web service discovery. In this method, hidden Web services can be discovered and be attached to similar groups before conducting search. However, the scalability is still a problem.

3 Discovery approach

Our approach is a novel three-phase. Figure 1 shows the overview of Web service discovery methodology. In the semantic analysis phase, there are two methods used to retrieve atomic Web services, namely Latent Semantic Analysis (LSA) supported by Wikipedia corpus and BM25 supported by WordNet. Before applying those approaches, standard text pre-processing is performed to parse WSDL documents into bags of words. During this stage, stop word removal and stemming has been executed.

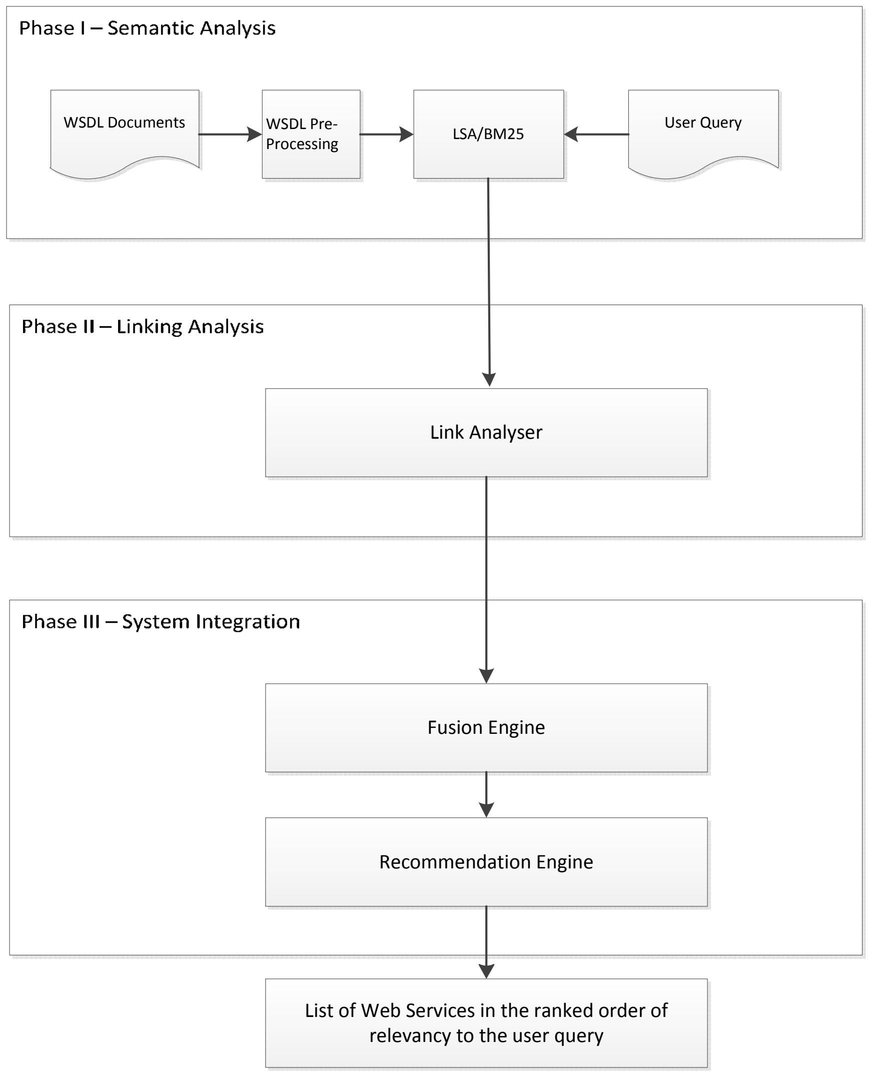

Phase I – Semantic Analysis
WSDL Documents
WSDL Pre-Processing
LSA/BM25
User Query
Phase II – Linking Analysis
Link Analyser
Phase III – System Integration
Fusion Engine
Recommendation Engine
List of Web Services in the ranked order of relevancy to the user query

Fig. 1. Overview of Web Service Discovery Methodology

3.1 Pre-Processing

Stop word removal aims to reduce words which act poorly as index terms. For example, those words can be "a", "the", "and" etc. An external stop word list is introduced to filter out those words to perform data analysis.

Stemming is a process to replace words with their root or stem forms by removing affixes (suffixes or prefixes). Words such as "computing", "computer" and "computed” will be replaced by the word “compute”. This process reduces not only the variety of words also the computation cost. The Porter Stemming Algorithm is used to conduct the stemming process.

3.2 Semantic Analysis

Latent Semantic Analysis (LSA)
Figure 2 shows the overview of Latent Semantic Analysis (LSA). In LSA, the semantic kernel is used to find semantic similarity between Web services and users’ queries. The semantic kernel is constructed from a general-purpose dataset. Wikipedia dataset is chosen because it is not domain-specific and covers various topics. Figure 2 shows the overview of LSA in phase I.

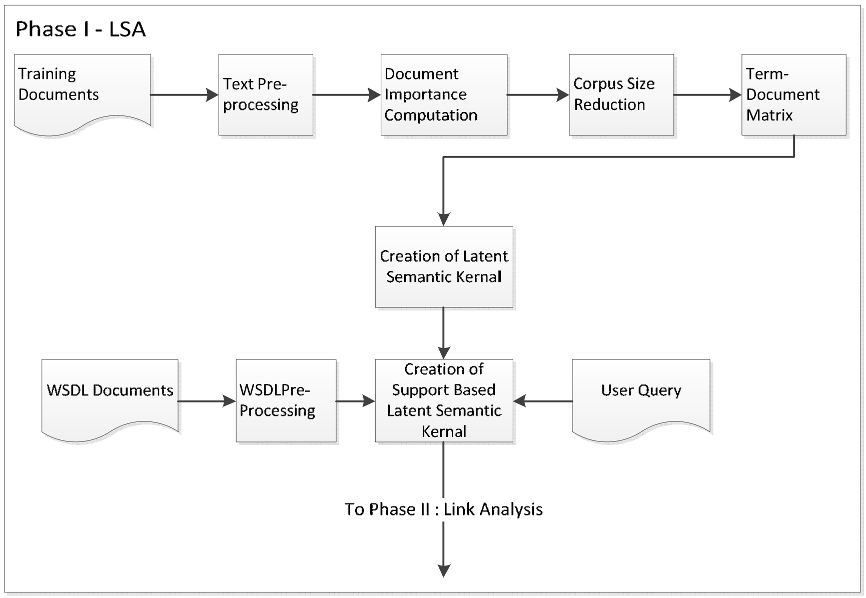

Phase I - LSA
Training Documents
Text Pre-processing
Document Importance Computation
Corpus Size Reduction
Term-Document Matrix
Creation of Latent Semantic Kernal
WSDL Documents
WSDLPre-Processing
Creation of Support Based Latent Semantic Kernal
User Query
To Phase II : Link Analysis

Fig. 2. Overview of LSA

To start, each pre-processed WSDL document is then encoded as a vector. Components of the vector are terms in the WSDL document. Each vector component reflects the importance by TF*IDF. The user query is also converted to a vector which is compared with the vector of a WSDL document. The similarity between the user query (Q) and the Web service document (W) is represented by the cosine value of two vectors. Equation 1 shows how to calculate the cosine value.

$$Sim\,(Q,W) = Cos\,(Q,W) = \frac{Q \cdot W}{||Q||\;||W||}\,. \tag{1}$$

However, we use semantic kernel (K) here to enhance the semantics between Q and W. The Q and W is replaced with $Q^T K$ and $K^T W$ respectively. Equation 2 shows the improved equation with semantic kernel.

$$Sim\,(Q,W) = Cos\,(Q,W) = \frac{Q^T K \cdot K^T W}{||Q^T K||\,||K^T W||}\,. \tag{2}$$

Finally, the top-k Web services are returned to users (k is set as 20).

BM25

BM25 is a bag-of-words retrieval algorithm that ranks documents based on the query terms appearing in each document. To increase the amount of query terms, WordNet is introduced to incorporate with BM25. WordNet is a general ontology, which can boost semantics from users' queries. Figure 3 shows the overview of using BM25 in phase I.

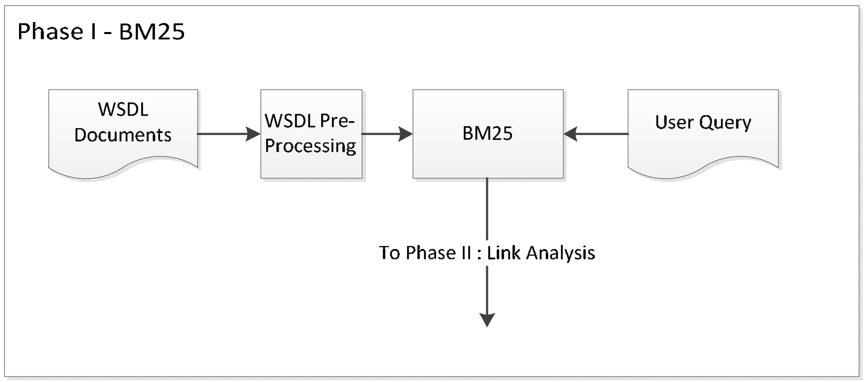


Fig. 3. Overview of BM25

After pre-processing, WSDL documents (W) are computed with users' queries (Q) for similarity. Equation 3 shows the major equation of BM25.

$$Score(Q,W) = \sum_i^n IDF(q_i) \cdot \frac{f(q_i,W) \cdot (k_1+1)}{f(q_i,W) + k_1 \cdot (1 - b + b \cdot \frac{|W|}{avgdl})}\,. \tag{3}$$

In $f(q_i, W)$, q_i is the term frequency in the WSDL document W. |W| is the length of the WSDL document and avgdl is the average document length in the text collection. Equation 4 shows the details of $IDF(q_i)$.

$$IDF(q_i) = log\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}. \quad (4)$$

N represents the total number of WSDL document in the collection and $n(q_i)$ is the number of WSDL documents containing q_i.

Same as LSA, the top-k Web services are returned to users (k is set as 20).

3.3 Linking Analysis

Web services are retrieved based on the query of a user. However, one Web service may not meet the requirement of the query of a user. For example, the query from a user is “weather by postcode” and the actual Web service is “weather by location”. Obviously, the Web service needs to corporate with another Web service such as “postcode to location”. Linking analysis aims to link possible Web services to satisfy the requirements of users. Figure 4 shows the overview of linking analysis.

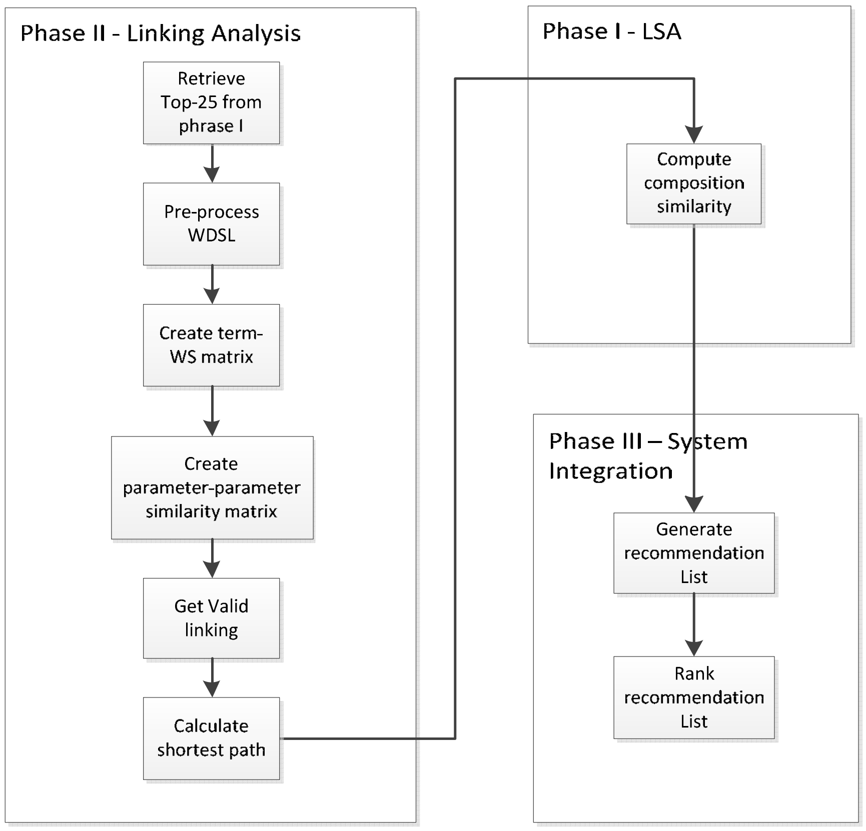

Phase II - Linking Analysis
Retrieve Top-25 from phrase I
Pre-process WDSL
Create term-WS matrix
Create parameter-parameter similarity matrix
Get Valid linking
Calculate shortest path
Phase I - LSA
Compute composition similarity
Phase III – System Integration
Generate recommendation List
Rank recommendation List

Fig. 4. Overview of Linking Analysis

In linking analysis, we use top 25 results from LSA or BM25 instead of directly linking Web services in the collection. In a WSDL document, the <PortType> tag consists of sets of <Operation> tags, which contain the description of invocable functions. We consider that Web services can be linked together if one Web service's output parameters match another one's input parameters in parameter name, parameter amount and data type. That information of input and output parameters is extracted for linking analysis. During the extraction, non-topic words such as "result" and "response" are filtered out.

Once we get parameter names, they are decomposed into tokens. For instance, "ChangePowerUnit" is split into "Change", "Power" and "Unit" from each capital letter. If two parameter tokens are exactly same, we consider it as exact match. However, there are parameters having tokens such as "car" and "vehicle" and they semantically can be linked. Therefore, we calculate the similarity between input and output parameters to semantically link two Web services. Equation 5 shows how to compute the similarity of two parameters.

$$Sim\ (P_1, P_2) = \frac{n \cdot Sum\ (sim\ (W_{P_1}, W_{P_2}))}{N}. \tag{5}$$

$Sum\ (sim\ (W_{P_1}, W_{P_2}))$ is the sum of the similarity between tokens in parameter P_1 and P_2. N represents the total number of parameter tokens in P_1 and P_2. For n, it is the number of parameter tokens which is less than the other one. For example, if P_1 has 2 tokens and P_2 has 3 tokens, n will be 2 and N will be 5. If $Sim\ (P_1, P_2)$ is greater than 0.98, we consider that the two parameters can be linked (linkable parameters). Furthermore, we use another factor, link strength, to decide if the two Web services can be linked. Link strength demonstrates the compatibility of two Web services by the number of linkable parameters. Equation 6 shows how to calculate the link strength.

$$Link\ Strength = \frac{N_l}{N_I}. \tag{6}$$

N_l is the total number of linkable parameters and N_I is the number of input parameters of one Web service. Once we have the link strength, functions of Web services are converted to a graph where nodes representing functions are connected with each other by link strength. Afterwards, we use Floyd Warshell algorithm to calculate the shortest path from each method to all other methods. We define composition strength as the average of link strength of a composition. All compositions are ordered by composition strength. Each composition is treated a new Web service and compared with users' queries for similarity by LSA.

3.4 System Integration

The main purpose of integration is to integrate the composition of Web services with atomic ones from LSA or BM25. The most important task is to decide which result appears in the final list. Generally, composition result has a higher accuracy than atomic one. In addition, if a Web service is the component of a composition, it will not appear in the final result. As a result, we select all compositions to the final result and then add atomic results to form top 20 recommendations. Figure 5 shows the overview of System Integration.

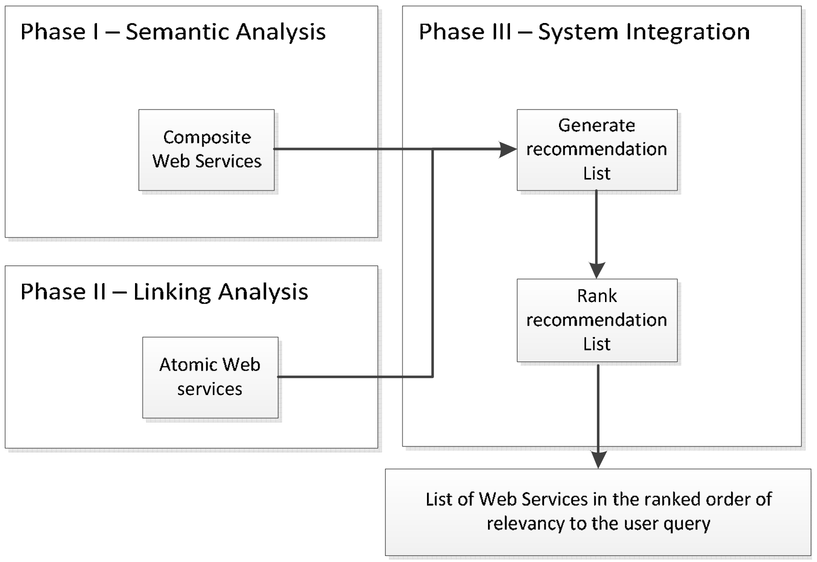

Phase I – Semantic Analysis
Composite Web Services
Phase II – Linking Analysis
Atomic Web services
Phase III – System Integration
Generate recommendation List
Rank recommendation List
List of Web Services in the ranked order of relevancy to the user query

Fig. 5. Overview of System Integration

4 Data Set

The document collection used is provided by the INEX 2010 organizing committee. The dataset contains over 1,700 documents in the format of WSDL 1.1, which are directly crawled from real-world public Web services indexed by the Google search engine.

5 Evaluation

There are 25 topics for evaluation and we submitted results but still wait for the manual evaluation/verification results currently. User queries are created by competition participants to ensure the variety of queries. Final results will be added after INEX2010 – Web Service Track has conducted the evaluation and published the results.

6 Conclusions and Future Work

The experiment result shows that Latent Semantic Analysis improves the search performance by discovering the semantic relationship between users' queries and WSDL documents. Link analysis automatically composes Web services to fulfill complex tasks.

In this paper, Web services are converted to bags of words and then compared with users' queries for similarity. However, WSDL documents are not like normal documents having high richness of terms. More decomposition rules are needed to deal with abbreviation and artificial names when parsing WSDL documents. Furthermore, the single term cannot describe function very well and simply investigating semantics by single words may mislead search result. In addition, the user query is overly simple and Web services are involved in more complex business scenarios. Service choreography and service orchestration are considered when deploying and invoking Web services. Web services contain more business relationships than normal documents, especially during invocation. More practical situations need to be investigated to effectively invoke Web services.

Acknowledgement
The coding of the research described here is supported by Ping-Liang Lin and Jianing Chen.

References

[1] Bose, A., Nayak, R., & Bruza, P. (2008). Improving Web service discovery by using semantic models. *Proceedings of the 9th International Conference on Web Information Systems Engineering* (pp. 366-380). Auckland, New Zealand: Springer-Verlag.

[2] Burstein, M. H., & McDermott, D. V. (2005). Ontology translation for interoperability among semantic Web services. *AI Magazine*, 26(1), 71-82.

[3] Dennis, S. (2007). *Handbook of latent semantic analysis*. Mahwah, N.J: Lawrence Erlbaum Associates.

[4] Ding, Z., Lei, D., Jia, Y., Bin, Z., & Lun, A. (2010). A Web service discovery method based on tag. *The 2010 International Conference on Complex, Intelligent and Software Intensive Systems* (CISIS) (pp. 404-408). Krakow, Poland: IEEE Computer Society.

[5] Hao, Y., & Zhang, Y. (2007). Web services discovery based on schema matching. *Proceedings of the 13th Australasian Conference on Computer Science* (pp. 107-113). Ballarat, Australia: Australian Computer Society.

[6] Hu, J., Fang, L., Cao, Y., Zeng, H.-J., Li, H., Yang, Q., et al. (2008). Enhancing text clustering by leveraging Wikipedia semantics. *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 179-186). Singapore, Singapore: Association for Computing Machinery.

[7] Lara, R., Roman, D., Polleres, A., & Fensel, D. (2004). A conceptual comparison of WSMO and OWL-S. In L.-J. Zhang & M. Jeckle (Eds.), *Web Services* (pp. 254-269). Berlin: Springer-Verlag.

[8] Li, Q., Liu, A., Liu, H., Lin, B., Huang, L., & Gu, N. (2009). Web services provision: Solutions, challenges and opportunities (invited paper). *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication* (pp. 80-87). Suwon, Korea: Association for Computing Machinery.

[9] Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., et al. (2007). Bringing Semantics to Web Services with OWL-S. *World Wide Web*, 10(3), 243-277.

[10] Nayak, R., & Iryadi, W. (2007). XML schema clustering with semantic and hierarchical similarity measures. *Knowledge-Based Systems*, 20(4), 336-349.

[11] Nayak, R., & Lee, B. (2007). Web service discovery with additional semantics and clustering. *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence* (pp. 555-558). Fremont, USA: IEEE Computer Society.

[12] Shafiq, O., Moran, M., Cimpian, E., Mocan, A., Zaremba, M., & Fensel, D. (2007). Investigating Semantic Web Service Execution Environments: A Comparison between WSMX and OWL-S Tools. *Proceedings of the 2nd International Conference on Internet*

and Web Applications and Services (pp. 31-36). Morne, Mauritius: IEEE Computer Society.

[13] Wu, C., Potdar, V., & Chang, E. (2009). Latent Semantic Analysis - The dynamics of semantics Web services discovery. In *Advances in Web semantics I: Ontologies, Web Services and Applied Semantic Web* (pp. 346-373). Berlin: Springer-Verlag.

A first approach to web service discovery

María J. Somodevilla, Beatriz Beltrán, David Pinto,
Darnes Vilariño, José Aaron
mariasg,bbeltran, dpinto,darnes@{cs.buap.mx}

Faculty of Computer Science
Benemérita Universidad Autónoma de Puebla, México

Abstract. A clustering approach for web services discovering based on techniques from Information Retrieval (IR), Natural Language Processing (NLP) and XML Retrieval was developed in order to use texts contained in WSDL files is proposed. It calculates the degree of similarity between words and their relative importance to support the task of web services discovering. The algorithm uses the information contained in the WSDL (Web Service Description Language) specifications and clusters web services based on their similarity. A second approach based on a simple information retrieval system that index terms by using an inverted index structure was also used. Both algorithms are applied to a set of 1947 real web services in 25 categories Web Service Discovery, all of them provided by INEX.

1 Introduction

The Service Oriented Architecture (SOA)[1] was developed based on the concept of a wide mesh of collaborating services, published and available for invocation. Web services are the set of protocols by which services are published, discovered, and used in a technology independent, standard form. As the number of web services repositories grows and the number of available services expands, finding the web service that one needs has become a key task within the invocation process. Web service discovery is concerned with locating web services that match a set of functional and non-functional criteria [1].

The Web Services Description Language (WSDL) [2] is the most basic mechanism used to describe web services. This leads many current discovery approaches to focus on locating web services based on their functional description.

An efficient and effective Web services discovery mechanism is important in many computing paradigms including Pervasive Computing, Service-Oriented Computing, and the most recent Cloud Computing, in which Web services constitute the chief building blocks. The Web Service Discovery track aims to investigate techniques for discovery of Web services based on searching service descriptions provided in Web Services Description Language (WSDL) . Participating groups will contribute to topic development and evaluation, which will

[1] XML Web Services. http://webservices.xml.com/lpt/a/129
[2] WSDL Specification. http://www.w3.org/TR/wsdl.html

then allow them to compare the effectiveness of their XML retrieval techniques for the discovery of Web services. This will lead to the development of a test collection that will allow participating groups to undertake future comparative experiments.

The rest of this paper is devoted to explain the two different approaches submitted to the competition, as well as the dataset used in the experiments.

2 Description of the presented approaches

Two algorithms based on Clustering and Information Retrieval in order to find the most appropiate web service (WSDL file) to a given topic were developed.

2.1 Clustering Approach

In Figure 1 we may see the approach that uses a clustering method. The complete description follows this figure.

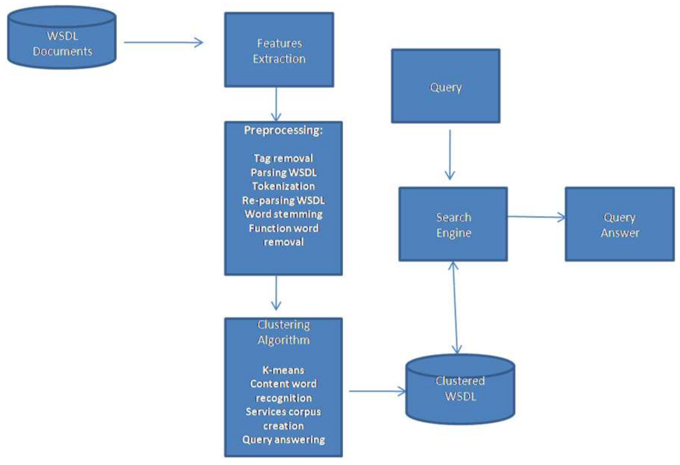


Fig. 1. An overview of the presented approach

1. Tag removal: A corpus was built with the content of the XML tags for each document, in addition to the attribute values of the labels.
2. Parsing WSDL: Stopwords and punctuation symbols were removed from the corpus.
3. Tokenization: MMA algorithm was applied [2], using a list of 53 000 English words which split them into tokens (i.e. GetAllitems as Get All Items).
4. Re-parsing WSDL: Stopwords and punctuation symbols were removed from the corpus again due to the MMA decomposition.
5. Word stemming: The Porter stemming algorithm was applied to the corpus.

6. Function word removal: Words with frequency less than 10 were eliminated.
7. K-means algorithm: K=2 was used; the distance criterion NGD is presented in Eq. (1), and the convergence criterion is that the centroid words are at least twice in different iterations.
8. Content word recognition: Thereafer, we removed the words of the cluster with minimal elements (i.e. service, soa, array and data).
9. Services corpus creation: A second corpus was constructed with the services of each XML file; again we used the MMA algorithm, we eliminate stopwords, and finally we applied the Porter algorithm.
10. Query answering: Using the two corpus constructed, a query can be answered by applying Eq. (2), and sorting the results from lowest to highest.

$$\mathrm{NGD}(x,y) = \frac{\max\{logf(x), logf(y)\} - logf(x,y)}{logM - \min\{logf(x), logf(y)\}} \tag{1}$$

$$O(S_i,S_j) = 0.5 * S'(S_i,S_j) + 0.5 * S''(S_i,S_j) \tag{2}$$

where:

$$S'(S_i,S_j) = \frac{\sum_{a\in S_i}\sum_{b\in S_j} Sim(a,b)}{|S_i||S_j|} \tag{3}$$

$$S''(S_i,S_j) = 1 - \mathrm{NGD}(S_i,S_j) \tag{4}$$

2.2 Using information retrieval for webservice discovery

In Figure 2 we may see the approach that uses information retrieval for finding the corresponding web services files that satisfies the user needs.

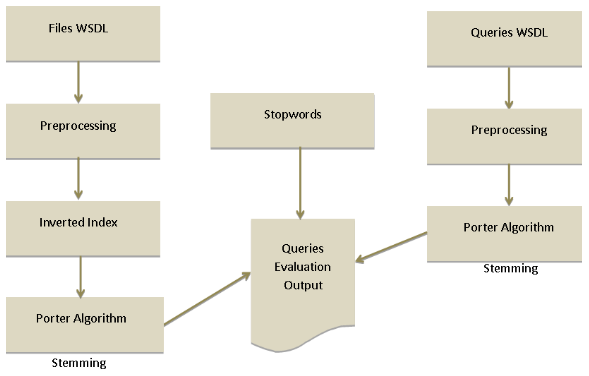


Fig. 2. An overview of the presented approach

The implementation uses an inverted index for storing all the terms detected in the WSDL files. For the case of function names, we have also used the MMA algorithm. Each term is used as the dictionary entry in the data structure, and one posting list is attached to each dictionary entry. Finally, given a query, we may calculate the intersection between pairs of posting lists (p_1 and p_2) as shown in are given in Algorithm 1 (taken from [3]).

Algorithm 1: Intersection of two posting lists

```
Input: Posting lists $p_1$ and $p_2$
Output: Relevant documents $D_1, D_2, \cdots$
answer = ⟨⟩
while $p_1$! = NIL and $p_2$! =NIL do
    if docID($p_1$) = docID($p_2$) then
        ADD(answer, docID($p_1$));
        $p_1$ = next($p_1$);
        $p_2$ = next($p_2$);
    else
        if docID($p_1$) < docID($p_2$) then
            $p_1$ = next($p_1$)
        else
            $p_2$ = next($p_2$)
        end
    end
end
return answer
```

3 Conclusions

We have presented details about the implemented approaches for tackling the problem of webservice discovery. Two different approaches were implemented, one based on clustering and the second on information retrieval techniques. Unfortunately, at the moment of publishing this paper, the evaluation results were not available, therefore, we are not able to discuss the performance of these approaches.

References

1. Dan, A., Davis, D., Kearney, R., Keller, A., King, R., Kuebler, D., Ludwig, H., Polan, M., Spreitzer, M., , Youssef, A.: Web services on demand: Wsla-driven automated management. IBM Systems Journal **43**(1) (2004) 136–158
2. Elgazzar, K., Hassan, A.E., Martin, P.: Clustering wsdl documents to bootstrap the discovery of web services. In: Proceedings of the 2010 IEEE International Conference on Web Services. ICWS '10, Washington, DC, USA, IEEE Computer Society (2010) 147–154
3. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK (2009)

RMIT participation in Web Service Discovery Track

James A. Thom

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

Abstract. This paper describes the participation of RMIT in the 2010 Web Service Discovery track of INEX. In this paper we describe a naive document level approach that focuses on how to preprocess the WSDL documents before indexing. An addendum to the paper briefly describes our participation in the ad hoc track.

Introduction

The web service discovery task has several key differences from ad hoc XML retrieval. These include:

– the lack of text nodes in many documents
– information is often included as attribute values
– attribute values are often not ordinary English words, but invented compounds often using case changes

Approach and Results

Our main observation was that much of the content in the WSDL was not in ordinary text nodes, but was in other parts of the XML documents such as attribute values. In many cases these were not ordinary English words but were often compounds such as "msdSSMPurgeResponse". So we wrote a simple sed script that attempted to extract useful words to index from the documents splitting up any compound names.

```
1i\
<html>
s/[^a-zA-Z]+/ /g
s/[A-Z]?[a-z]+|[A-Z]+/ & /g
$a\
</html>
```

We then indexed the resulting files with Zettair, an open source search engine developed at RMIT. We used a stable released version of Zettair (version 0.9.3) which we ran on a MacBook Pro computer with 2.4 GHz Intel Core 2 Duo running Mac OS X version 10.6.4 with 3 MB of L2 Cache and 4 GB memory. Our runs used the CO titles of the topics as the queries and we used the default similarity measure in Zettair (a language model with Dirichlet-smoothing) which has been shown to perform adequately in the ad hoc task in previous years.

The results for the Web Services Discovery track are not yet available.

Addendum

We also submitted two "vanilla" document level runs to the ad hoc track that were similar to runs from 2009. In the queries we used the CO titles of the topics and two similarity measures: `RMIT09title` used the default similarity measure in Zettair (a language model with Dirichlet-smoothing), and `RMIT09titleO` used Okapi BM25. The best-entry-point was set at the start of the article.

XML Retrieval More Efficient Using Double-Scoring Scheme

Tanakorn Wichaiwong and Chuleerat Jaruskulchai
Department Of Computer Science,
Faculty of Science,Kasetsart University,
Bangkok Thailand

Abstract

This is the first year for the Kasetsart University participation of INEX. We participated in three tracks; Ad Hoc, Data Centric, and Web Service Discovery tracks. In this paper, we report experimental results of our approach using Double-Scoring scheme base on BM25 model for retrieval large-scale XML collection. This model is commonly used in the information retrieval community. Our approaches present a way to reduce parameter tuning step by extended new indices to store all of selected fields, to avoid the amount of parameters tuned weights for each selected fields of BM25F.

Keywords

XML Retrieval, Information Retrieval, Indexing Units, Ranking Schemes

1. Introduction

The widespread use of Extensible Markup Language (XML) [1] documents in digital libraries led to the development of information retrieval (IR) methods specifically designed for XML collections. Most traditional IR systems are limited to whole document retrieval; however, since XML documents separate content and structure, XML-IR systems are able to retrieve only the relevant portions of documents. Therefore, users who utilize an XML-IR system could potentially receive highly relevant and precise material. In the Initiative for the Evaluate of XML Retrieval (INEX) [2] reports, The BM25F run performs excellent when evaluated at the element level and results showed that first ranked in INEX. However, there is still limitation in parameters tuned weights for each selected fields. Our approaches reduce the parameters tuning by extended indices scheme.

This paper is organized as follows: section 2 reviews related works, section 3 explains the implementation of our system, section 4 presents the experiment, and conclusions and recommendations for further work are provided in section 5

2. Related Works

On INEX 2005, Robertson et al. [3, 4, 5] applied an earlier version of BM25F to XML element retrieval, reporting 65% improvements over BM25 measured by nxCG on INEX-IEEE collection with a different task where overlap is allowed. In that work, an element's score is computed from multiple fields, which may include the body of the element, the document's title, the documents abstract, and ancestral section titles.

On INEX 2009, Kelly and Charles [6] applied BM25F on Wikipedia's collection; on each element, they constructed two fields, one for "title" and another for "body". The title field consists of concatenation of an article title and any section titles. The body field contains the rest of the context. In the INEX reports [7], the results showed that ranked first with iP [0.01] is 0.6333.

2.1 BM25F Scheme Overview

Robertson et al. [3, 4, 5], presented the BM25F scheme, an extension of BM25 that exploits structural information. Under BM25F, terms contained in a document's title, for example, may be given more weight than terms contained in the document's body. Using BM25F scheme that can compute an element's score as follows:

$$\mathrm{BM25F}(e) = \sum_{t \in q \cap e} \frac{X_{e,t}}{K + X_{e,t}} * W_t$$

Note that;
BM25F(e) measures the relevance of element e to a query q.
q is a set of query terms.
$X_{e,t}$ is a weighted normalized term frequency.

K is a common tuning parameter for BM25.

W_t is the inverse document frequency weight of a term t.

The weighted normalized term frequency is obtained by first performing length normalization, on a term frequency $W_{e,f,t}$ of a term t of field *f* in an element *e*,

$$W_{e,f,t} = \frac{X_{e,f,t}}{1 + B_f\,(\frac{l_{e,f}}{l_f} - 1)}$$

Where B_f is a parameter to tune, $l_{e,f}$ is a length of a field *f* in an element *e*, lf is an average field length then multiplied the normalized term frequency $W_{e,f,t}$ by field weight W_f ,

$$X_{e,t} = \sum_f W_f * W_{e,f,t}$$

3. XML Retrieval Model

3.1 Inverted File Definition

The Zettair search engine has good performance in [8], but this engine only support for document level. We have to converted element level to document level by absolute XPath [9] context replace to <DOCNO> tag in Zettair TREC format then our system is able to use the best features of Zettair [10], when we replace all tag to <DOCNO> by absolute XPath then the Leaf-Only indexing is closest to traditional information retrieval since each XML node is a bag of words of itself, and can be scored as ordinary plain text document then we calculate the leaf element score of its context using BM25 algorithm; For instance, take a document named x1.

```
<?xml version="1.0"?>
<article>
        <title>xml</title>
        <body>
              <section>
              <title>xml</title>
              <p>information</p>
              <p>retrieval</p>
             </section>
        </body>
</article>
```

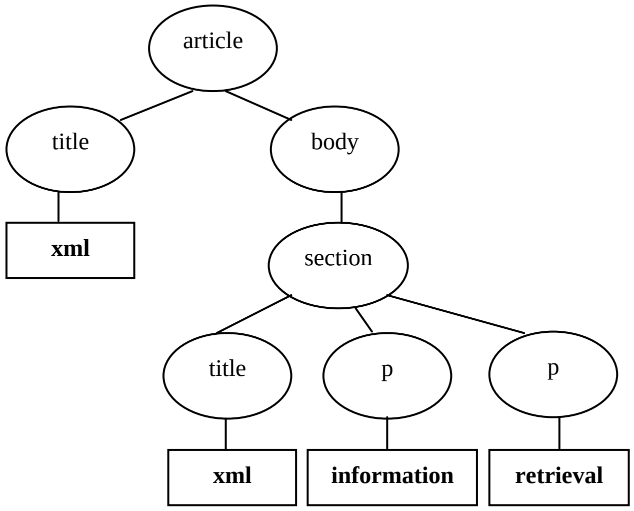


Figure 1. The Example of XML Element Tree

Figure 1 depicts an example of an XML element tree of x1; we can build an index using absolute XPath expression to identify a leaf XML node that has text contained within the document, relative to document and its parents, as follows.

x1/article[1]/title[1]: *xml*
x1/article[1]/body[1]/section[1]/title[1]: *xml*
x1/article[1]/body[1]/section[1]/p[1]: *information*
x1/article[1]/body[1]/section[1]/p[2]: *retrieval*

Finally, term position identifies the ordinal position of the term within the XPath context. The next step to data preprocesses for Zettair search engine application, and then we convert Xpath into TREC format as follow;

<DOC><DOCNO>x1/article[1]/title[1]</DOCNO>xml</DOC>
<DOC><DOCNO>x1/article[1]/body[1]/section[1]/title[1]</DOCNO>xml</DOC>
<DOC><DOCNO>x1/article[1]/body[1]/section[1]/p1[1]</DOCNO>information</DOC>
<DOC><DOCNO>x1/article[1]/body[1]/section[1]/p2[1]</DOCNO>Retrieval</DOC>

3.2 Leaf-Node Scoring Scheme

The Leaf-Only indexing is closest to traditional information retrieval since each XML node is a bag of words of itself, and can be scored as ordinary plain text document then we calculate the leaf element score of its context using BM25 as following;

$$LeafScore(e,Q) = \sum_{t \in Q} W_t * \frac{(k_1 + 1) * tf_e}{k_1 * ((1-b) + b * \frac{len(e)}{avel}) + tf_e}$$

Note that;
LeafScore(e, Q) measures the relevance of element e to a query Q.
W_t is the inverse element frequency weight of term t.
tf_e is the frequency of term t occurring in element e.
len(e) is the length of element e.
avel is the average length of elements in whole collection.
k1 and *b* are used to balance the weight of term frequency and element length.

3.3 Double-Scoring Scheme

On the point of field weight, our approaches reduce parameter tuning step by extended new indices to store all of selected fields, namely Selected Weight (SW). The selected indexing is also closest to traditional information retrieval since each XML node is a bag of words of itself, and can be scored as ordinary plain text document, then we calculate the leaf element score of its context using BM25 as following;

$$SW(e,Q) = \sum_{t \in Q} W_t * \frac{(k_1 + 1) * tf_e}{k_1 * ((1-b) + b * \frac{len(e)}{avel}) + tf_e}$$

After this step, we can compute the element score that using;

$$LeafScore(e,Q) <= \sum_{t \in Q} LeafScore(e,Q) * SW(e,Q)$$

Refer to an example of an XML element tree; we classify tag by manual, then we can build new indices as follows.

SW Indices;
x1/article[1]/title[1]: *xml*
x1/article[1]/body[1]/section[1]/title[1]: *xml*

Leaf-Node Indices;
x1/article[1]/body[1]/section[1]/p[1]: *information*
x1/article[1]/body[1]/section[1]/p[2]: *retrieval*

3.4 Score Sharing Scheme

In previous reports [11], we compute the scores of all elements in the collection that contain query terms. We must consider the scores of elements by accounting for their relevant descendents. The scores of retrieved elements are now shared between leaf node and their parents in the document XML tree according to the following scheme.

$$Score(PNode) \longleftarrow Score(PNode) + [(LeafScore) * \beta^n]$$

Note that;
PNode is a current parent node.
β is tuning parameter.
If [0 – 1], then preference is given to the leaf node over the parents.
Otherwise, preference should be given to the parents.
n is the distance between the current parent node and the leaf node.

4. Experiment Setup

In this section, we present and discuss the results that were obtained at INEX collections. We also present the results of an empirical sensitivity analysis of various β parameters, performed with the Wikipedia collection. This experiment was done on Intel Pentium i5 4 * 2.79 GHz with the memory of 6 GB, Microsoft Windows 7 Ultimate 64-bit Operating System and using Microsoft Visual C#.NET 2008 for develop system.

4.1 INEX Test Collections

The INEX document collections are following;

On Ad hoc track, the Wikipedia XML Corpus of the English Wikipedia in early 2009 [12] that contains 2,666,190 articles and the total size is 50.7 GB.

On Data Centric track, the IMDB data collection newly built from www.imdb.com [13]. It consists of information about more than 1,590,000 movies and people involved in movies and the total size is 1.40 GB.

On Web Service Discovery track, this track will use a collection of WSDL documents. These WSDL documents were directly crawled from real-world public Web services indexed by the Google search engine. The test collection was pre-processed so that only valid WSDL1.1-compliant descriptions are retained for XML-based retrieval that contains 1,987 articles.

At first, the system parses all the structures of each XML document with XML parser and parses all the selective nodes of each XML document. After that, our system uses the Leaf-Only indexing scheme in experiments.

4.2 Experiment Results

4.2.1 Ad Hoc Track

In this section, we tuned parameters using INEX-2008 Ad hoc track evaluation scripts distributed by the INEX organizers. Our tuning approach was such that the sum of all relevance scores are maximized as shown the total number of leaf nodes is 2,500 and the β parameter is set to 0.10, which is used to compute the sharing score. We have used the value is K1 = 1.80 and B = 0.40 for evaluate the sensitivity of the element length in BM25 on both indices.

We practically submitted three runs; I138BM25ESS010, I138BM25ESS015 and I138BM25ESS020. Our results showed that I138BM25ESS010 ranked 78th with MAiP is 0.1296, I138BM25ESS015 ranked 87th with MAiP is 0.1132 and our other run I138BM25ESS020 ranked 91th with MAiP is 0.1057. Table 1 and Figure 2 depict the measured as Focused Retrieval.

Table 1. The Effectiveness of the Focused Task in INEX-Wiki

RUN ID	β	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
I138BM25ESS010	0.10	0.4299	0.4053	0.3695	0.3271	**0.1296**
I138BM25ESS015	0.15	0.4596	0.4394	0.3735	0.3254	**0.1132**
I138BM25ESS020	0.20	0.4960	0.4532	0.3589	0.2765	**0.1057**

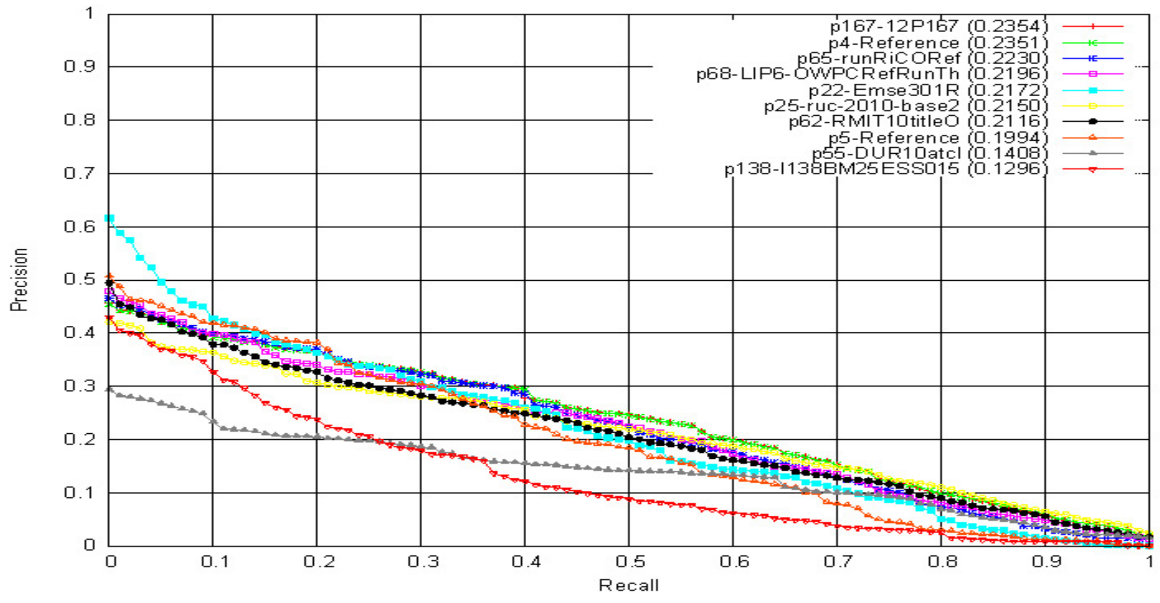


Figure 2. INEX-2010 Ad hoc Result on Focused Retrieval

4.2.2 Data Centric Track

On Data Centric track, we have used the value is K1 = 1.80 and B = 0.40 for evaluate the sensitivity of the element length in BM25 on both indices, the total number of leaf nodes is 2,500 and the β parameter is set to 0.60, which is used to compute the sharing score.

We practically submitted only one run; our result showed that I138BM25ESS060 ranked 4th with MAgP is 0.1811. Table 2 and Figure 3 depict the measured as Focused Retrieval on Data Centric track.

Table 2. The Effectiveness of the Data Centric

RUN ID	β	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
I138BM25ESS060	0.60	0.5821	0.5113	0.3244	0.2719	**0.1211**

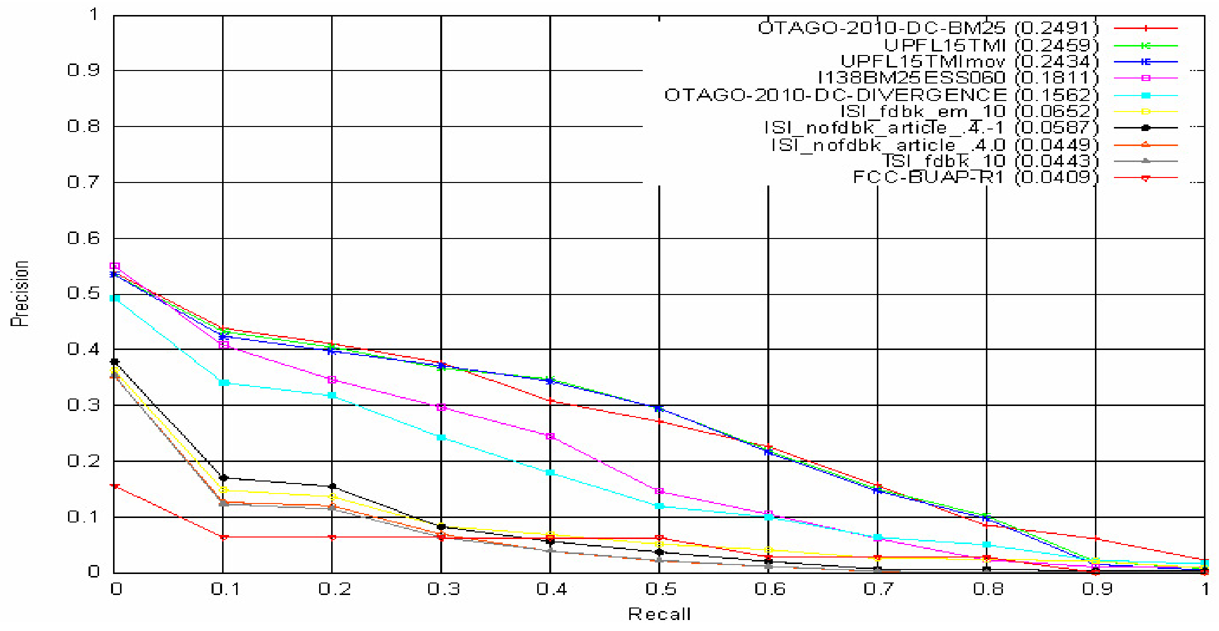


Figure 3. INEX-2010 Data Centric Result on Focused Retrieval

4.2.3 Web Service Discovery Track

On the Web Service Discovery track, these raw name tokens cannot be utilized directly due to various reasons such as the sublanguage patterns, or programming conventions, etc. Therefore, they need to be converted to natural languages before being indexed using IR models. We applied the Capitalization scheme [14] to solve and then we have used the value is K1 = 1.80 and B = 0.40 for evaluate the sensitivity of the element length in BM25 on both indices, the total number of leaf nodes is 2,500 and the β parameter is set to 0.10, 0.15, and 0.25, which is used to compute the sharing score.

We practically submitted three runs; I138BM25ESS010, I138ANYBM25SS015 and I138ANYSS025.

5. Conclusions

The widespread use of XML documents in digital libraries led to the development of IR methods specifically designed for XML collections. Most traditional IR systems are limited to whole document retrieval; however, since XML documents separate content and structure, XML-IR systems are able to retrieve only the relevant portions of documents. Therefore, users who utilize an XML-IR system will potentially receive highly relevant and highly precise material.

In this paper, we report experimental results of our approach using Double-Scoring scheme base on BM25 model for retrieval large-scale XML collection. This strategy can process by using only common parameter on BM25 by extended new indices to store all of selected fields, to reduce the amount of parameters tuned weights for each selected fields of BM25F.

In our future work, we plan to study the sensitivity of the evaluation to the K1 and B parameter and how to make inferences regarding structural aspects based on CAS queries.

6. References

[1] Extensible Markup Language (XML) 1.1 (Second Edition). http://www.w3.org/TR/xml11/

[2] INitiative for the Evaluation of XML Retrieval (INEX). http://www.inex.otago.ac.nz/

[3] N. Craswell, H. Zaragoza, and S. Robertson. Microsoft Cambridge at TREC 14: Enterprise track. In Proceedings of the TREC 14, 2005.

[4] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In Proceedings of CIKM 2004, pages 42-49, 2004.

[5] W. Lu, S. Robertson, A. Macfarlane. 2006. Field-Weighted XML Retrieval Based on BM25. Proceedings of INEX 2005. LNCS. pp. 126-137.

[6] K. Y. Itakura and C. L. A. Clarke. University of Waterloo at INEX 2009: Adhoc, Book, Entity Ranking, and Link-the-Wiki Tracks. In Advances in Focused Retrieval: Seventh International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX-2009), pages 249–259, 2009.

[7] Geva, S. et al,. 2009. Overview of INEX 2009 Ad Hoc Track. The INEX 2009 Workshop Pre-proceeding. Schloss Dagstuhl, Germany, pp. 16-50.

[8] C. Middleton and R. Baeza-Yates. 2009. A Comparison of Open Source Search Engines. http://wrg.upf.edu/WRG/dctos/Middleton-Baeza.pdf.

[9] XML Path Language (XPath) Version 1.0. http://www.w3.org/TR/xpath

[10] Zettair. The Zettair search engine, 2009. http://www.seg.rmit.edu.au/zettair/.

[11] W. Tanakorn and J. Chuleerat, “A Simple Approach to Optimize XML Retrieval,” The 6th International Conference on Next Generation Web Services Practices, Goa, India, November 23-25, 2010.

[12] R. Schenkel, F. Suchanek, and G. Kasneci. YAWN: A semantically annotated Wikipedia XML corpus. In 12. GI-Fachtagung fÄur Datenbanksysteme in Business, Technologie und Web, pages 277-291, 2007.

[13] Information courtesy of The Internet Movie Database http://www.imdb.com

[14] W. Tanakorn, K. Kitti, and J. Chuleerat, “A Simple Approach to Optimize Text Compression’s Performance,” The 4th International Conference on Next Generation Web Services Practices, Seoul, Korea, October 20-22, 2008.

Overview of the INEX 2010 XML Mining Track: Clustering and Classification of XML Documents

Christopher M. De Vries[1], Richi Nayak[1], Sangeetha Kutty[1], Shlomo Geva[1], Andrea Tagarelli[2]

Faculty of Science and Technology,
Queensland University of Technology, Brisbane, Australia[1]

University of Calabria, Italy[2]

chris@de-vries.id.au, {r.nayak, s.kutty, s.geva}@qut.edu.au,
tagarelli@deis.unical.it

Abstract. This report explains the objectives, datasets and evaluation criteria of both the clustering and classification tasks set in the INEX 2010 XML Mining track. The report also describes the approaches and results obtained by participants.

Key words: XML document mining, INEX, Wikipedia, Structure, Content, Clustering, Classification

1 Introduction

The XML Document Mining track was launched for exploring two main ideas: (1) identifying key problems and new challenges of the emerging field of mining semi-structured documents, and (2) studying and assessing the potential of Machine Learning (ML) techniques for dealing with generic ML tasks in the structured domain i.e. classification and clustering of semi-structured documents. This track has run for six editions during INEX 2005, 2006, 2007, 2008, 2009 and 2010. The first five editions have been summarized in [1,2,3,4] and we focus here on the 2010 edition.

INEX 2010 included two tasks in the XML Mining track: (1) unsupervised clustering task and (2) semi-supervised classification task where documents are organized in a graph. The clustering task requires the participants to group the documents into clusters without any knowledge of category labels using an unsupervised learning algorithm. On the other hand, the classification task requires the participants to label the documents in the dataset into known categories using a supervised learning algorithm and a training set. This report gives the details of clustering and classifications tasks.

2 Corpus

Working with XML documents is particularly challenging task for ML and IR. XML documents are defined by their logical structure and content. The current

Wikipedia collection contains structure as document structure such as sections, titles and tables, semantic structure as entities mined by YAWN, and navigation structure as document to document links. In 2008 and 2009 the classification task focused on exploiting the link structure of the Wikipedia and continues to do so this year. The clustering task has continued in the same manner as previous years and uses any available content or structure.

A 146,225 document subset of the INEX XML Wikipedia collection was used as a data set for the clustering and classification tasks. The subset is determined by the reference run used for the ad hoc track. The reference run contains the 1500 highest ranked documents for each of the queries in the ad hoc track. The queries were searched using an implementation of Okapi BM25 in the ANT search engine. Using the reference run reduced the collection from 2,666,190 to 146,225 documents. This is a new approach for selecting the XML Mining subset. In previous years it was selected by choosing documents from Wikipedia portals.

The clustering evaluation uses ad hoc relevance judgments for evaluation and most of the relevant documents are contained in the subset. Table 1 contains details of documents relevant to queries missing from the subset. The reference run contains approximately 90 percent of the relevant documents.

Topic	Relevant	Missing	Topic	Relevant	Missing
2010003	231	24 (10.39%)	2010035	16	3 (18.75%)
2010004	124	29 (23.39%)	2010036	94	0 (0.00%)
2010006	151	20 (13.26%)	2010037	11	0 (0.00%)
2010007	49	6 (12.24%)	2010038	433	8 (1.85%)
2010010	251	6 (2.39%)	2010039	138	0 (0.00%)
2010014	64	7 (10.94%)	2010040	60	3 (5.00%)
2010016	506	72 (14.23%)	2010041	35	0 (0.00%)
2010017	5	0 (0.00%)	2010043	130	11 (8.46%)
2010018	34	0 (0.00%)	2010045	159	60 (37.74%)
2010019	6	0 (0.00%)	2010046	53	0 (0.00%)
2010020	34	0 (0.00%)	2010047	18	0 (0.00%)
2010021	203	28 (13.79%)	2010048	72	11 (15.28%)
2010023	115	31 (26.96%)	2010049	42	6 (14.29%)
2010025	19	0 (0.00%)	2010050	147	5 (3.40%)
2010026	54	5 (9.26%)	2010054	292	42 (14.38%)
2010027	77	4 of (5.19%)	2010056	269	37 (13.75%)
2010030	80	32 (40.00%)	2010057	74	0 (0.00%)
2010031	18	1 (5.56%)	2010061	13	0 (0.00%)
2010032	23	2 (8.70%)	2010068	222	2 (0.90%)
2010033	134	16 (11.94%)	2010069	358	82 (22.91%)
2010034	115	9 (7.83%)			
Total				5451	587 (10.77%)

Table 1. Relevant Documents Missing from the XML Mining Subset

3 Categories

In previous years, document categories have been selected using Wikipedia portals where each portal becomes a category. The drawback of this approach is that it only finds categories for documents related to portals. Last year the categories used for clustering evaluation were produced by YAWN that creates categories based on entities found from the YAGO ontology. These categories are very fine grained and narrow and were found not to be particularly useful.

A new approach extracting categories was taken this year. The Wikipedia categories listed for each document are very similar to the YAGO categories as YAGO contains entities based on Wikipedia information. Both the Wikipedia and YAGO categories are noisy and very fine grained. However, the Wikipedia categories exist in a category graph where there are 24 high level topical categories called the "main topic classifications" [5]. Unfortunately, the category graph is not a hierarchy and contains cycles. Many of the paths from a document to the main topic classifications do not make sense. Additionally, users who add categories to Wikipedia pages often attach them to fine grained categories in the graph. They may not realize what links the internal structure of the graph contains when choosing particular categories. The category graph can be changed over time also changing the original intent of the author. Therefore, categories were extracted by finding the shortest paths through the graph between a document and any of the main topic classifications. This is motivated Occam's Razor where the simplest explanation is often the correct one. Figure 1 illustrates the Wikipedia category graphs and highlights a hypothetical shortest category path for the document Hydrogen.

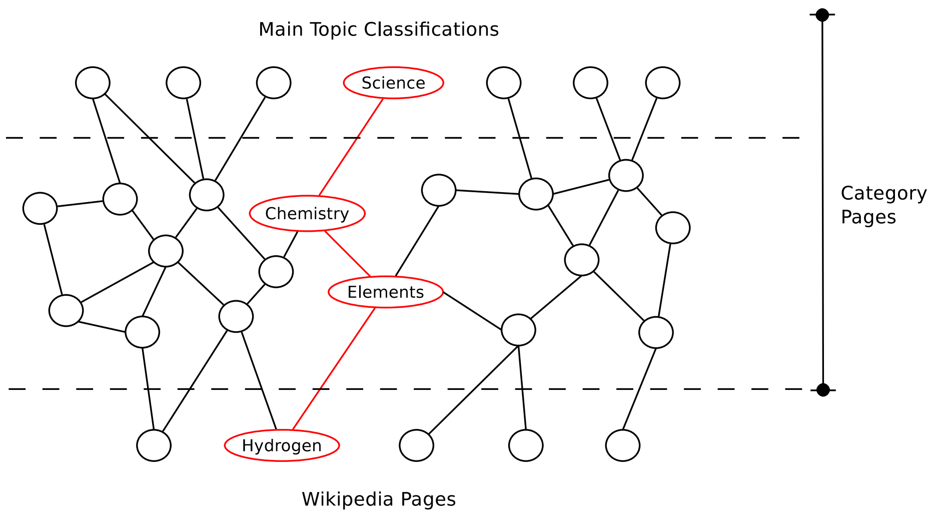


Fig. 1. Complicated and Noisy Wikipedia Category Graph

For INEX 2010 the category graph from the 22nd of June 2010 Wikipedia dump was used. The graph exists of Wikipedia pages with the “Category:” prefix such as “Category:Science”. The graph is extracted by finding links between category pages. Generally speaking, a category page links to another category page that is broader in scope. Wikipedia pages indicate their categories by linking to a category page.

Figure 2 lists the algorithm used to extract the categories. The INEX 2010 categories were extracted where only the 2 broadest levels of categories were extracted ($t = 2$). Only categories containing more than 3000 documents were used. This approach extracts multiple categories for a document resulting in a multi-label set of documents for INEX 2010. Note that paths that contain the “Category:Hidden” category are not used. Table 2 lists the categories that were extracted.

P is the set of Wikipedia pages (articles) to find categories for. C is the set of Categories in the Wikipedia. M the set of categories in the main topic classifications. $G = (V, E)$ is the Wikipedia category graph consisting of a set of vertices V and edges E with pages P attached to it. $P \subset V$. $C \subset V$. $M \subset V$. $M \subset C$. t is a parameter indicating the broadest t levels to consider as categories. If t is 1 then only the main topic classifications are considered. If t is 2 then the main topic classifications and any categories 1 edge away in the graph are considered and so on.

Note that a path is a sequence of graph vertices visited from page $p \in P$ to main topic $m \in M$. For example, Hydrogen $\rightarrow$ Category:Elements $\rightarrow$ Category:Chemistry $\rightarrow$ Category:Science, is the path for the Wikipedia document Hydrogen.

```
ExtractCategories($G, M, P, t$)
1 $E$ = a map from page $p \in P$ to a list of categories for $p$
2 **for** $p \in P$
3     $S$ = the set of shortest paths between $p$ and any category in $M$
4     **for** $s \in S$
5         **if** path $s$ does not contain Category:Hidden
6             $B$ = the set of last $t$ vertices in path $s$
7             **for** $b \in B$
8                 append b to list E[p]
9 **return** $E$
```

Fig. 2. Algorithm to Extract Categories from the Wikipedia

The category extraction process could be enhanced in the future using frequent pattern mining to find interesting repeated sequences in the shortest paths. Other graph algorithms such as the Minimum Spanning Tree algorithm could be used to simplify the graph. The browsable category tree starting at the “main

Category	Documents	Category	Documents
People	48186	Agriculture	5975
Society	34912	Education	4367
Culture	27986	Companies	4314
Geography	22747	Biology	4309
Politics	18519	Recreation	4276
Humanities	14738	Environment	4216
Countries	13966	Musical culture	4195
Arts	11979	Geography stubs	4052
History	10821	Information	3919
Business	10249	American musicians	3845
Applied sciences	9278	Language	3764
Life	9018	Literature	3660
Technology	8920	Belief	3412
Entertainment	8887	Creative works	3395
Nature	7400	Human geography	3370
Science	7311	Places	3202
Computing	6835	Law	3156
Health	6329	Cultural history	3117

Table 2. XML Mining Categories

topic classifications" [5] appears to have processed the category graph as well. Using this post-processed graph could also improve the categories.

4 Clustering Task

The task was to utilize unsupervised machine learning techniques to group the documents into clusters. Participants were asked to submit multiple clustering solutions containing 50, 100, 200, 500 and 1000 clustering.

4.1 Clustering Evaluation Measures

The clustering solutions are evaluated by two means. Firstly, we utilize the categories-to-clusters evaluation which assumes that the categorization of the documents in a sample is known (i.e., each document has known category labels). Any clustering of these documents can be evaluated with respect to this predefined categorization. It is important to note that the category labels are not used in the process of clustering, but only for the purpose of evaluation of the clustering results.

The standard measures of Purity, Entropy, NMI and F1 are used to determine the quality of clusters with regard to the categories. Negentropy [6] is also used. It measures the same system property as Entropy but it is normalized and inverted so scores fall between 0 and 1 where 0 is the worst and 1 is the best. The evaluation measures the mapping of categories-to-clusters where the categories are multi-label but the clusters are not. A document can have multiple categories but documents can only belong to one cluster. Each measure is defined to deal with a multi-label ground truth.

The NCCG measure is defined to calculate the spread of relevant documents from ad hoc queries over clusters. It was defined is last years overview paper [4].

Divergence from Random Most measures of quality for clustering can be tricked by changing the number of clusters or the number of documents assigned to clusters. The Purity and Entropy measures can be fooled if each document is placed in its own cluster. Every cluster becomes pure because it only contains one document. The NCCG measure can be fooled by creating one cluster with all the documents except for every other cluster containing one document. As the NCCG measure orders clusters by the number of relevant documents they contain, the large cluster containing most documents will almost always be ranked first. Therefore, all relevant documents will exist in one cluster, achieving the highest score possible.

Any measure that can be tricked by creating a pathological clustering solution can be adjust for by subtracting a cluster solution from a uniform randomly generated solution with the same number of clusters with the same number of documents in each cluster. Apart from how documents are assigned to clusters, the random baseline appears the same as the real solution. Therefore, each solution needs a uniform random baseline generated. This is done by taking all document IDs, shuffling them uniformly randomly and splitting the document IDs into clusters the same size as the solution being measured. The score for the uniform random solution in subtracted from the matching solution being measured. The graphs and tables in the following section contain the results for all metrics where this approach was taken.

The submissions this year from BUAP contained several large clusters and many other small clusters. This tricked the NCCG metric into giving arbitrarily high scores. When the scores are subtracted from a uniform random baseline with the same properties they performed no better than a randomly generated solution. This can be seen in Figure 6.

4.2 Clustering Participants, Submissions and Evaluations

The clustering tasks had submissions from three participants from Peking University, BUAP and Queensland University of Technology. The submissions labeled Random are a random solution that does not use any information about the documents. A cluster for each document is chosen uniformly at random from one of the k clusters required. Figures 3 to 6 graph the best performing submissions from each participant for Purity, Negentropy, NMI and NCCG. The divergence from random for each metric is also graphed.

The full details of the results are listed in tables in a separate document available from `http://de-vries.id.au/inex10/full_results.pdf`. The tables have been broken into sections matching the required numbers of clusters 50, 100, 200, 500 and 1000. Some submissions were outside the 5 percent tolerance of number of clusters. These form separate groups in the tables.

The group from Peking University made a submission based on the the structured link vector model (SLVM). It incorporated document structure, links and content. This year they focused on the preprocessing step for document structure and links. They modified two popular clustering algorithms, AHC clustering algorithm and K-means algorithm, to work with this model.

The group from BUAP proposed an iterative clustering method for grouping the Wikipedia documents. The recursive clustering process iteratively brings together subsets of the complete collection by using two different clustering methods: k-star and k-means. In each iteration, we select representative items for each group which are then used for the next stage of clustering.

The group from the Queensland University of Technology used a 1024 bit document signature representation generated by quantizing random indexing or random projections of TF-IDF vectors. The k-means algorithm was modified to cluster binary strings of data using the hamming distance, including a different approach to calculating means of binary vectors.

5 Classification Task

The goal of the classification task was to utilize supervised or semi-supervised machine learning techniques to predict categories of documents from a set of known categories described in Section 3. The training set of documents contained 17 percent of the collection where each category had at least 20 percent of the category labels available.

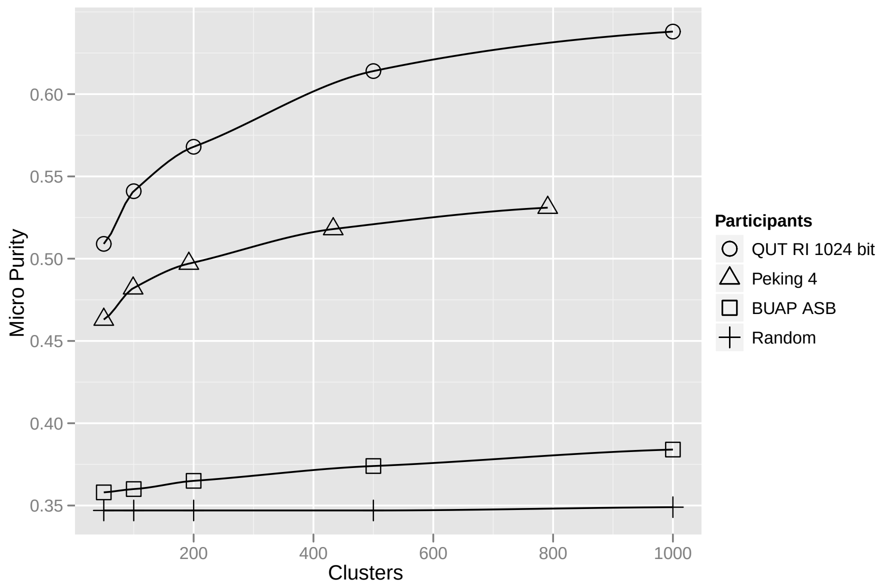
5.1 Classification Evaluation Measures

Classification is evaluated using Type I and II errors made by classifiers. Each category is transformed into a binary classification problem. One category is evaluated at a time using all documents. The scores are calculated based on the Type I and II errors and then micro and macro averaged. Micro averaging weights the average by the category size and macro averaging does not. Table 3 defines the Type I and II errors for a category.

	In Category	Not in Category
Predicted in Category	True Positive (tp)	False Positive (fp)
Predicted not in Category	False Negative (fn)	True Negative (tn)

Table 3. Type I and II Classification Errors

The F1, Precision, Recall, Accuracy and True Negative Rate (TNR) scores are calculated as described in Equations 1 to 5. F1 is the harmonic mean of precision and recall.



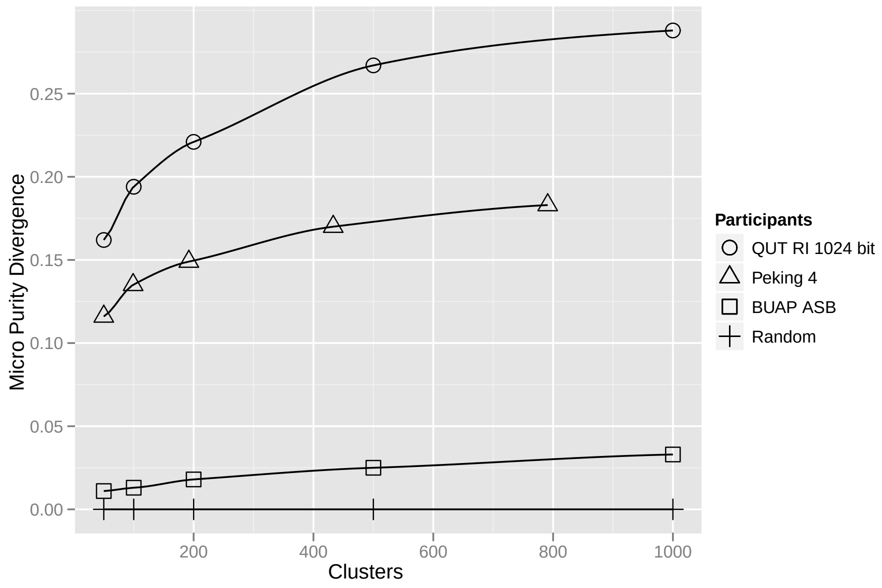
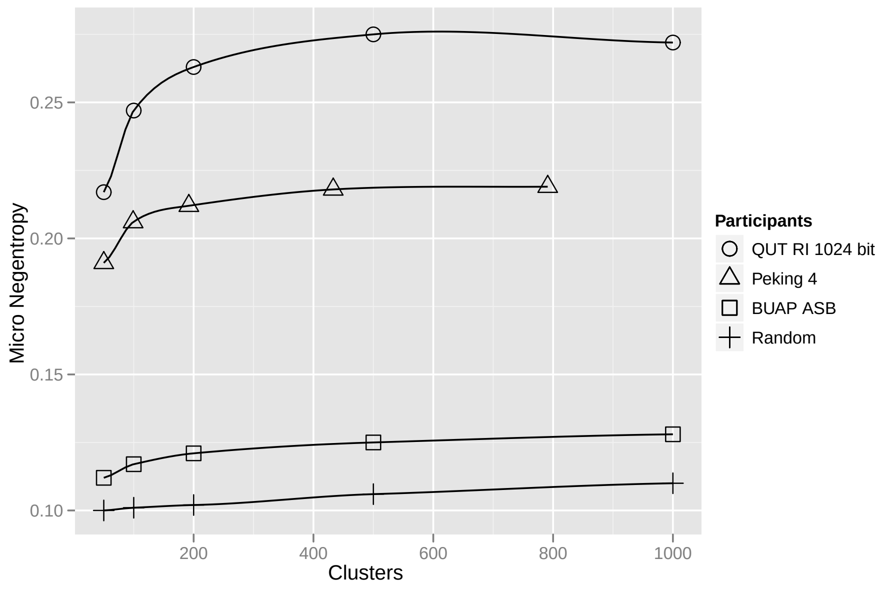


Fig. 3. Micro Purity



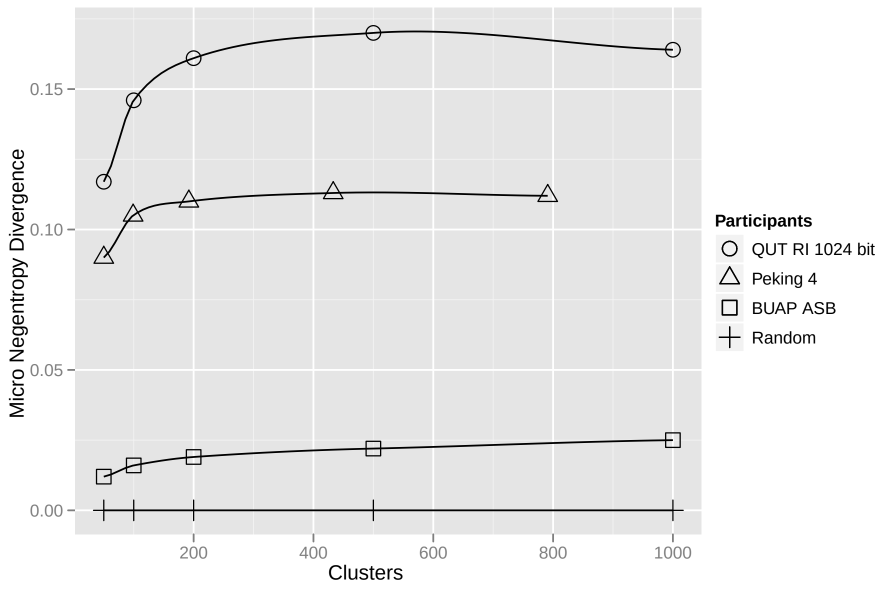


Fig. 4. Micro Negentropy

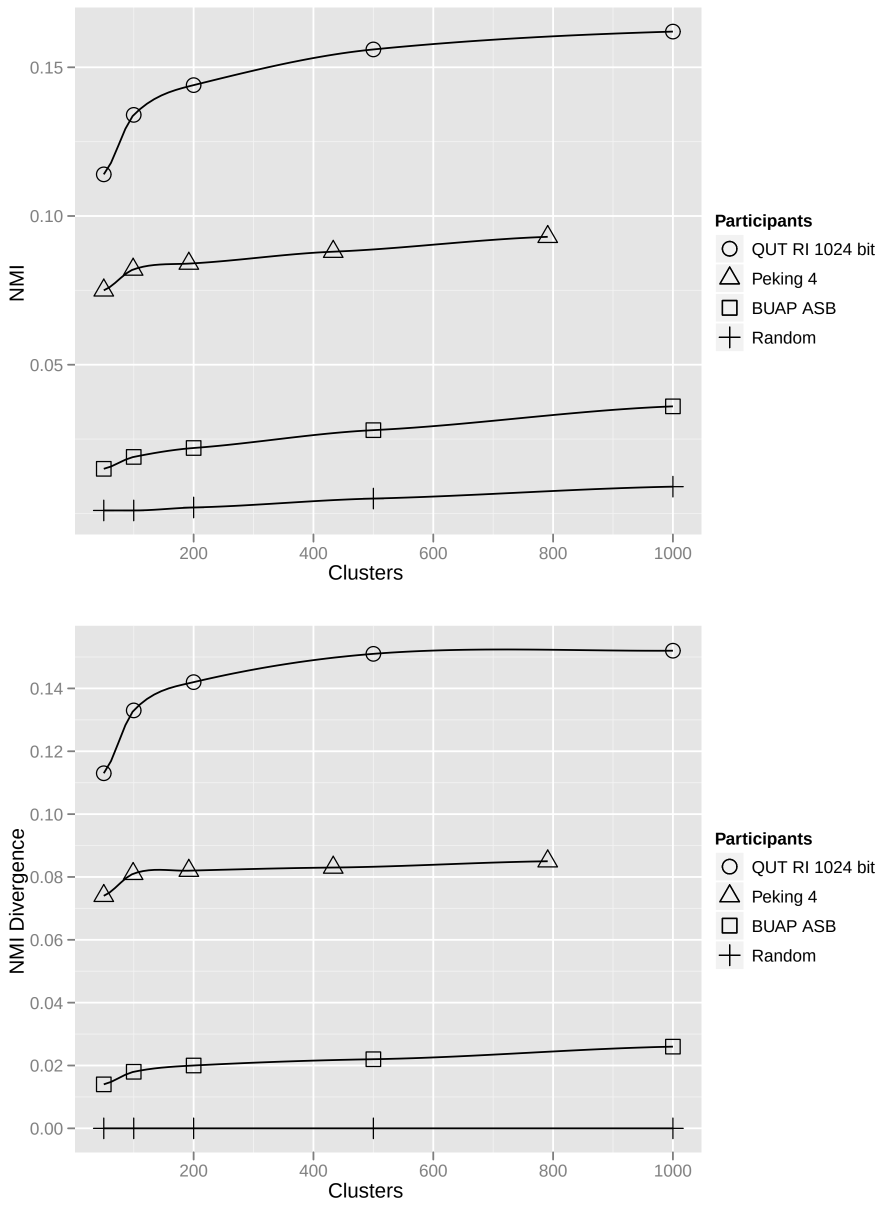
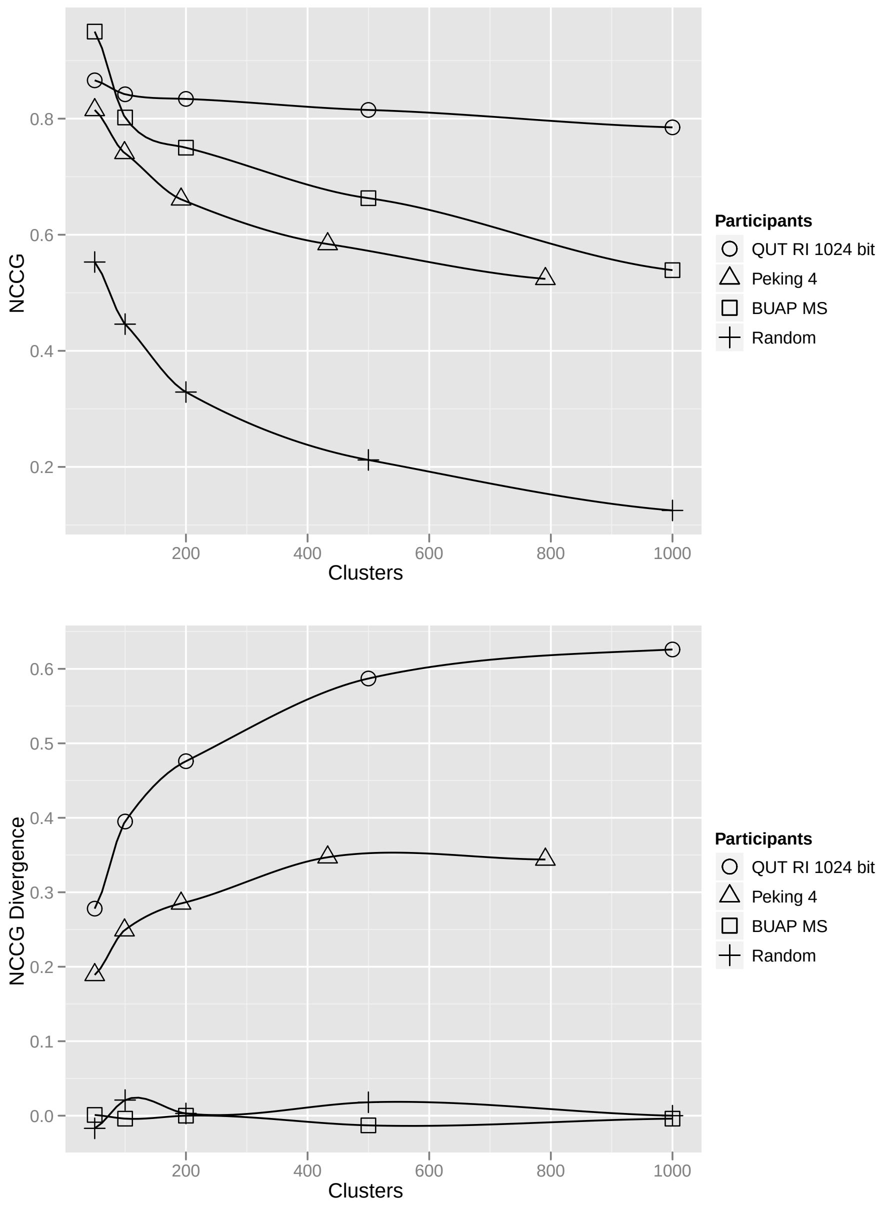

NMI
0.15
0.10
0.05
Clusters
200
400
600
800
1000
Participants
QUT RI 1024 bit
Peking 4
BUAP ASB
Random
NMI Divergence
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Clusters
Participants
QUT RI 1024 bit
Peking 4
BUAP ASB
Random

Fig. 5. NMI



NCCG
Clusters
NCCG Divergence
Clusters
Participants
QUT RI 1024 bit
Peking 4
BUAP MS
Random

Fig. 6. NCCG

$$\mathrm{F1} = \frac{2 \times tp}{2 \times tp + fn + fp} \tag{1}$$

$$\mathrm{Precision} = \frac{tp}{tp + fp} \tag{2}$$

$$\mathrm{Recall} = \frac{tp}{tp + fn} \tag{3}$$

$$\mathrm{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \tag{4}$$

$$\text{True Negative Rate (TNR)} = \frac{tn}{tn + fp} \tag{5}$$

5.2 Classification Participants, Submissions and Evaluations

Two groups from Peking University and the Queensland University of Technology (QUT) made submissions for the classification task. The results are listed in Table 4.

The group from Peking University made a submission based on the the structured link vector model (SLVM). It incorporated document structure, links and content. This year they focused on the preprocessing step for document structure and links.

The group from QUT made a submission using content only to provide a baseline approach. Documents were represented in the bag of words vector space model using the BM25 weighting for each term where the tuning parameters $K1 = 2$ and $b = 0.75$. A Support Vector Machine (SVM) was used to classify each document by treating each category as a binary classification problem.

6 Conclusion

The XML Mining track in INEX 2010 brought together researchers from Information Retrieval, Data Mining, Machine Learning and XML fields. The clustering task allowed participants to evaluate clustering methods against a real use case and with significant volumes of data. The task was designed to facilitate participation with minimal effort by providing not only raw data, but also pre-processed data which can be easily used by existing clustering software. The classification task allowed participant to explore algorithmic, theoretical and practical issues regarding the classification of interdependent XML documents.

	F1		Precision		Recall	
Submission	Micro	Macro	Micro	Macro	Micro	Macro
QUT BM25 SVM	**0.536**	**0.473**	0.562	0.527	0.523	0.440
Peking tree1 sim3 linkTxt 0	0.460	0.380	0.553	0.525	0.436	0.334
Peking tree2 sim3 linkTxt N	0.518	0.446	0.436	0.359	0.652	**0.614**
Peking tree2 sim2 linkTxt 0	0.452	0.371	0.562	0.536	0.423	0.321
Peking tree1 sim1 linkTxt 0	0.399	0.314	**0.582**	**0.570**	0.363	0.252
Peking tree1 sim3 linkTxt 67	0.508	0.435	0.422	0.345	0.653	0.612
Peking tree2 sim2	0.521	0.452	0.480	0.414	0.574	0.510
Peking tree1 sim3	0.518	0.444	0.443	0.368	0.635	0.582
Peking tree1 sim3 linkTxt N	0.517	0.444	0.432	0.356	**0.656**	0.613
Peking tree1 sim2 linkTxt N	0.521	0.454	0.456	0.389	0.615	0.559

	Accuracy		TNR	
Submission	Micro	Macro	Micro	Macro
QUT BM25 SVM	**0.897**	**0.944**	0.932	0.970
Peking tree1 sim3 linkTxt 0	0.891	0.943	0.931	0.974
Peking tree2 sim3 linkTxt N	0.866	0.918	0.882	0.933
Peking tree2 sim2 linkTxt 0	0.892	0.943	**0.934**	0.976
Peking tree1 sim1 linkTxt 0	0.888	0.943	**0.934**	**0.980**
Peking tree1 sim3 linkTxt 67	0.860	0.914	0.874	0.928
Peking tree2 sim2	0.883	0.932	0.910	0.954
Peking tree1 sim3	0.869	0.920	0.886	0.936
Peking tree1 sim3 linkTxt N	0.864	0.917	0.878	0.931
Peking tree1 sim2 linkTxt N	0.873	0.926	0.893	0.944

Table 4. Classification Results

References

1. Denoyer, L., Gallinari, P., Vercoustre, A.: Report on the xml mining track at inex 2005 and inex 2006. Comparative Evaluation of XML Information Retrieval Systems (2007) 432–443
2. Denoyer, L., Gallinari, P.: Report on the XML mining track at INEX 2007 categorization and clustering of XML documents. In: ACM SIGIR Forum. Volume 42., ACM (2008) 22–28
3. Denoyer, L., Gallinari, P.: Overview of the INEX 2008 XML mining track. Advances in Focused Retrieval (2009) 401–411
4. Nayak, R., De Vries, C., Kutty, S., Geva, S., Denoyer, L., Gallinari, P.: Overview of the INEX 2009 XML mining track: Clustering and classification of XML documents. Focused Retrieval and Evaluation (2010) 366–378
5. : Wikipedia main topic classifications, `http://en.wikipedia.org/wiki/Category:Main_topic_classifications`. (2010)
6. De Vries, C., Geva, S.: Document Clustering with K-tree. Advances in Focused Retrieval (2009) 420–431

An Iterative Clustering Method for the XML-Mining task of the INEX 2010

Mireya Tovar[1], Adrián Cruz[2], Blanca Vázquez[3],
David Pinto[1], Darnes Vilariño[1]
{mtovar,dpinto,darnes}@cs.buap.mx, abadrector@gmail.com,
blanca_tec@hotmail.com

[1]Benemérita Universidad Autónoma de Puebla, México
[2]Instituto Tecnológico de Cerro Azul, México
[3]Instituto Tecnológico de Tuxtla Gutiérrez, México

Abstract. In this paper we propose two iterative clustering methods for grouping Wikipedia documents of a given huge collection into clusters. The recursive clustering process clusters iteratively subsets of the complete collection. In each iteration, we select representative items for each group which are then used for the next stage of clustering.
The presented approaches are highly scalable algorithms which may be used with huge collections that in other way (for instance, using the classic clustering methods) would be almost impossible of being clustered. The obtained results outperformed the random baseline presented in the INEX 2010 clustering task of the XML-Mining track.

1 Introduction

The INEX 2010 clustering task was presented with the purpose of being an evaluation forum for providing a platform to measure the performance of clustering methods over a real-world and high-volume Wikipedia collection.

Clustering analysis refers to the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait, often proximity, according to some defined distance measure [1,2,3].

Clustering methods are usually classified with respect to their underlying algorithmic approaches. Hierarchical, iterative (or partitional) and density-based are some possible categories belonging to this taxonomy.

In this paper we report the obtained results when two different approaches for clustering the INEX2010 collection were used. The description of both approaches are given in the following section.

2 Description of the two approaches

In order to be able to cluster high volumes of data, we have approached two clustering techniques by partitioning the complete document collection. The process followed by the so called K-biX/k-biN approach is presented in Figure 1, whereas a scheme of a similar process but in this case using K-Means is shown in Figure 2. An explanation of both approaches follows.

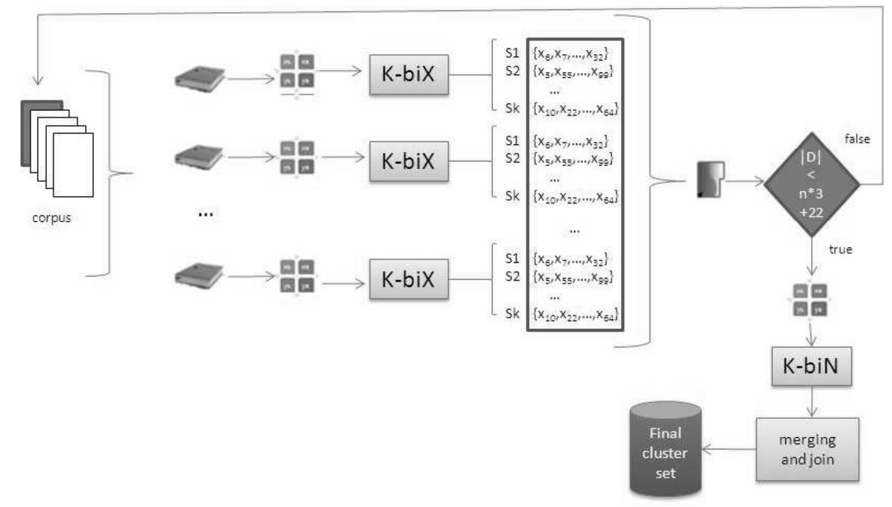


Fig. 1. The K-biX/K-biN approach of BUAP Team at the INEX 2009 clustering task

2.1 The K-biX / K-biN clustering approach

This approach consider two main modules: the K-biX and the K-biN. The former is described first and later we describe the latter.

The K-biX clustering method: This clustering method receives as input a similarity matrix sorted in a non-increasing order. Thereafter, it brings together all those items whose similarity value is greater than a given threshold (in this particular case, we have used the average similarity of the matrix). The procedure of K-biX is given in Algorithm 1.

The presented clustering method (K-biX) is unable to find a fixed number of clusters. Instead it tries to discover the optimal number of clusters. Therefore, we have proposed an additional clustering method for fixing the number of clusters to those required in the competition (50, 100, 200, 500 and 1000). The description of this technique is given as follows.

The K-biN clustering method: The K-biX clustering method considers the clustering with a number fixed of clusters. This number depends of some criterion given in advance, and the similarity degree among the documents processed. The Algorithm 2 presents the K-biN technique.

Algorithm 1: Algorithm K-biX used for clustering the INEX 2010

```
Input: A $n \times n$ similarity matrix $\varphi(d_i, d_j)$ sorted in a non-increasing order, a
       threshold $\epsilon$
Output: A set of clusters $C_1, C_2, ...$
1  $D = \{d_1, d_2, ...\}$;
2  $REG = (|D|^2 - |D|)/2$;
3  $loop = 1$;
4  while ($loop \leq REG$) and ($\varphi(d_i, d_j) \geq \epsilon$) do
5      if ($|d_i| > |d_j|$) then
6          $tmp = d_i$;
7          $d_i = d_j$;
8          $d_j = tmp$;
9      if ($d_i \notin rel$ and $d_j \notin rel$) then
10         $fusion_{d_i} = \{d_i, d_j\}$;
11         $rel_{d_i} = \{d_i\}$;
12         $rel_{d_j} = \{d_i\}$;
13     else if ($d_i \in rel$) and($d_j \notin rel$) then
14         $fusion_{rel_{d_i}} = fusion_{rel_{d_i}} \cup \{d_j\}$;
15         $rel_{d_j} = \{d_i\}$;
16     else if $d_j \in rel$ and ($d_i \notin rel$) then
17         $fusion_{rel_{d_j}} = fusion_{rel_{d_j}} \cup \{d_i\}$;
18         $rel_{d_i} = \{d_j\}$;
19     $loop = loop + 1$;
20 $cluster = 1$;
21 foreach $d_x \in fusion$ do
22     $first\_d\_representative(fusion_{d_x})$;
23     $C_{cluster} = fusion_{d_x}$;
24     $cluster = cluster + 1$;
25 foreach $d_x \in |D|$ do
26     if $d_x \notin rel$ then
27         $C_{cluster} = \{d_x\}$;
28         $cluster = cluster + 1$;
29 return $C_1, C_2, ...$
```

Algorithm 2: Algorithm K-biN used for clustering the INEX 2010

```
Input: A $n \times n$ similarity matrix $\varphi(d_i, d_j)$ sorted in a non-increasing order, $n$
       number of clusters
Output: A set of clusters $C_1, C_2, ...C_n$
1  $D = \{d_1, d_2, ...\}$;
2  $REG = (|D|^2 - |D|)/2$;
3  $gs = n + 1$;
4  $cn = 0$;
5  $loop = 1$;
6  while ($loop \leq REG$) and ($gs > n$) do
7      if ($|d_i| > |d_j|$) then
8          $tmp = d_i$;
9          $d_i = d_j$;
10         $d_j = tmp$;
11     if ($d_i \notin rel$ and $d_j \notin rel$) then
12         $fusion_{d_i} = \{d_i, d_j\}$;
13         $rel_{d_i} = \{d_i\}$;
14         $rel_{d_j} = \{d_i\}$;
15         $cn = cn + 2$;
16     else if ($d_i \in rel$) and($d_j \notin rel$) then
17         $fusion_{rel_{d_i}} = fusion_{rel_{d_i}} \cup \{d_j\}$;
18         $rel_{d_j} = \{d_i\}$;
19         $cn = cn + 1$;
20     else if $d_j \in rel$ and ($d_i \notin rel$) then
21         $fusion_{rel_{d_j}} = fusion_{rel_{d_j}} \cup \{d_i\}$;
22         $rel_{d_i} = \{d_j\}$;
23         $cn = cn + 1$;
24     if $|D| - n \geq cn$ then
25         $f = 0$;
26         $cd = 0$;
27         foreach $d_x \in fusion$ do $f = f + 1$;
28         foreach $d_x \in |D|$ do
29             if $d_x \in rel$ then  $cd = cd + 1$;
30         $gs = f + cd$;
31     $loop = loop + 1$;
32 $cluster = 1$;
33 foreach $d_x \in fusion$ do
34     $first\_d\_representative(fusion_{d_x})$;
35     $C_{cluster} = fusion_{d_x}$;
36     $cluster = cluster + 1$;
37 foreach $d_x \in D$ do
38     if $d_x \notin rel$ then
39         $C_{cluster} = \{d_x\}$;
40         $cluster = cluster + 1$;
41 return $C_1, C_2, ..., C_n$
```

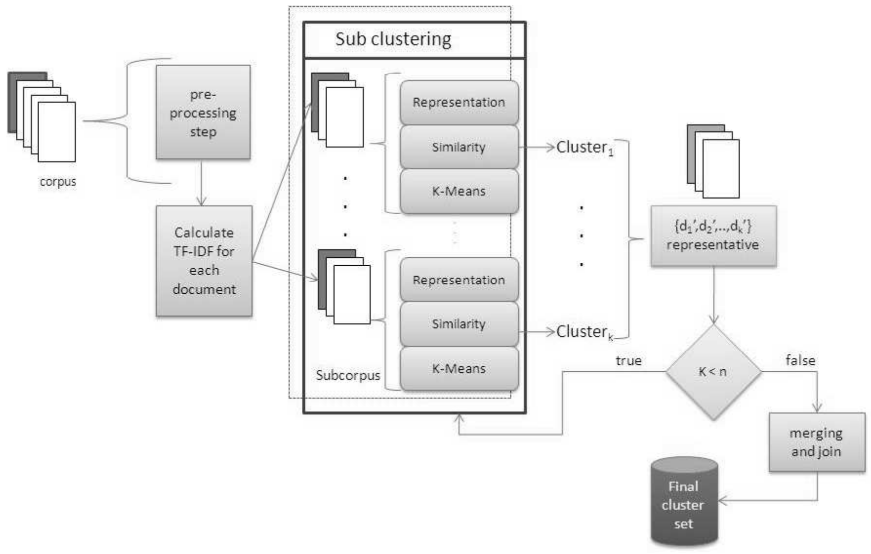


Fig. 2. The K-Means based approach of BUAP Team at the INEX 2009 clustering task

2.2 The K-Means based clustering approach

A second approach have considered to use the widely known K-Means algorithm, which assigns each object to the cluster whose center is nearest. The center is the average of all the points of the cluster. That is, its coordinates are the arithmetic mean for each dimension separately over all the points in the cluster. The K-Means clustering method follows as shown in Algorithm 3 ([3]).

Algorithm 3: Algorithm of the K-Means clustering method

```
  Input: $K$ number of clusters, a similarity matrix $\sigma(d_i, d_j)$
  Output: A set of clusters $\{C_1, C_2, \cdots, C_K\}$
1 Randomly generate $K$ clusters and determine the cluster centers;
2 repeat
3     Assign each point to the nearest cluster center;
4     Recompute the new cluster centers;
5 until some convergence criterion is met (usually that the assignment has not
  changed) ;
6 return $C_1, C_2, \cdots, C_K$
```

The main advantages of this algorithm are its simplicity and speed which allows it to run on large datasets. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments.

The proposed approach using the K-Means clustering method is depicted in Algorithm 4.

Algorithm 4: Algorithm used for clustering the INEX 2010 with K-Means

```
   Input: A document collection $D$, n number of clusters (50, 100, 200, 500 or
          1000 )
   Output: A set of clusters $C_1, C_2, ...C_n$
1  Represent each document according to TF-IDF;
2  Split $D$ into $m$ subsets $D_i$ made of $\frac{|D|}{m}$ documents;
3  foreach $D_i \subset D$ such as $D_i = \{d_{i,1}, d_{i,2}, d_{i,3}, ..., d_{i,\frac{|D|}{m}}\}$ do
4      Calculate the similirity matrix $M_i$ of $D_i$ using the cosine measure;
5      Apply the K-Means clustering method to $M_i$ in order to obtain $k$ clusters;
6      $(\{C_{i,1}, C_{i,2}, ..., C_{i,k}\})$;
7  end
8  $Loop = 1$;
9  while $(Loop \leq MAX\_ITERATIONS)$ do
10     Select a random representative document $d_{i,j}$ for each cluster $C_{i,j}$ obtained;
11     Let $D'$ be the set of documents $d_{i,j}$ i.e., only those that represent each
       cluster obtained;
12     Calculate the similiraty matrix $M_i'$ of $D_i'$ using the cosine measure;
13     Apply the K-Means clustering method to $M_i'$ in order to obtain $n$ clusters
       $(\{C_{i,1}', C_{i,2}', ..., C_{i,n}'\})$;
14     Let $C_{i,j} = C_{i,j} \cup C_{i,j'}$, where $d_i, j \in C_{i,r}'$ and $d_{i,j'} \in C_{i,r}'$ with $1 \leq r \leq k$ and
       $j <> j'$;
15     $Loop = Loop + 1$;
16 end
17 return $C_1, C_2, ...C_n$
```

2.3 Construction of the similarity matrix

The clustering methods aforementioned assume that a matrix that represents the similarity degree among all the documents of the collection is given in advance. In this case, the construction of the similarity matrix was carried out by means of the TF-IDF measure which is described into detail as follows.

The Term Frequency and Inverse Document Frequency (*tf-idf*) is a statistical measure of weight often used in natural language processing to determine how important a term is in a given corpus, by using a vectorial representation. The importance of each term increases proportionally to the number of times this term appears in the document (frequency), but is offset by the frequency of the term in the corpus. In this document, we will refer to the *tf-idf* as the complete similarity process of using the *tf-idf* weight and a special similarity measure proposed by Salton [4] for the Vector Space Model, which is based on the use of the cosine among vectors representing the documents.

The tf component of the formula is calculated by the normalized frequency of the term, whereas the *idf* is obtained by dividing the number of documents in the corpus by the number of documents which contain the term, and then taking the logarithm of that quotient. Given a corpus D and a document d_j ($d_j \in D$), the tf-*idf* value for a term t_i in d_j is obtained by the product between the normalized frequency of the term t_i in the document d_j (tf_{ij}) and the inverse document frequency of the term in the corpus ($idf(t_i)$) as follows:

$$tf_{ij} = \frac{tf(t_i, d_j)}{\sum_{k=1}^{|d_j|} tf(t_k, d_j)} \tag{1}$$

$$idf(t_i) = log\left(\frac{|D|}{|d : t_i \in d, d \in D|}\right) \tag{2}$$

$$tf\text{-}idf = tf_{ij} * idf(t_i) \tag{3}$$

Each document can be represented by a vector where each entry corresponds to the tf-*idf* value obtained by each vocabulary term of the given document. Thus, given two documents in vectorial representation, d_i and d_j, it is possible to calculate the cosine of the angle between these two vectors as follows:

$$\text{Cos}_\theta(\overrightarrow{d_i}, \overrightarrow{d_j}) = \frac{\overrightarrow{d_i} \cdot \overrightarrow{d_j}}{\left\|\overrightarrow{d_i}\right\| \left\|\overrightarrow{d_j}\right\|}$$

The similarity matrix is then constructed on the basis of the above formulae, i.e., for each possible pair of documents, we need to calculate how similar they are by using the cosine measure. Once the similarity matrix is calculated, we may proceed with the clustering step, as described in the previous subsections.

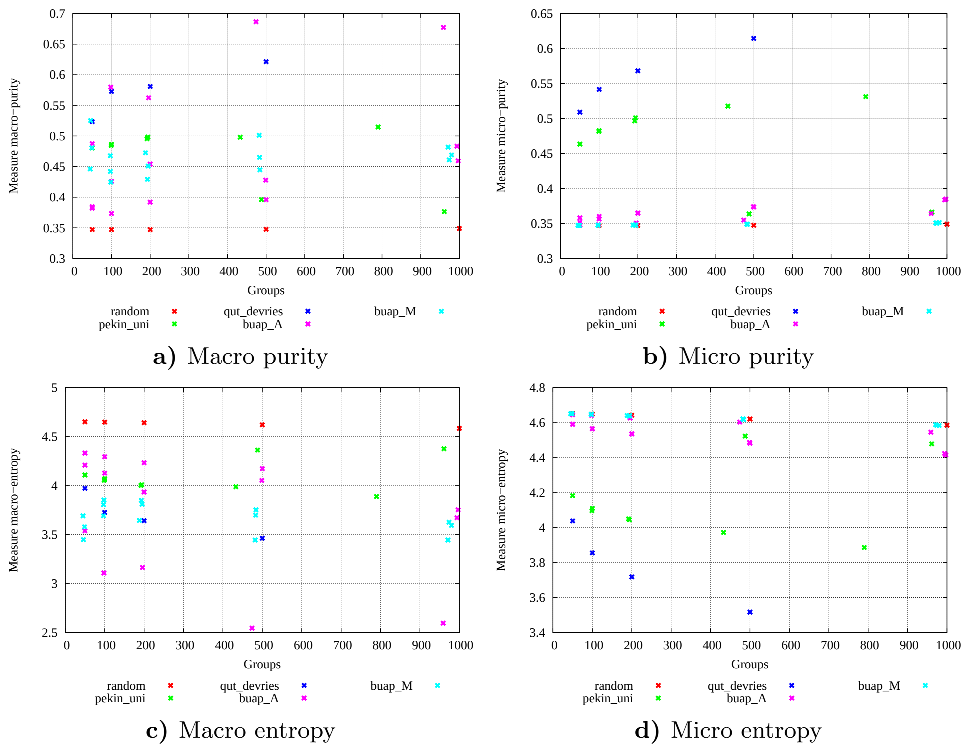
The obtained results are presented and discussed in the following section.

3 Experimental results

The clustering task of INEX 2010 evaluated unsupervised machine learning solutions against the ground truth categories by using standard evaluation criteria such as Purity, Entropy and F-score ($f1$).

Even if the complete description of the dataset used in the clustering task of INEX 2010 is given in the track overview paper, we may describe general features of this corpus.

The clustering task use a 146,225 document subset of INEX 2010 collection that has been pre-processed to provide various representations of the documents such as, vector space representation of terms, frequent bi-grams, XML tags, trees, links and named entities. From these representations, we have used unigrams and frequent bi-grams (original terms and stemmed terms). The obtained results are presented in Figures 3 and 4.



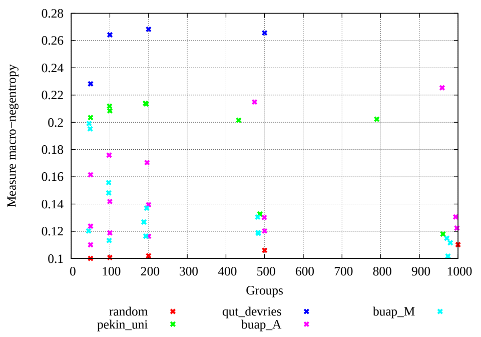
a) Macro purity **b)** Micro purity **c)** Macro entropy **d)** Micro entropy

Fig. 3. Evaluations of the runs with Purity and Entropy (macro and micro) at the INEX 2010 clustering task

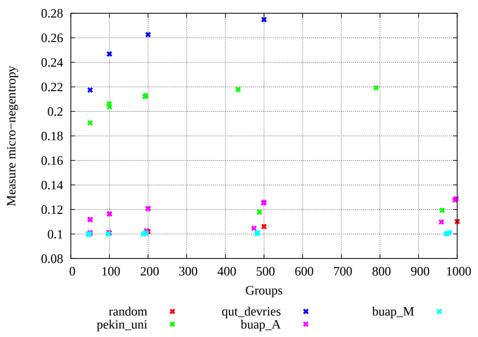
In general, it can be noticed that the presented approaches performs slightly better than the random assignment. Our thought is that we have not iterated the algorithm in order to converge to an optimal clustering. We are considering to repeat the experiments with the gold standard in hand in order to analyze what went wrong with both approaches.

4 Conclusions

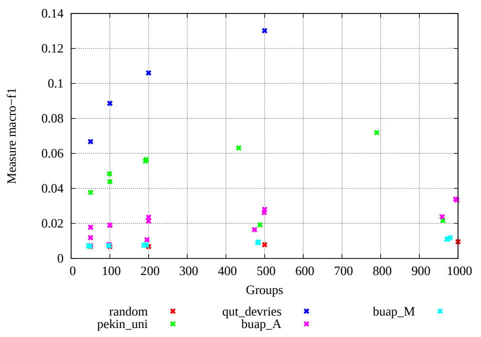
A recursive method based on the K-biX/K-biN and K-Means clustering methods has been proposed in this paper. The aim of the two presented approaches was to allow high scalability of the clustering algorithms. Traditional clustering of huge volumes of data requires to calculate a two dimensional similarity matrix. A process which needs quadratic time complexity with respect to the number of documents. The lower the dimensionality of the similarity matrix, the faster the clustering algorithm will be executed. However, the performance of both approaches were not as expected, because it just slightly improved a baseline



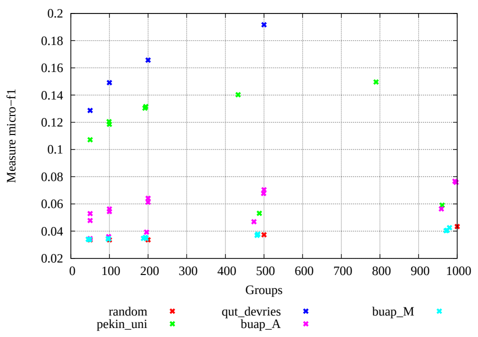
a) Macro negentropy



b) Micro negentropy



c) Macro $f1$



d) Micro $f1$

Fig. 4. Evaluations of the runs with negentropy and $f1$ (macro and micro) at the INEX 2010 clustering task

made up of a random assignment. We would like to analyze this behaviour when the gold standard is released by the task organizers.

5 Acknowledgments

This work has been partially supported by CONACYT and PROMEP projects.

References

1. MacKay, D.J.C.: Information Theory, Inference and Learning Algorithms. Cambridge University Press (2003)
2. Mirkin, B.G.: Mathematical Classification and Clustering. Springer (1996)
3. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press (1967) 281–297
4. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Communications of the ACM **18**(11) (1975) 613–620

PKU at INEX 2010 XML Mining Track

Songlin Wang, Feng Liang, and Jianwu Yang

Institute of Computer Sci. & Tech., Peking University,
Beijing 100871, China
{wangsonglin, liangfeng, yangjianwu }@icst.pku.edu.cn

Abstract. This paper presents our participation in the INEX 2010 XML Mining track. Our classification and clustering solutions for XML documents have used both the structure and content information, where the frequent subtrees as structural units are used for content extraction from the XML document. In addition, we use the WordNet and the link information for better performance. We have experimented the tuning of various parameters using the collection of INEX 2009 rather than the collection used for INEX 2010, and we apply the structured link vector model for classification.

Keywords: XML Document, Classification, Clustering, Frequent Subtree, Structured Link Vector Model (SLVM).

1 Introduction

The two tasks in INEX 2010's XML mining track are categorization and clustering. The clustering task is an unsupervised process through which all the documents can be classified into clusters and the problem is to find meaningful clusters without any prior information. The categorization task is a supervised task for which, given a set of categories, a training set of reclassified documents is provided. Using this training set, the task keep on learning the categories description in order to be able to classify a new document into one or more categories. And in this XML mining tack, the corpus is a subset of the Wikipedia corpus with 144,625 documents that belong to 36 categories resulting in a multi-label classification task.

In this paper, the documents are represented according to the structured link vector model (SLVM) [1], which was extended from the conventional vector space model (VSM) [2], and used the closed frequent subtrees as structural units. More precisely, we focus on the preprocessing step, particularly the feature selection, frequent subtrees selection, the process of link information, and these steps can be essential for improving the performance of the categorization and clustering.

Moreover, 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 need to extract text from a subset of terms that can be used to represent the documents in view of their categorization efficiently. In this year, a file with a list of links between XML documents has been given, and we will show that those links add relevant information for the categorization of documents.

This paper is organized as follows. In section 2, document representation is discussed. In section 3, we show some parameters tuning in the corpus of INEX2009. Section 4 describes the approach taken to the classification task and the associated results. In section 5, we review the clustering task and discuss the results. The paper ends with a discussion of future research and conclusion in section 6.

2 System overview

2.1 Document model for categorization(SLVM)

The Vector space model (VSM) [2] has been widely used for representing text documents as vectors which contain terms weights. Given a collection D of documents, a document $\mathrm{d_x}$ of D is represented by a vector $\mathrm{d_x} = (\mathrm{d_{(x,1)}}, \mathrm{d_{(x,2)}}, \mathrm{d_{(x,3)}} \dots \dots \mathrm{d_{(x,n)}})$, where $\mathrm{d_{(x,i)}}$ is the weight of the term $\mathrm{t_i}$ in the document $\mathrm{d_x}$.

SLVM is extended from VSM. The basic idea is to extract structure units from XML documents, then extract content information from each structure units. The content information of each structure units is regarded as a VSM model. Every XML document is modeled as a matrix in SLVM [1].

So, the document $\mathrm{d_x}$ can be represented by a document feature matrix $\chi \in R^{m\times n}$,

$$\mathrm{d_x} = (\mathrm{d_{x(1)}}, \mathrm{d_{x(2)}}, \mathrm{d_{x(3)}} \dots \dots \mathrm{d_{x(m)}}) \quad \textbf{(1)}$$

$$\mathrm{d_{x(i)}} = (\mathrm{d_{x(i,1)}}, \mathrm{d_{x(i,2)}}, \mathrm{d_{x(i,3)}} \dots \dots \mathrm{d_{x(i,n)}}) \quad \textbf{(2)}$$

Where m is the number of unit of document $\mathrm{d_x}$, and $\mathrm{d_{x(i,j)}}$ is the weight of the term $\mathrm{t_j}$ in the unit $\mathrm{d_{x(i)}}$ of document $\mathrm{d_x}$. SLVM combines both structure information and content information. In order to calculate the weight of the terms, TFIDF and BM25 formula can be used.

2.2 BM25 formula

Our system is based on the BM25 weights function [3].

$$\mathrm{BM25}(\mathrm{q},\mathrm{d}) = \mathrm{TF}(q,d) \times \mathrm{IDF}(q) \quad (3)$$

$$\mathrm{TF}(\mathrm{q},\mathrm{d}) = \frac{f(q,d) \times (\mathrm{k}_1 + 1)}{f(q,d) + \mathrm{k}_1 \times \left(1 - \mathrm{b} + \mathrm{b} \times \frac{\mathrm{d}}{\mathrm{D}_{\mathrm{avg}}}\right)} \quad (4)$$

$$\mathrm{IDF}(\mathrm{q}) = \log \frac{N - n(q) + 0.5}{n(q) + 0.5} \quad (5)$$

With:
- $f(q,d)$: the frequency of term q in article d.
-N: the number of articles in the collection.
-n(q): the number of articles containing the term q.
-$\frac{\mathrm{d}}{\mathrm{D}_{\mathrm{avg}}}$: the ratio between the length of articles d and the average article length.
-k_1 and b: the classical BM25 parameters.

Parameter k_1 is able to control the term frequency saturation. Parameter b allows setting the importance of $\frac{\mathrm{d}}{\mathrm{D}_{\mathrm{avg}}}$.

2.3 Chi-square feature selection

The main idea of the feature selection is to choose a subset of input variables by eliminating features with little or no predictive information, and the feature selection can significantly improve the comprehensibility of the resulting classifier models. The Chi-square test is a commonly used method, which evaluates the features individually by measuring their chi-square statistic with respect to the classes.

3 Parameters tuning (INEX 2009)

We have used the train collection and test collection of INEX 2009 XML Mining Track for this experiment. The collection is composed of about 54,889 XML documents of the Wikipedia XML Corpus, and this subset of Wikipedia represents 39 categories, each corresponding to one subject or topic. The training set with the category annotations is composed of 11,028 elements, which is about 20% of the whole collection. On the training set, the mean of the number of category by document is 1.46 and 9809 documents belong to only one category.

First we list all the features encountered in the documents of the collection, about one million features are built for all of the documents. Then the chi-square test has been used to select the features, and we find about 2,000 features for each categorization can acquire a good value.

The INEX 2009 tuning results show in table1, where all the value with the BM25 have been improved, so our system is based in the BM25 weights function for this year task.

Table 1. The result of BM25

	Ma_acc	Mi_acc	Ma_roc	Mi_roc	Ma_prf	Mi_prf	Map
TFIDF	0.966	0.953	0.855	0.863	0.496	0.543	0.709
BM25	0.967	0.952	0.932	0.933	0.537	0.579	0.745

3.1 Pruning the tree

In this paper, we utilize the frequent subtrees as structural units to extract the content information from the XML documents, and a series of pre-processing have been done for the subtrees.

As the last year, we use the closed frequent subtrees as structural units. The documents of INEX 2009 is more complex than the collection of ever before, so we employ some pruning strategies for mining closed frequent subtrees and use the chi-square test to select a part of subtrees as useful structural. We obtain the document vectors at three different pruning levels, 0, 1/3, and 1/2, which yield a better performance for almost all the evaluation measures.

Table 2 presents the effectiveness comparison of classification structures at various pruning levels, we choice 10 subtrees for each category, and uses no more than 40,000 features for all categories. We only use 35,000 features when the WordNet is used, as calculation of the similarity between the words is very slowly.

Table 2. The results of pruning the document tree

	Ma_acc	Mi_acc	Ma_roc	Mi_roc	Ma_prf	Mi_prf	Map
0	0.956	0.940	0.879	0.861	0.428	0.438	0.606
1/3	0.961	0.944	0.896	0.872	0.464	0.465	0.626
1/2	0.961	0.942	0.876	0.873	0.447	0.475	0.633

In addition, we find that the average of subtrees for each XML document has increased, which is useful for the classification, when some pruning strategies are used for mining closed frequent subtrees.

3.2 WordNet

In some documents, there are only a few words, which can't express the information of the categories completely. So we use the WordNet to extend the information of the documents.

Given the closed frequent subtrees $T(t_1, t_2, t_3 \dots t_s)$, the features $W(w_1, w_2, w_3 \dots w_m)$, and two documents D_i, D_j that contain the tree t_k of T, and suppose their parts of the t_k are $D_{i_{tk}}(w_1', w_2', w_3' \dots w_m')$ and $D_{j_{tk}}(w_1'', w_2'', w_3'' \dots w_n'')$.

The similarity between the t_k parts of the documents $D_i$and D_j given as:

$$S_{ij_{tk}} = \sum_{a=1}^{\mathrm{Max}\{m,n\}} w_a' * w_a'' \qquad (6)$$

After using the WordNet, we can get the similarity matrix between the words, and the similarity between the t_k parts of the documents given as:

$$S_{ij_{tk}} = \sum_{a=1}^{m}\sum_{b=1}^{n} S_{w_a' w_b''} w_a' * w_b'' \qquad (7)$$

Where $S_{w_a' w_b''}$ is the similar between the word w_a' and w_b''.

Given, the similarity matrix between the words, we can change the $D_{j_{tk}}$ as follow:

$$D_{j_{tk}}(w_1'' + \sum_a^m S_{w_1'' w_a'}, w_2'' + \sum_a^m S_{w_2'' w_a'}, w_3'' + \sum_a^m S_{w_3'' w_a'} \dots w_n'' + \sum_a^m S_{w_n'' w_a'}) \qquad \textbf{(8)}$$

When $D_{j_{tk}}(w_1'', w_2'', w_3'' \dots w_n'')$ have all features of $W(w_1, w_2, w_3 \dots w_m)$, this method is equal to (7), however no document can contain all of the selected features, and in this way we can increase the feature number of the document, especially the words that don't appear in the document, which is important to some small documents for classification and clustering, it equal to use the similarity matrix between the words to do smoothing. For example, when the features of $D_{j_{tk}}$ don't contain word A, but it contain word B and word C, which the similarity with A is greater than 0, and then we can use $S_{w_A w_B} + S_{w_A w_C}$ to smoothing the weight of word A in $D_{j_{tk}}$.

Definition: $S_{w_a w_b}$ means the similarity between the two words w_a and w_b, if $S_{w_a w_b} > \gamma$, where γ is the parameter of similarity, w_a and w_b are synonym, we can merge the word w_a and w_b together.

3.3 Using the link information to improve the result

In this section, we show that the classification accuracy can be improved based on word features and out-linked features. And the XML documents of INEX can provide much richer information to classifiers for classification. Links between the documents can be used for this task. In this year a file with a list of links between XML documents has been given, and we will show that those links add relevant information for the categorization of documents. As how to use the link information, we have tried three ways:

3.3.1 Change the document structural

After a series of pre-processing for the documents, we can change the documents in the following format:

```
<article>
    <content_information> ……</ content _information>
    <frq_subtrees>……</frq_subtrees>
</article>
```

In the above table, the text of the document is included by the tag of "<content_information>", "<frq_subtrees>" contain the frequent subtrees of the document, which is the XML structure of the documents. As well, in order to join the link information of the document, we should add another tag, which is named "<link_informatiom>", and the table can be changed as follow:

```
<article>
    <content_information> ……</ content _information>
    <frq_subtrees>……</frq_subtrees>
    <link_information>……</link_information>
</article>
```

3.3.2 Change the vector of the document

The closed frequent subtrees $\mathrm{T}(\mathrm{t}_1, \mathrm{t}_2, \mathrm{t}_3 \ldots \mathrm{t}_s)$, and a given document D , which contain the tree t_k of T, and suppose the part of the t_k is

$$\mathrm{D}_{\mathrm{tk}} \quad (\mathrm{w}_1', \mathrm{w}_2', \mathrm{w}_3' \ldots \mathrm{w}_m') \tag{9}$$

Where m is the total number of features appear in the tree t_k part of the document collection, w_i' is the weight of term i in document D and $0 \le \mathrm{w}_i' \le 1$. Hence, if $\mathrm{w}_i' > 0$,it means that word i exists in the tree t_k part of the document D. Therefore, document D will be represented by a vector of features

$$\mathrm{D}_{\mathrm{tk}} \quad (\mathrm{w}_1', \mathrm{w}_2', \mathrm{w}_3' \ldots \mathrm{w}_m', \mathrm{l}_1', \mathrm{l}_2', \mathrm{l}_3' \ldots \mathrm{l}_n') \tag{10}$$

Where n is the total number of links appear in the tree t_k part of the document, l_i' is the weight of link l_i' in document D and $0 \le \mathrm{l}_i' \le 1$. Hence, if $\mathrm{l}_i' > 0$,it means that link l_i' exists in the tree t_k part of the document D.

3.3.3 Using the link for tuning

Build SVM model base on the link information, get the correlativity between the documents and categories, and improve the correlativity of documents and categories that calculation form the text information:

$$S(c|d) = \alpha \times S_l(c|d) + (1-\alpha) S_t(c|d) \tag{11}$$

Where $S(c|d)$ is the final correlativity between the document d and the category c; $S_l(c|d)$ is the correlativity between the document d and the category c by using the link information only; and $S_t(c|d)$ is the correlativity between the document d and the category c by using the text information only; parameter α allows setting the importance of $S_l(c|d)$.

In this experiment, we use 80000 features for the text content, and use BM25 for the text representation. The SVM based text classification results are listed in Table 3. In the following tables, we can notice that, out-linked features work very well in the experiment, and this simply means that add in out-link features can improve the performance of classification.

Table 3. The result of using the link for tuning

	Ma_acc	Mi_acc	Ma_roc	Mi_roc	Ma_prf	Mi_prf	Map
α=0	0.967	0.952	0.932	0.932	0.537	0.579	0.746
Using link for tuning	0.968	0.956	0.912	0.916	0.570	0.608	0.780

3.4 Compare the results

Table 4. Compare the result with INEX 2009

Team	Micro F1	Macro F2	APR
University of Wollongong+	51.2	47.9	68
University of Peking+	51.8	48	70.2
XEROX Research center+	60	57.1	67.8
University of Saint Etienne+	56.4	53	68.5
University of Granada+	26.2	25.3	72.9
Our result*	**60.9**	**57.1**	**78.0**

+ Result of INEX09; * the wordNet is not used.

4 The clustering methods

AHC Clustering Algorithm and K-means Algorithm are two popular clustering algorithms in statistics and machine learning, but these methods do not work well in the massive data sets. So we modify these algorithms, considering the structure of the document and the massive data set. We have approached the simple countering technique based on batch of the complete document collection; the process followed is present in Figure 1.

```
• Given
• X: a set of N vectors
• K:desired number of clusters
• Begin
    • Select K random seeds $\{\overrightarrow{s_1}, \overrightarrow{s_2}, \dots \dots, \overrightarrow{s_k}\}$ from X
    • Each time load part of the documents, and computer the
      distance between the document and clusters
    • Do
        • Do all of the document
            • Load a batch of documents
            • Assign each document to cluster
              which is the minimal distance, and calculate
              the cluster for next time.
        • End_do
        • Calculate the clusters and the distance between
          the clusters and document.
    • End_do (if the gap of distance between two consecutive
      cycles exceeds the given threshold.)
    • Load the documents and assign each document to
      clusters.
    • Print_result.
• Done
```

Figure 1. An algorithm for clustering

Another algorithm is based on AHC Clustering Algorithm, and the process of the Local AHC Clustering Algorithm is present as Figure 2.

```
• Let D be the whole complete document collection, split D into subset $D_i$
  containing 1000 documents, and for each $D_i$, we do the follow process, after we
  combine the result ,and continue, until the number of clusters meet our
  requirement:
    • Given:
    • Subsets $D_i$ and a threshold $\tau$
    • Calculate similarity matrix M
    • Do  (if max element in M > threshold $\tau$)
        • Find the max element of M: $m_{i,j}$ and constructs a cluster
          $C_i$ made up of the document $d_i$ and $d_j$ , it marks these
          documents as assigned.
        • Do  all unassigned document $d_k$
```

```
            • If $m_{i,k} > \tau$
                • add $d_k$ to cluster $C_i$ and mark $d_k$ assigned
        • End_do
        • Update row i and column j, set $m_{k,j} = m_{i,k} = avg(m_{c,k})$, where $d_c \in C_i$.
        • Update row c and column c, set $m_{k,c} = m_{c,k} = -1$, where $d_c \in C_i$.
    • End_do
```

Figure 2. The Local AHC Clustering Algorithm

5 INEX 2010 results

We present in this section the official results obtained by our system during INEX2010 on the new INEX 2010 collection, we submitted 9 runs, and all of our submit results are based on the BM25, reference run given by the INEX organizers.

1.1. System settings:

Most of the settings given in section 3 have been reused for our INEX2010 runs, except:

- BM25 parameters k_1 = 2.0 and b = 0.75;
- For each category, we set the frequent tree number is 10, and the threshold of support is 200;
-To improve the result of classification, we used the link information, and α we set 0.40 and 0.46, which is obtained from tuning in section 3;
-The threshold for each category, we tuning from the training set, and we get three groups.

1.2. Results

Tables 5 present the official results of classification:

Table 5. The results of classification

Team	Run	Mi_p	Ma_p	Mi_r	Ma_r	Mi_t	Ma_t	Mi_a	Ma_a	Mi_f	Ma_f
Peking	T1_S3_link0	0.553	0.525	0.436	0.334	0.931	0.974	0.891	0.942	0.518	0.446
	T1_S3_link67	0.422	0.345	0.653	0.612	0.874	0.928	0.860	0.914	0.508	0.435
	T1_S3	0.433	0.368	0.635	0.582	0.886	0.936	0.869	0.920	0.518	0.444
	T1_S3_linkN	0.432	0.356	**0.656**	**0.613**	0.878	0.931	0.864	0.917	0.517	0.444
	T1_S2_linkN	0.456	0.389	0.615	0.559	0.893	0.944	0.873	0.926	0.522	0.454
	T1_S1_link0	**0.582**	**0.570**	0.363	0.252	**0.934**	**0.980**	0.888	0.943	0.400	0.320
	T2_S2_link0	0.562	0.536	0.422	0.321	0.934	0.976	0.892	0.943	0.452	0.372
	T2_S2	0.48.	0.414	0.574	0.510	0.910	0.954	0.883	0.932	0.521	0.452
	T2_S3_linkN	0.436	0.359	0.652	0.614	0.882	0.933	0.886	0.918	0.518	0.446
Qut	Inex10_sub	0.561	0.527	0.523	0.440	0.932	0.970	**0.896**	**0.944**	**0.536**	**0.473**

Firstly, for all of the runs , we use 80,000 features for all categories, and frequent subtrees have select two groups: one is T1, in which the trees can be a subtree of another; another is T2, in which the frequent subtree cannot be a subtree of another. We also use different thresholds for each categories, S1, S2, S3. And we use the link information to improve the result of the castigation, the run 1, 2, 4,5,6,7 and 9 have use, but the run 3 and 8 have not. In addition, in order to save time, we just use the WordNet to calculate the similarity matrix between the words that select form INEX2009 collection, so there is a certain deviation between the words.

Secondly, from the result table, we can obtain some conclusions. The threshold of each category has played an important rule, and the results including the information are not as good as last year. We suspect that this might be caused by the fact that we tuned the parameters in the collection of INEX 2009, which were not suitable in this year’s data. For this year, each document belong to the number of categories is more than last year, and the standard evaluation of classification has changed, thus, future experiments should do this parameters adjustment. In the part of frequent subtrees we use two extreme cases, T1 and T2, and the result are not so much difference as we expected, though they have a different effect on the INEX2009 data set, so this part needs to do further research experiments.

Tables 6 present the official results of clustering.

Table 6. The results of clustering

	Runs	Mi_p	Ma_e	Mi_e	Ma_n	Mi_n	Ma_f1	Mi_f1	Nmi	M_NCCG	S_NCCG
Pek	Result_1000_4	0.531	3.89	3.89	0.202	0.219	0.072	0.150	0.093	0.524	0.220
	Result_500_4	0.518	3.99	3.97	0.201	0.218	0.063	0.140	0.088	0.584	0.235
	Result_200_4	0.467	4.00	4.05	0.314	0.212	0.056	0.130	0.084	0.661	0.219
	result502_0.15	0.364	4.36	4.52	0.133	0.118	0.019	0.053	0.020	0.391	0.221
	Result_100_4	0.481	4.06	4.10	0.212	0.206	0.048	0.120	0.082	0.741	0.193
	result_0.18_1004	0.336	4.38	4.48	0.118	0.119	0.022	0.060	0.024	0.290	0.169
	Result_50_4	0.463	4.11	4.18	0.203	0.191	0.038	0.101	0.075	0.185	0.157
	Result_100_3	0.483	4.07	4.11	0.209	0.204	0.040	0.119	0.081	0.745	0.190
	Result_200_2	0.501	4.01	4.00	0.213	0.213	0.057	0.132	0.085	0.662	0.212

In table 6, the runs 4 and 6 we use the Local AHC Clustering Algorithm, and the rest we used k-means based on batch of the complete document collection.

6 Conclusions

In this paper, we applied SLVM to XML documents classification and clustering, and we used the frequent subtrees as structural units, both the structure and content information have been used, especially the information of link, which performance well, and in order to extend the information of features, we used the WordNet to smooth the weight of the words in the documents.

It is not so easy to reuse setting of parameters tuned on the different collection. In the experiments of INEX2009, we show that use link information can improve the result of classification by 2%, but it not do so at this year, so we have to experiment more deeply on 2010 collection. Furthermore, 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. Some of those experiments will probably be include in the final version of this paper although, as we exposed in the introduction, we not consider that approach to be very realistic.

Acknowledgment

The work reported in this paper was supported by the National Natural science Foundation of China Grant 60642001 and 60875033.

References

1. Yang,j., Chen,x.: A semi_structured document model for text mining. In: Journal of computer secience and Technology, 17(5) 603-610 (202)
2. Salton G, and McGill MJ.: Introduction to Moder infomrmation Retrieval, McGraw-Hill, 1983.
3. S.E.Robertson and K.Spark Jones.: Relevance wighting of search terms. JASIST, 27(3):129-146, 1976
4. Chi, Y., Nijssen, S., Muntz, R.R, Kok, J.N.: Fequent Subtree Mining – An Overview. Foundamenta Information, 2005
5. Chi,Y., Yang Y., Xia Y., Muntz, R.R.: CMTreeMiner: Mining Both Closed and MaxImal Frequent Subtrees. The Eighth Pacific Asia Conference on Knowledge Discover and Data Mining, 2004
6. Xie, W., Manmadov, M., Yearwood, J.: Using Links to Aid Web Classification. In: ICIS 2007(2007) 0-7695-2841-4/07

Author Index

ISBN 978-90-814485-4-3
90000 >
9 789081 448543