

Shlomo Geva
Jaap Kamps
Ralf Schenkel
Andrew Trotman (Eds.)

LNCS 6932

Comparative Evaluation of Focused Retrieval

9th International Workshop of the Initiative
for the Evaluation of XML Retrieval, INEX 2010
Vught, The Netherlands, December 2010
Revised Selected Papers

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Shlomo Geva Jaap Kamps Ralf Schenkel
Andrew Trotman (Eds.)

Comparative Evaluation of Focused Retrieval

9th International Workshop of the Initiative
for the Evaluation of XML Retrieval, INEX 2010
Vught, The Netherlands, December 13-15, 2010
Revised Selected Papers

Volume Editors

Shlomo Geva
Queensland University of Technology
Faculty of Science and Technology
Brisbane QLD 4001, Australia
E-mail: s.geva@qut.edu.au

Jaap Kamps
University of Amsterdam
Archives and Information Studies/Humanities
1012 XT Amsterdam, The Netherlands
E-mail: kamps@uva.nl

Ralf Schenkel
Saarland University
Multimodal Computing and Interaction
66123 Saarbrücken, Germany
E-mail: schenkel@mnci.uni-saarland.de

Andrew Trotman
University of Otago
Department of Computer Science
Dunedin 9054, New Zealand
E-mail: andrew@cs.otago.ac.nz

ISSN 0302-9743
ISBN 978-3-642-23576-4
DOI 10.1007/978-3-642-23577-1

e-ISSN 1611-3349
e-ISBN 978-3-642-23577-1

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011936359

CR Subject Classification (1998): H.3, H.3.3-4, H.2.8, H.2.3, H.2.4, E.1

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

© Springer-Verlag Berlin Heidelberg 2011

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

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

Traditional IR focuses on pure text retrieval over “bags of words,” but the use of structure—such as document structure, semantic metadata, entities, or genre/topical structure—is of increasing importance on the Web and in professional search. INEX has been pioneering the use of structure for focused retrieval since 2002, by providing large test collections of structured documents, uniform evaluation measures, and a forum for organizations to compare their results.

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

Ad Hoc Track: investigated 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: investigated techniques to support users in reading, searching, and navigating full texts of digitized books, by constructing reading lists of books for a given topic, or by looking for book pages that refute or confirm a factual statement.

Data-Centric Track: investigated focused retrieval over a strongly structured collection of IMDb documents, containing information about various entities like movies, actors, directors.

Interactive Track: investigated the behavior of users when interacting with XML documents, working on large set of Amazon book data (formal book descriptions) augmented by LibraryThing data (user-generated data).

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

Question Answering Track: investigated real-world focused information needs formulated as natural language questions, using the collection structure to construct readable summaries of question context and lists of answers.

Relevance Feedback Track: investigated the utility of incremental passage level feedback by simulating a searcher’s interaction, with submissions in the form of a executable computer program.

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

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

The aim of the INEX 2010 workshop was to bring together researchers who participated in the INEX 2010 campaign. During the year participating organizations contributed to the building of a large-scale XML test collection by creating topics, performing retrieval runs, and providing relevance assessments.

The workshop concluded the results of this large-scale effort, summarized and addressed issues encountered, and devised a work plan for the future evaluation of XML retrieval systems. The proceedings report on the final results of INEX 2010. We received 42 submissions, already being a selection of work at INEX, and accepted a total of 37 papers based on peer-reviewing, yielding an 88% acceptance rate.

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).

After many years at *Schloss Dagstuhl*, and a trip to Brisbane, Australia, in 2009, the INEX workshop returned to Europe and was held in The Netherlands. Thanks to the Amsterdam team for preserving the unique atmosphere of INEX—a setting where informal interaction and discussion occur naturally and frequently—in the unique location of *Huize Bergen* in Vught.

We thank the Dutch Association for Information Science (Werkgemeenschap Informatiewetenschap, WGI) for sponsoring the best student award, which was presented to Ning Gao (Peking University) for the paper entitled “Combining Strategy for XML Retrieval.”

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

May 2011

Shlomo Geva
Jaap Kamps
Ralf Schenkel
Andrew Trotman

Organization

Steering Committee

Charles L. A. Clarke	University of Waterloo, Canada
Norbert Fuhr	University of Duisburg-Essen, Germany
Shlomo Geva	Queensland University of Technology, Australia
Jaap Kamps	University of Amsterdam, The Netherlands
Mounia Lalmas	Yahoo Research Barcelona, Spain
Stephen Robertson	Microsoft Research Cambridge, UK
Andrew Trotman	University of Otago, New Zealand
Arjen P. de Vries	CWI, The Netherlands
Ellen Voorhees	NIST, USA

Chairs

Shlomo Geva	Queensland University of Technology, Australia
Jaap Kamps	University of Amsterdam, The Netherlands
Ralf Schenkel	Max-Planck-Institut für Informatik, Germany
Andrew Trotman	University of Otago, New Zealand

Track Organizers

Ad Hoc

Paavo Arvola	University of Tampere, Finland
Shlomo Geva	Queensland University of Technology, Australia
Jaap Kamps	University of Amsterdam, The Netherlands
Ralf Schenkel	Max-Planck-Institut für Informatik, Germany
Andrew Trotman	University of Otago, New Zealand

Book

Antoine Doucet	University of Caen, France
Gabriella Kazai	Microsoft Research Cambridge, UK
Marijn Koolen	University of Amsterdam, The Netherlands
Monica Landoni	University of Lugano, Switzerland

Data-Centric

Qiuyue Wang	Renmin University, China
Andrew Trotman	University of Otago, New Zealand

Interactive (iTrack)

Thomas Beckers	University of Duisburg-Essen, Germany
Norbert Fuhr	University of Duisburg-Essen, Germany
Ragnar Nordlie	Oslo University College, Norway
Nils Pharo	Oslo University College, Norway

Link-the-Wiki

Shlomo Geva	Queensland University of Technology, Australia
Andrew Trotman	University of Otago, New Zealand

Question Answering (QA)

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

Relevance Feedback

Timothy Chappell	Queensland University of Technology, Australia
Shlomo Geva	Queensland University of Technology, Australia

Web Service Discovery

James Thom	RMIT University, Australia
Chen Wu	Curtin University of Technology, Australia

XML-Mining

Chris De Vries	Queensland University of Technology, Australia
Sangeetha Kutty	Queensland University of Technology, Australia
Richi Nayak	Queensland University of Technology, Australia
Andrea Tagarelli	University of Calabria, Italy

Table of Contents

Ad Hoc Track

Overview of the INEX 2010 Ad Hoc Track	1
<i>Paavo Arvola, Shlomo Geva, Jaap Kamps, Ralf Schenkel, Andrew Trotman, and Johanna Vainio</i>	
The Potential Benefit of Focused Retrieval in Relevant-in-Context Task	33
<i>Paavo Arvola and Johanna Vainio</i>	
ENSM-SE and UJM at INEX 2010: Scoring with Proximity and Tag Weights.	44
<i>Michel Beigbeder, Mathias Géry, Christine Largeron, and Howard Seck</i>	
LIP6 at INEX'10: OWPC for Ad Hoc track	54
<i>David Buffoni, Nicolas Usunier, and Patrick Gallinari</i>	
A Useful Method for Producing Competitive Ad Hoc Task Results	63
<i>Carolyn J. Crouch, Donald B. Crouch, Sandeep Vadlamudi, Ramakrishna Cherukuri, and Abhijeet Mahule</i>	
Relaxed Global Term Weights for XML Element Search.	71
<i>Atsushi Keyaki, Kenji Hatano, and Jun Miyazaki</i>	
Searching the Wikipedia with Public Online Search Engines	82
<i>Miro Lehtonen</i>	
Extended Language Models for XML Element Retrieval.	89
<i>Rongmei Li and Theo van der Weide</i>	

Book Track

Overview of the INEX 2010 Book Track: Scaling Up the Evaluation Using Crowdsourcing	98
<i>Gabriella Kazai, Marijn Koolen, Jaap Kamps, Antoine Doucet, and Monica Landoni</i>	
LIA at INEX 2010 Book Track	118
<i>Romain Deveaud, Florian Boudin, and Patrice Bellot</i>	
The Book Structure Extraction Competition with the Resurgence Software for Part and Chapter Detection at Caen University	128
<i>Emmanuel Giguët and Nadine Lucas</i>	

Focus and Element Length for Book and Wikipedia Retrieval 140
Jaap Kamps and Marijn Koolen

Combining Page Scores for XML Book Retrieval 154
Ray R. Larson

OUC's Participation in the 2010 INEX Book Track 164
Michael Preminger and Ragnar Nordlie

Data Centric Track

Overview of the INEX 2010 Data Centric Track 171
Andrew Trotman and Qiuyue Wang

DCU and ISI@INEX 2010: Adhoc and Data-Centric Tracks 182
Debasis Ganguly, Johannes Leveling, Gareth J.F. Jones, Sauparna Palchowdhury, Sukomal Pal, and Mandar Mitra

Automatically Generating Structured Queries in XML Keyword Search 194
Felipe da C. Hummel, Altigran S. da Silva, Mirella M. Moro, and Alberto H.F. Laender

UPF at INEX 2010: Towards Query-Type Based Focused Retrieval 206
Georgina Ramírez

BUAP: A First Approach to the Data-Centric Track of INEX 2010 219
Darnes Vilariño, David Pinto, Carlos Balderas, Mireya Tovar, and Saul León

Interactive Track

Overview of the INEX 2010 Interactive Track 227
Nils Pharo, Thomas Beckers, Ragnar Nordlie, and Norbert Fuhr

Using Eye-Tracking for the Evaluation of Interactive Information Retrieval 236
Thomas Beckers and Dennis Korbar

Link the Wiki Track

Overview of the INEX 2010 Link the Wiki Track 241
Andrew Trotman, David Alexander, and Shlomo Geva

University of Otago at INEX 2010 250
Xiang-Fei Jia, David Alexander, Vaughn Wood, and Andrew Trotman

Question Answering Track

Overview of the INEX 2010 Question Answering Track (QA@INEX) ...	269
<i>Eric SanJuan, Patrice Bellot, Veronique Moriceau, and Xavier Tannier</i>	
The GIL Summarizers: Experiments in the Track QA@INEX'10	282
<i>Edmundo-Pavel Soriano-Morales, Alfonso Medina-Urrea, Gerardo Sierra Martínez, and Carlos-Francisco Méndez-Cruz</i>	
The Cortex Automatic Summarization System at the QA@INEX Track 2010	290
<i>Juan-Manuel Torres-Moreno and Michel Gagnon</i>	
The REG Summarization System with Question Reformulation at QA@INEX Track 2010	295
<i>Jorge Vivaldi, Iria da Cunha, and Javier Ramírez</i>	

Relevance Feedback Track

Overview of the INEX 2010 Focused Relevance Feedback Track	303
<i>Timothy Chappell and Shlomo Geva</i>	
Exploring Accumulative Query Expansion for Relevance Feedback	313
<i>Debasis Ganguly, Johannes Leveling, and Gareth J.F. Jones</i>	
Combining Strategies for XML Retrieval	319
<i>Ning Gao, Zhi-Hong Deng, Jia-Jian Jiang, Sheng-Long Lv, and Hang Yu</i>	

Web Service Discovery Track

Overview of the INEX 2010 Web Service Discovery Track	332
<i>James A. Thom and Chen Wu</i>	
Semantics-Based Web Service Discovery Using Information Retrieval Techniques	336
<i>Jun Hou, Jinglan Zhang, Richi Nayak, and Aishwarya Bose</i>	
The BUAP Participation at the Web Service Discovery Track of INEX 2010	347
<i>María Josefa Somodevilla, Beatriz Beltrán, David Pinto, Darnes Vilariño, and José Cruz Aaron</i>	
XML Retrieval More Efficient Using Double Scoring Scheme	351
<i>Tanakorn Wichaiwong and Chuleerat Jaruskulchai</i>	

XML Mining Track

Overview of the INEX 2010 XML Mining Track: Clustering and Classification of XML Documents	363
<i>Christopher M. De Vries, Richi Nayak, Sangeetha Kutty, Shlomo Geva, and Andrea Tagarelli</i>	
An Iterative Clustering Method for the XML-Mining Task of the INEX 2010	377
<i>Mireya Tovar, Adrián Cruz, Blanca Vázquez, David Pinto, Darnes Vilarriño, and Azucena Montes</i>	
PKU at INEX 2010 XML Mining Track	383
<i>Songlin Wang, Feng Liang, and Jianwu Yang</i>	
Author Index	397

Overview of the INEX 2010 Ad Hoc Track

Paavo Arvola¹, Shlomo Geva², Jaap Kamps³,
Ralf Schenkel⁴, Andrew Trotman⁵, and Johanna Vainio¹

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. This 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, retrieving 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, suggesting 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 [13].

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 choose 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 IO9UniXRun1 /article[1]/bdy[1]/sec[1]
1 Q0 9996 2 0.9998 IO9UniXRun1 /article[1]/bdy[1]/sec[2]
1 Q0 9996 3 0.9997 IO9UniXRun1 /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 IO9UniXRun1 465 3426
1 Q0 9996 2 0.9998 IO9UniXRun1 3892 960
1 Q0 9996 3 0.9997 IO9UniXRun1 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 [10] 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:

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

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, 9].

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 [12].

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 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.

```

<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)

Each topic contains

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

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

Phrasetitle. A more verbose explanation of the information need given as a series of phrases, just as the (title) is given as a series of keywords.

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

Narrative. A detailed explanation of the information need and the description of what makes an element relevant or not. The (narrative) should explain

```

<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

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.

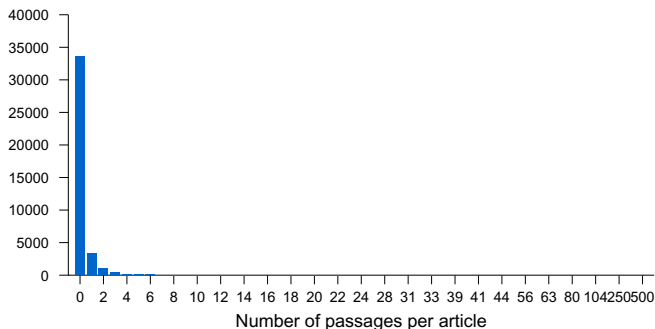
The `<castitle>` contains the CAS query, an XPath expressions of the form: `A[B]` or `A[B]C[D]` where `A` and `C` are navigational XPath expressions using only the descendant axis. `B` and `D` are predicates using functions for text; the arithmetic operators `<`, `<=`, `>`, and `>=` for numbers; or the 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

Table 1. Statistics over judged and relevant articles per topic

	total		# per topic				
	topics	number	min	max	median	mean	st.dev
judged articles	52	39,031	735	757	751	750.6	4.2
articles with relevance	52	5,471	5	506	65	105.2	112.8
highlighted passages	52	13,154	5	4,343	111	253.0	625.6
highlighted characters	52	17,641,119	3,841	2,624,502	129,440	339,252.3	527,349.0

**Fig. 3.** Distribution of passages over articles

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 there 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.

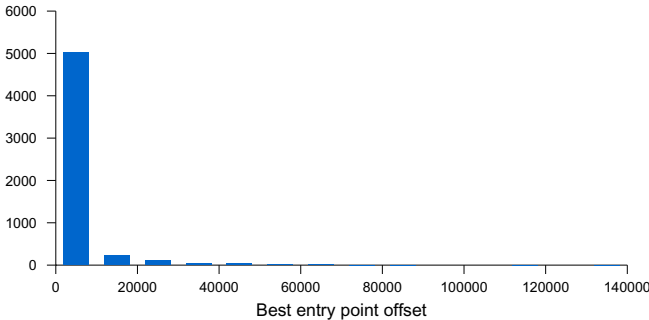
- The 52 assessed topics were numbered 2010 n with n : 003, 004, 006, 007, 010, 014, 016–021, 023, 025–027, 030–041, 043, 045–050, 054, 056, 057, 061, 068–070, 072, 075, 079, 095–097, 100, and 105–107.

Table 1 presents statistics of the number of judged and relevant articles, and passages. In total 39,031 articles were judged. Relevant passages were found in 5,471 articles. The mean number of relevant articles per topic is 105, but the distribution is skewed with a median of 65. There were 13,154 highlighted passages. The mean was 253 passages and the median was 111 passages per topic.

Figure 3 presents the number of articles with the given number of passages. The vast majority of relevant articles (3,388 out of 5,471) had only a single highlighted passage, and the number of passages quickly tapers off.

Table 2. Statistics over relevant articles

	total		# per relevant article				
	topics	number	min	max	median	mean	st.dev
best entry point offset	52	5,471	2	130,618	665	3,166.1	7,944.9
first relevant character offset	52	5,471	2	90,258	525	2,622.2	6,850.0
length relevant documents	52	5,471	249	179,200	5,545	12,084.9	17,274.5
relevant characters	52	5,471	4	179,166	897	3,224.5	7,326.1
fraction highlighted text	52	5,471	0.00036	1.000	0.239	0.358	0.332

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

Assessors were requested to provide a separate best entry point (BEP) judgment, for every article where they highlighted relevant text. Table 2 presents statistics on the best entry point offset, on the first highlighted or relevant character, and on the fraction of highlighted text in relevant articles. We first look at the BEPs. The mean BEP is well within the article with 3,166 but the distribution is very skewed with a median BEP offset of only 665. Figure 4 shows the distribution of the character offsets of the 5,471 best entry points. It is clear that the overwhelming majority of BEPs is at the beginning of the article.

The statistics of the first highlighted or relevant character (FRC) in Table 2 give very similar numbers as the BEP offsets: the mean offset of the first relevant character is 2,662 but the median offset is only 525. This suggests a relation between the BEP offset and the FRC offset. Figure 5 shows a scatter plot the BEP and FRC offsets. Two observations present themselves. First, there is a clear diagonal where the BEP is positioned exactly at the first highlighted character in the article. Second, there is also a vertical line at BEP offset zero, indicating a tendency to put the BEP at the start of the article even when the relevant text appears later on.

Table 2 also shows statistics on the length of relevant articles. Many articles are relatively short with a median length of 5,545 characters, the mean length is 12,085 characters. The length of highlighted text in characters has a median of 897 (mean length is 3,225). Table 2 also show that amount of relevant text varies from almost nothing to almost everything. The mean fraction is 0.36, and the median is 0.24, indicating that typically one-third of the article is relevant. Given

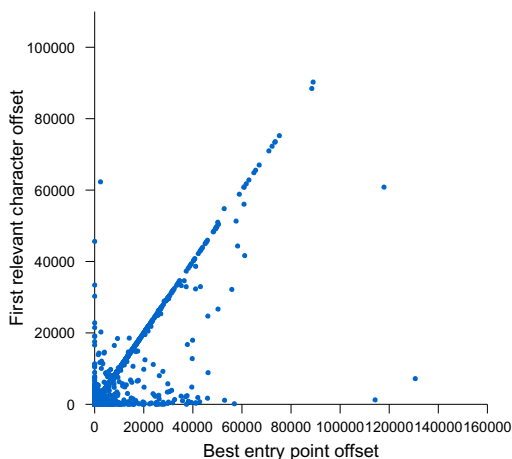


Fig. 5. Scatter plot of best entry point offsets versus the first relevant character

that the majority of relevant articles contain such a large fraction of relevant text plausibly explains that BEPs being frequently positioned on or near the start of the article.

3.4 Questionnaires

At INEX 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 3) featured 20 questions capturing contextual data on the search request. The post-assessment questionnaire (shown in Table 4) featured 14 questions capturing further contextual data on the search request, and the way the topic has been judged (a few questions on GPXrai were added to the end).

The responses to the questionnaires show a considerable variation over topics and topic authors in terms of topic familiarity; the type of information requested; the expected results; the interpretation of structural information in the search request; the meaning of a highlighted passage; and the meaning of best entry points. There is a need for further analysis of the contextual data of the topics in relation to the results of the INEX 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 3. Candidate Topic Questionnaire

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

Table 4. Post Assessment Questionnaire

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

Table 5. Participants in the Ad Hoc Track

Id Participant	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 5 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,

Table 6. 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-runRiCOfRef	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

we allowed sets of runs with 15, 150, 1,500 results per topic. The submissions 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 6 shows the top 10 participating groups (only the best run per group is shown) in the Relevant in Context Task. The first column lists the participant, see Table 5 for the full name of group. The second to fifth column list generalized precision at 5, 10, 25, 50 retrieved articles. The sixth column lists mean average generalized precision.

Here we briefly summarize the information available about the experiments conducted by the top 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 modeled by triangular functions. For the Run Emse303R, the height of the triangle was enlarged proportionally to a weight learnt with the 2009 queries and assessments [5]. 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 `jsectionj` 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 7 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 5 for the full name of group. The second to fifth column list generalized precision at 5, 10, 25, 50 retrieved articles. The sixth column lists mean average generalized precision.

Here we briefly summarize the information available about the experiments conducted by the top 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 7. 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 8. 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-UPFP LM45co	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 8 shows the best run of the top 10 participating groups. The first column gives the participant, see Table 5 for the full name of group. The second 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 9. 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 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 9 shows the best run of the participating groups. The first column gives the participant, see Table 5 for the full name of group. The second to fifth column give the interpolated precision at 0%, 1%, 5%, and 10% recall. The sixth column gives mean average interpolated precision over 101 standard recall levels (0%, 1%, ..., 100%).

Here we briefly summarize what is currently known about the experiments conducted by the top 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.

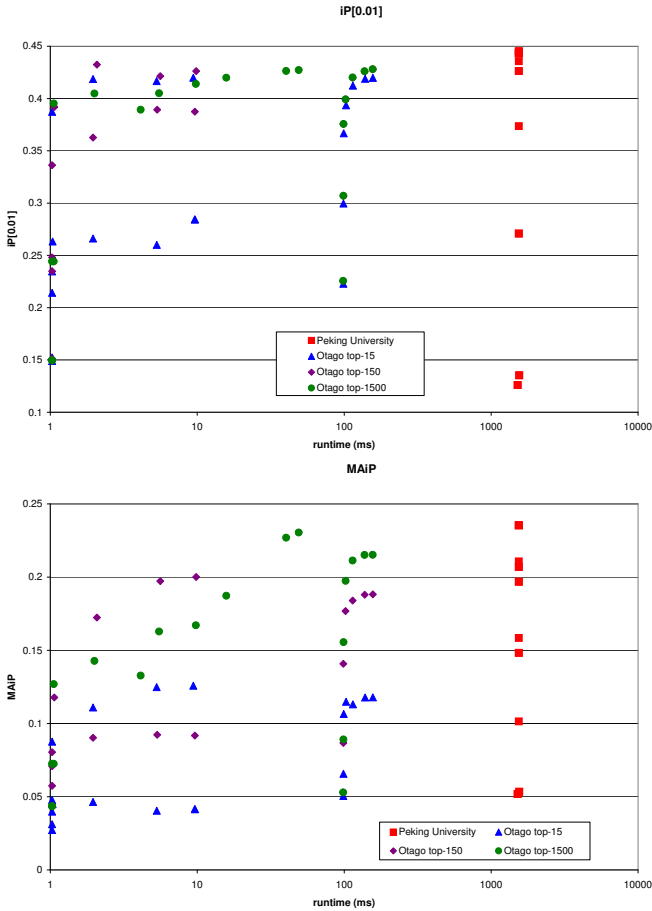


Fig. 6. Trade-off between Effectiveness and Efficiency: iP[0.01] (top) and MAiP (bottom)

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.

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

(a) Relevant in Context Task		(b) Restricted Relevant in Context Task	
	1 2 3 4 5 6 7 8 9 10		1 2 3 4 5 6 7 8 9 10
p22	* * * * * * * * *	p167	- * * * * * * * *
p167	- * * * * * * *	p98	- - - * * * * *
p98	- - * * * * *	p5	* * * * * * *
p5	* * * * * *	p4	* * * * * *
p4	* - - * *	p65	* * * * *
p65	- - * *	p78	- - * *
p25	- * *	p55	- - *
p62	* *	p6	- *
p55	-	p29	*
p6		p72	

(c) Restricted Focused Task		(d) Efficiency Task	
	1 2 3 4 5 6 7 8 9 10		1 2 3 4 5
p68	- - - - - * * * *	p167	- - * *
p55	- - - - - - *	p4	- * *
p9	- - - - - *	p68	* *
p98	- - - - - *	p29	*
p167	- - - - *	p98	
p65	- - - -		
p5	- - -		
p557	- -		
p4	-		
p29			

Figure 6 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 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, and the fifth run (*p98-I10LIA4FBas*) uses FOL passages. 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 10 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

Table 11. 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-runRiCOfref	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

9–10. Of the 45 possible pairs of runs, there are 36 (or 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 11 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 5 for the full name of group. The second to fifth column list generalized precision at 5, 10, 25, 50 retrieved articles. The sixth column lists mean average generalized precision.

Table 12. 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

Comparing Table 11 using the F-score and Table 6 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 12 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 5 for the full name of group. The second to fifth column list generalized precision at 5, 10, 25, 50 retrieved articles. The sixth column lists mean average generalized precision.

Comparing Table 12 using the F-score and Table 7 using the T2I-score, we see some agreement.

- 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.

Table 13. 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-runRiCOPref	0.5808	0.5346	0.7529	0.3177	0.3382

- 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 13 shows the best run of the top 10 participating groups. The first column gives the participant, see Table 5 for the full name of group. The second 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

Table 14. 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-I10LIA1EITri	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-runRiCOfRef	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-runReRiCOfRef	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-runFocCOfRef	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	

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 ranks 2, 8.

If we break-down all runs over the original tasks, shown in Table 14, 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 6, seven Restricted in Context runs correspond to Table 7, seven Restricted Focused runs correspond to Table 8, and five Efficiency runs correspond to Table 9. 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, retrieving 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.

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. This 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 [7]. 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 120 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 [11]. There are many ways of (re)using the resulting test collection for passage retrieval, XML element retrieval, or article retrieval.

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.

This is the fifth year that INEX has studied ad hoc retrieval against the Wikipedia. In 2006–2008 the English Wikipedia of early 2006 transformed into XML was used covering 659,338 Wikipedia articles [4]. Over the three years a combined test collection of 291 topics was created. In 2009–2010 a new collection was created based on a late 2008 dump of the English Wikipedia, containing 2,666,190 Wikipedia articles and incorporating semantic annotations from YAGO [based on 12]. Over the last two years a combined test collection of 120 topics was created. The test collections on Wikipedia have large sets of topics, 291 for the 2006–2008 Wikipedia and 120 for the 2009–2010 Wikipedia. There are relevance judgments at the passage level (both best-entry-points as well as the exact relevant text) plus derived article-level judgments. The resulting judgments are relatively “complete” due to the varied pools and especially the encyclopedic corpus [11]. There is a range of evaluation measures for evaluating the various retrieval tasks [1, 9], in addition to the standard measures that can be used for article-level retrieval. In addition, there is rich information on topic authors and assessors, and their topics and judgments based on extensive questionnaire, allowing for detailed further analysis and reusing topics that satisfy particular conditions [6, 8]. After five years, there seems little additional benefit in continuing with focused retrieval against the Wikipedia corpus, given the available test collections that are reusable in various ways. It is time for a new challenge, and other tracks have started already addressing other aspects of ad hoc retrieval: the INEX 2010 Book Track using a corpus of scanned books, the INEX 2010 Data Centric Track using a corpus of IMDb data, and the INEX 2010 Interactive Track using a corpus of Amazon and Library Thing data.

Acknowledgments. Jaap Kamps was supported by the Netherlands Organization for Scientific Research (NWO, grants 612.066.513, 639.072.601, and 640.-001.501). Paavo Arvola and Johanna Vainio were supported by the Academy of Finland (grants #115480 and #130482).

References

- [1] Arvola, P., Kekäläinen, J., Junkkari, M.: Expected reading effort in focused retrieval evaluation. *Information Retrieval* 13, 460–484 (2010)
- [2] Beigbeder, M.: Focused retrieval with proximity scoring. In: *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC 2010)*, pp. 1755–1759. ACM Press, New York (2010)
- [3] Clarke, C.L.A.: Range results in XML retrieval. In: *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, Glasgow, UK, pp. 4–5 (2005)
- [4] Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. *INEX 2006* 40, 64–69 (2006)

- [5] Géry, M., Largeton, C., Thollard, F.: 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, pp. 763–769. IEEE Computer Society Press, Los Alamitos (2008)
- [6] Kamps, J., Larsen, B.: Understanding differences between search requests in XML element retrieval. In: Trotman, A., Geva, S. (eds.) Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology, pp. 13–19 (2006)
- [7] Kamps, J., Koolen, M., Lalmas, M.: Locating relevant text within XML documents. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 847–849. ACM Press, New York (2008)
- [8] Kamps, J., Pehcevski, J., Kazai, G., Lalmas, M., Robertson, S.: INEX 2007 evaluation measures. In: Fuhr, N., Kamps, J., Lalmas, M., Trotman, A. (eds.) INEX 2007. LNCS, vol. 4862, pp. 24–33. Springer, Heidelberg (2008)
- [9] Kamps, J., Lalmas, M., Larsen, B.: Evaluation in context. In: Agosti, M., Borbinha, J., Kapidakis, S., Papatheodorou, C., Tsakonas, G. (eds.) ECDL 2009. LNCS, vol. 5714, pp. 339–351. Springer, Heidelberg (2009)
- [10] Kekäläinen, J., Järvelin, K.: Using graded relevance assessments in IR evaluation. *Journal of the American Society for Information Science and Technology* 53, 1120–1129 (2002)
- [11] Pal, S., Mitra, M., Kamps, J.: Evaluation effort, reliability and reusability in XML retrieval. *Journal of the American Society for Information Science and Technology* 62, 375–394 (2011)
- [12] Schenkel, R., Suchanek, F.M., Kasneci, G.: YAWN: A semantically annotated Wikipedia XML corpus. In: 12. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web (BTW 2007), pp. 277–291 (2007)
- [13] Trotman, A., Geva, S.: Passage retrieval and other XML-retrieval tasks. In: Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology, University of Otago, Dunedin New Zealand, pp. 43–50 (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	goo100RF ^T	RFoc	CO	FOL	
9	1295	yahRF ^T	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	runRiC ^O Ref	RiC	CO	FOL	Reference run Article-only
65	1274	runReRiC ^O Ref	RRiC	CO	FOL	Reference run
65	1275	runFoc ^O Ref	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	I10LIA1EITri	RiC	CO	Ele	Phrases
98	1260	I10LIA1F ^T Tri	RiC	CO	FOL	Phrases
98	1270	I10LIA2F ^T Tri	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

The Potential Benefit of Focused Retrieval in Relevant-in-Context Task

Paavo Arvola and Johanna Vainio

University of Tampere, School of Information Sciences
33014 University of Tampere, Finland
{paavo.arvola,s.johanna.vainio}@uta.fi

Abstract. This study addresses the Relevant-in-Context retrieval task and seeks justification for systems providing focused answers to be successful in it. 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 the relevant material is located in the document start. 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 various measures, in order to study how different T2I values affect the mutual rankings and measure the systems in locating the relevant material within a document.

Keywords: Relevant-in-Context, evaluation, metrics.

1 Introduction

The Relevant-in-Context (RiC) retrieval task can be considered as focused retrieval enhanced document retrieval, where the retrievable unit is a document having the best matching passages highlighted. In other words, the user's attention is drawn to the relevant content of the document. This kind of grouping the result passages by their document is called *fetch and browse* retrieval [5]. It is worth mentioning that the RiC task is considered as the most credible task of all tasks in the ad hoc track [12].

Document parts, referred to as elements, have both *hierarchical* and *sequential order* within the document. The sequential order corresponds with the order of the running text. There is also an implicit *chronological order* (temporal order) of a document's text, which is formed, when the document is read by a user. This is actually the internal ranking of the parts within the document. From this perspective, a focused retrieval system should have an impact on this chronological reading order, so that the relevant content of a document is found and read with minimal effort.

The reading order assumption behind the official measure (T2I(300)) of the RiC task considers the sequential order of an individual document [1,3]. When everything within a document is delivered by the retrieval system, this resembles to a full document retrieval scenario, where 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 [13]. If, instead of everything, only parts of the document are retrieved, those are assumed to be read sequentially first. Thus, a simple browsing model for a document with two consecutive phases is assumed:

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

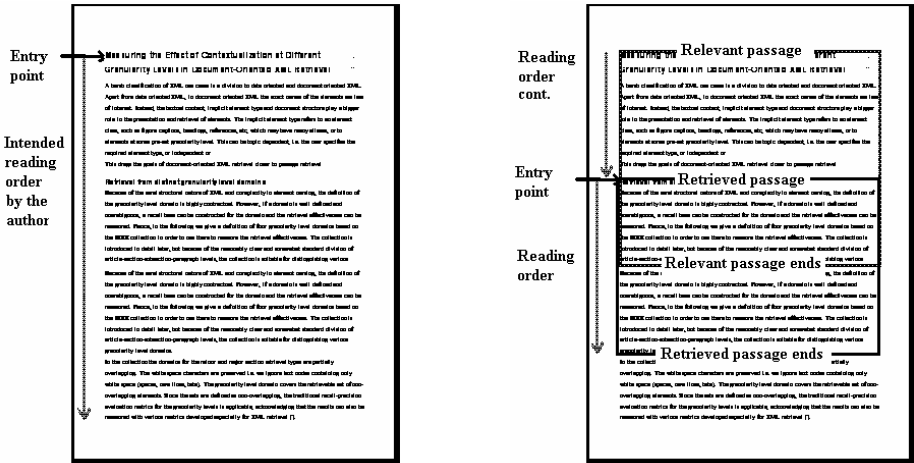


Fig. 1. Conventional reading order (left) and focused retrieval driven reading order (right)

Figure 1 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.

However, in Figure 1, this is not the case. The relevant content is already at the beginning of the document, which part is shown to the user by default and there is no way a focused retrieval system to improve the chronological reading order of the document. In general, the potential benefit of a focused retrieval system depends also on the document qualities, which are in this study the location of relevant material and the relevance density [4], i.e. how much relevant material there is in relation to all text in the document. With these qualities in mind, we aim to study the potential and actual benefit of the RiC task by using the INEX data.

The rest of the study is organized so that in Section 2, we report results for the INEX 2010 runs measured with different T2I assumptions and as the effort the user has to take in order to localize the relevant content with localizing effort (LE) metric. The metric is based on how much text the user is expected to browse through before

discovering the relevant material from a document. The metric is introduced in Section 2.1. In Section 3, 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.

2 Metrics and Results

The official INEX measures, such as the T2I based character precision (ChP) as well as the F-Score used in previous years, are defined in the ad hoc track overview paper of the proceedings (see also [7, 8]). Thus, in this study we introduce only the localizing effort metrics.

2.1 Cumulating the Localizing Effort

In the RiC task, separate scores are calculated for each individual retrieved document as a document score, and for the document result lists as a list score. In T2I based character precision recall metric (ChPR) and F-Score metrics the relevance score values scale between 0 and 1, and the list score is calculated analogously to generalized precision-recall [10], whereas in localizing effort (LE), the document scoring is looser and the list score is calculated by cumulated effort (CE), which has evolved from the cumulated gain metric [6]. Next, we present briefly the LE metrics for the document score and CE for the list score.

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:

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

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:

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

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 :

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

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:

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

$$NR = 5$$

Next, we report results using the CE, generalized precision metrics, for the list score together with localizing effort score, ChPR and F-Score for each individual result document.

2.2 Results

The official measure T2I(300) is interpreted so, that the user is willing to read up to 300 *irrelevant* characters within a result document before quitting. The amount of 300 characters is a low figure and applies only to a small screen or a snippet. In comparison, the amount of 2000 characters corresponds approximately to the first page of the present study. Therefore, in Table 1 (columns 5-7), we report results of submitted INEX runs¹ using higher T2I values, namely 500, 1000 and 2000 characters as the T2I threshold. The results using F-Score with β value 0.25 and 1 are reported (columns 8-9) as well. Kendall's tau rank correlation value between the T2I(300) and

¹ One run producing zero results is omitted.

T2I(2000) measures is relatively high 0.877 in comparison for instance to the values between T2I(300) and the F-Scores ($\beta = 0.25$: 0.471, $\beta = 1$: 0.374).

Figure 2 shows the NCE curves of four INEX 2010 runs by the top 5 participants (Emse301R, I10LIA1FTri, 31p167, Reference/qtau) plus the reference run as full document run: Reference_1. Using the average measure MANCE@300 the run I10LIA1FTri of LIA – Univ. Avignon ranked 1st among the automatic runs. The MANCE@300 and T2I(300) have a rank correlation of 0.808 (Table 1, column 10).

In Table 1, the columns 3 and 4 report the median average number of retrieved characters and median average first occurrence of the retrieved text in characters from the document start, when top 20 documents from each assessed topics are considered. In other words, for each topic the median value of the top 20 ranked documents is calculated and the averaging is done over all topics. This is done in order to illustrate the “focusedness” of the runs. Only top 20 (accepted) documents are considered, because lower ranks tend to match more poorly, and either do not affect so much the overall performance.

It seems that systems retrieving large quantities of material per document retain their performance levels regardless of the measure. However, systems delivering focused answers, such as the runs of QUT, get harshly penalized when measured with the F-Score. The same phenomenon is present with the starting point criterion, i.e. the earlier the better for the F-Score, although the Emse303r and Emse301r perform well regardless of the measure, even though they start on average very late.

In the next section, we analyze in detail what are the potential benefits of the RiC task in the light of the starting point of the relevant text and the amount of relevant material per document.

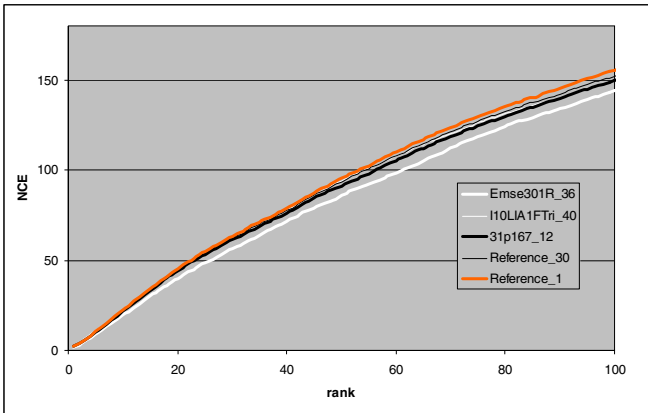


Fig. 2. Normalized cumulated effort curves of the runs of some of the best participants

Table 1. In columns 5 - 9 MA_GP scores for official INEX runs in ChPR using T2I values 500, 1000 and 2000 in F-Score using beta values 0.25 and 1. Column 10 shows mean average cumulated effort at 300 documents. Median average number of retrieved characters and median average starting point of retrieved text in columns 3 and 4, when top 20 documents from each assessed topic are considered.

name	institute	# rel chars	start pos	T2I(500)	T2I(1000)	T2I(2000)	F0.25	F1	MANCE@300
Emse303R	ENSM-SE	4208	1653	20.3	20.5	19.7	19.1	19.2	161.4
Emse301R	ENSM-SE	4242	1662	20.5	21.2	20.2	19.7	20.3	158.9
32p167	Peking University	5422	215	17.1	18.2	18.6	13.2	13.5	169.0
36p167	Peking University	5422	215	17.1	18.2	18.6	13.2	13.5	169.0
31p167	Peking University	3681	320	17.0	18.2	18.4	9.5	8.1	167.5
37p167	Peking University	3681	320	17.0	18.2	18.4	9.5	8.1	167.5
I10LIA1FTri	LIA - University of Avignon	1112	514	16.5	17.1	17.0	13.0	6.9	164.9
I10LIA1FUni	LIA - University of Avignon	1022	526	16.6	17.2	17.2	12.9	6.8	167.9
Reference	Queensland University of Technology	241	96	16.5	18.0	18.3	6.3	3.3	169.8
22p167	Peking University	8016	52	16.7	18.0	18.3	16.3	18.4	172.8
26p167	Peking University	8016	52	16.7	18.0	18.3	16.3	18.4	172.8
23p167	Peking University	7859	52	16.6	17.9	18.2	15.7	17.6	172.7
29p167	Peking University	7859	52	16.6	17.9	18.2	15.7	17.6	172.7
21p167	Peking University	9014	59	16.6	18.0	18.2	17.3	20.1	173.4
27p167	Peking University	9014	59	16.6	18.0	18.2	17.3	20.1	173.4
Reference	University of Otago	9115	0	16.1	17.8	18.2	17.1	20.0	171.0
v_sstem	University of Otago	7309	0	15.9	17.5	17.9	16.9	19.9	172.3
v_porter	University of Otago	7412	0	15.7	17.2	17.7	16.8	19.8	175.1
I10LIA1EUni	LIA - University of Avignon	1721	1069	15.1	16.7	17.1	3.6	2.2	173.3
runRiCOREf	Radboud University Nijmegen	8964	0	15.4	17.0	17.2	16.2	18.9	173.9
I10LIA1ETri	LIA - University of Avignon	1569	1078	15.0	16.7	16.9	3.4	2.1	170.1
ruc-2010-base2	Renmin University of China	8850	0	15.5	17.3	17.7	16.7	19.7	173.7
prfr-0.05	Queensland University of Technology	237	131	14.8	16.3	16.4	5.6	3.0	172.2
v_otago_w_pmi	University of Otago	7229	0	15.2	16.9	17.2	16.4	19.2	176.0
v_otago_stem_1	University of Otago	7219	0	15.2	16.9	17.2	16.3	19.2	176.1
v_ostem_w_its	University of Otago	7222	0	15.2	16.8	17.2	16.3	19.2	176.0
RMIT10titleO	RMIT University	3420	0	14.6	15.7	15.9	15.4	18.0	173.4
ruc-2010-base1	Renmin University of China	8044	0	15.0	16.8	17.0	16.2	19.0	177.8
RMIT10title	RMIT University	9935	0	14.5	16.1	16.3	15.4	18.2	180.6
v_no_stem	University of Otago	7255	0	14.3	15.6	15.9	15.2	17.9	177.0
Emse301	ENSM-SE	2726	1989	12.3	12.1	11.3	12.0	12.5	192.5
prfr-0.10	Queensland University of Technology	247	195	12.9	14.3	14.3	5.0	2.7	174.5
Emse303	ENSM-SE	1907	1443	11.8	11.6	11.0	11.8	12.0	194.0
24p167	Peking University	7961	69	11.7	13.0	12.9	12.1	14.4	175.4
30p167	Peking University	7961	69	11.7	13.0	12.9	12.1	14.4	175.4
DUR10datcl	Doshisha University	6478	0	11.0	11.4	11.4	11.2	12.9	211.1
v_sstem_w_its	University of Otago	7402	0	10.9	12.1	12.3	11.9	14.0	203.1
prfr0.25	Queensland University of Technology	269	347	9.2	10.3	10.1	3.7	2.2	186.2
0	University of Amsterdam	4581	853	6.8	6.5	6.3	6.0	5.7	213.7
0	University of Amsterdam	4885	682	6.5	6.2	6.0	5.6	5.3	217.7
UWBOOKRIC2010	University of Waterloo	2798	9401	6.2	6.0	6.1	4.6	3.6	223.0
ISI2010_ric_ro	INDIAN STATISTICAL INSTITUTE	2845	804	6.4	6.8	6.5	6.9	6.3	207.3
ISI2010_ric_aggr	INDIAN STATISTICAL INSTITUTE	14613	791	2.6	2.8	2.6	2.9	2.8	240.3
UJM_33456	Saint Etienne University	4150	267	2.1	2.3	2.2	2.1	2.4	249.3
DUR10BU_D	Doshisha University	2880	3385	1.6	1.5	1.6	1.7	1.5	256.3
DUR10BU_S	Doshisha University	2880	3385	1.6	1.5	1.6	1.7	1.5	256.3

3 Analysis Based on Three Document Retrieval Scenarios

In a nutshell, 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 defined based on the exhaustivity and specificity dimensions of relevance [9]. 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, 11].

However, aiming to rank the documents based on their relevance density is problematic from 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. Namely, most user interfaces offer the document start to the user by default and there is no need to guide the user elsewhere.

We claim that focused retrieval can be beneficial in locating relevant material from a document especially, when

- the relevance density of the relevant result document is low and
- the relevant material doesn't occur at the beginning of the document.

Based on these axioms, 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, i.e. the ranking is optimal. When density is considered, we define the optimality so that the documents are ranked in descending order by their density. When, in turn, the starting point is considered, the optimality means that the documents are ranked by the distance of the first relevant character from the document start. It is worth noting that the optimal scenario is actually the worst scenario from the perspective of the potential benefit of focused retrieval. The average scenario is the average of the relevant documents in the recall base and the realistic scenario is based on the fetch phase of some best performing runs of INEX 2010 (Emse303R, 32p167, I10LIA1FTri, Reference/qttau).

Figure 3 presents 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 in studying about finding the relevant

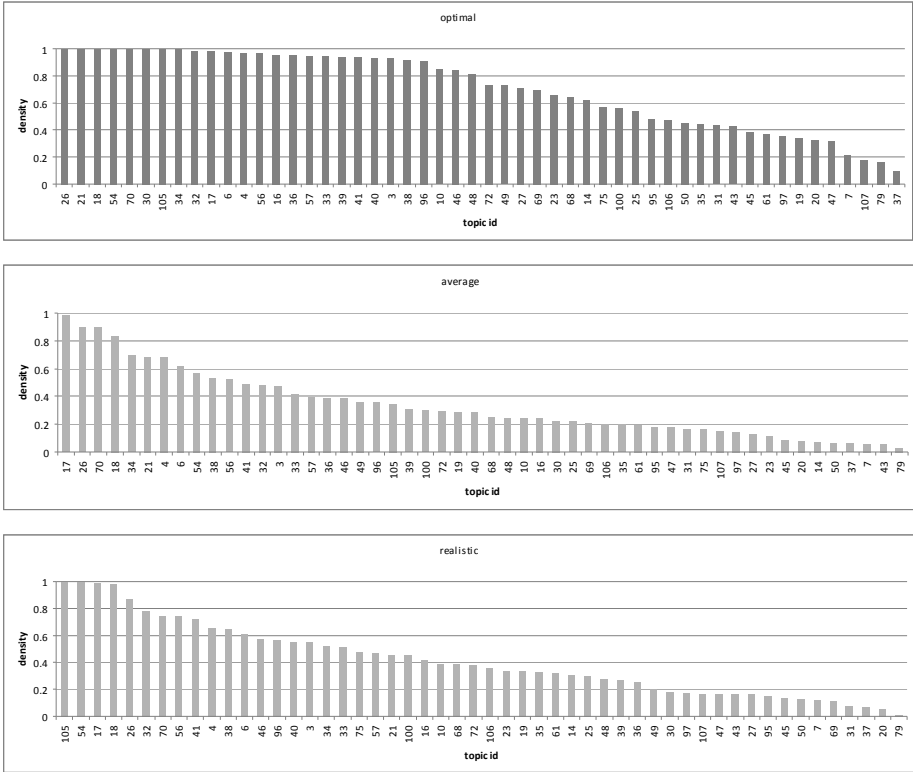


Fig. 3. The average densities of retrieved documents by topic with optimal, average and realistic document retrieval scenarios. The topics are sorted according to the average density for each scenario.

material from a document. The average densities over all topics are 0.71, 0.34 and 0.42, for optimal, average and realistic scenarios respectively.

Figure 4 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 more room for improvements there are from the focused retrieval perspective. The document retrieval scenarios are equivalent to the analysis based on density in Figure 3, 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 more beneficial a focused retrieval system can be. The average distances are approximately 230, 3049 and 1836 characters for optimal, average and realistic scenarios respectively.

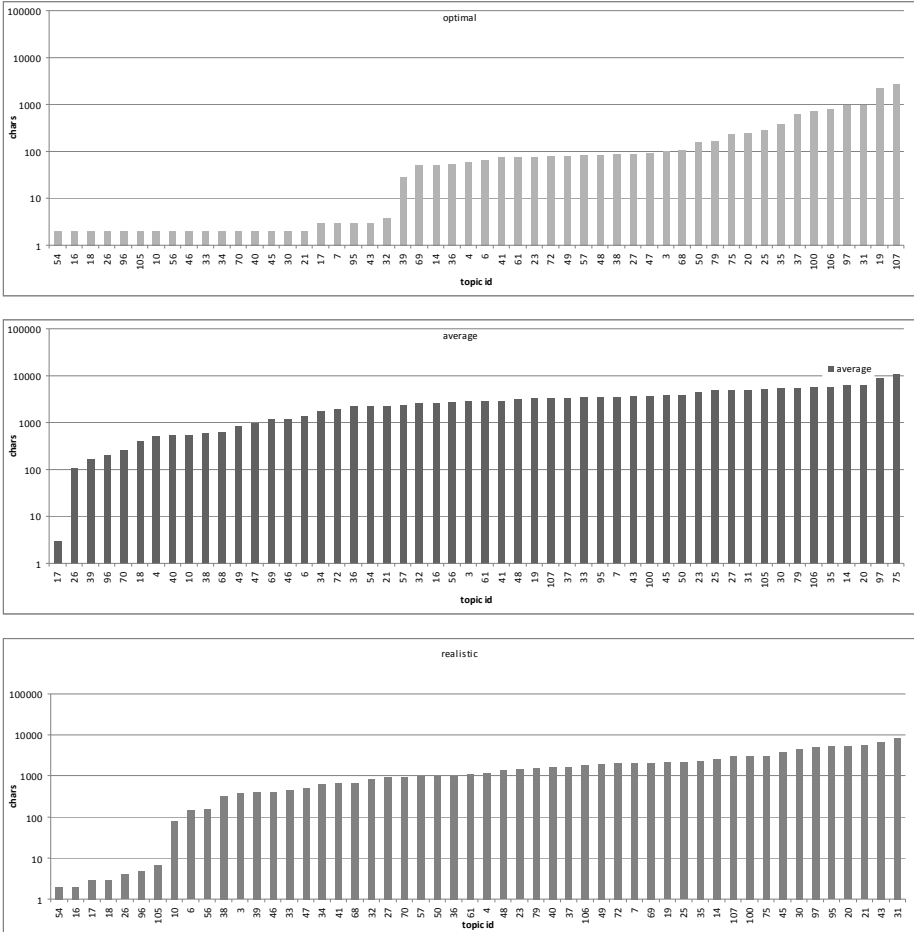


Fig. 4. The average distances of the first relevant passage from the document start measured in characters by topic with optimal, average and realistic document retrieval scenarios. The topics are sorted according to the distance for each scenario. Note the logarithmic scale.

4 Discussion and Conclusions

The focused retrieval strategy is not beneficial, if a result document is densely relevant or the relevant content is located in the document start. Therefore, 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. The current official INEX measures alongside with the localizing effort seemed to favor systems retrieving focused answers in comparison with the F-Score. Increasing the T2I value from 300 to 2000 did not affect substantially on the results.

In order to study the potential benefit of the RiC task 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.

This study elicits the information about the quality of the topics and the test collection in relation to focused retrieval. The current 2010 collection seems to serve the aims of focused retrieval better than the previous collection [4], yet the current collection covers still several types of topics. In future studies, two separate research lines could be distinguished: one focusing on topics which can be answered with shorter passages and another over realistic set of topics, where some topics can be answered only with verbose passages.

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 non 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 grants #115480 and #130482.

References

1. Arvola, P.: Passage Retrieval Evaluation Based on Intended Reading Order. In: Workshop Information Retrieval, LWA 2008, pp. 91–94 (2008)
2. Arvola, P., Junkkari, M., Kekäläinen, J.: Applying XML Retrieval Methods for Result Document Navigation in Small Screen Devices. In: Proceedings of MUIA at MobileHCI 2006, pp. 6–10 (2006)
3. Arvola, P., Kekäläinen, J., Junkkari, M.: Expected reading effort in focused retrieval evaluation. *Information Retrieval* 13(4), 460–484 (2010)
4. Arvola, P., Kekäläinen, J., Junkkari, M.: Focused access to sparsely and densely relevant documents. In: Proceedings of SIGIR 2010, pp. 781–782 (2010)
5. Chiamarella, Y.: Information retrieval and structured documents. *Lectures on Information Retrieval*, pp. 286–309 (2001)
6. Järvelin, K., Kekäläinen, J.: Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems* 20(4), 422–446 (2002)

7. Kamps, J., Lalmas, M., Pehcevski, J.: Evaluating relevant in context: Document retrieval with a twist. In: Proceedings SIGIR 2007, pp. 749–750 (2007)
8. Kamps, J., Pehcevski, J., Kazai, G., Lalmas, M., Robertson, S.: INEX 2007 evaluation measures. In: Fuhr, N., Kamps, J., Lalmas, M., Trotman, A. (eds.) INEX 2007. LNCS, vol. 4862, pp. 24–33. Springer, Heidelberg (2008)
9. Kazai, G., Lalmas, M.: Notes on what to measure in INEX. In: Proceedings of the INEX, Workshop on Element Retrieval Methodology, INEX 2005, pp. 22–38 (2005)
10. Kekäläinen, J., Järvelin, K.: Using graded relevance assessments in IR evaluation. *Journal of the American Society for Information Science and Technology* 53, 1120–1129 (2002)
11. Piwowarski, B., Lalmas, M.: Providing consistent and exhaustive relevance assessments for XML retrieval evaluation. In: Proceedings of CIKM 2004, pp. 361–370 (2004)
12. Trotman, A., Pharo, N., Lehtonen, M.: XML-IR Users and Use Cases. In: Fuhr, N., Lalmas, M., Trotman, A. (eds.) INEX 2006. LNCS, vol. 4518, pp. 400–412. Springer, Heidelberg (2007)
13. de Vries, A.P., Kazai, G., Lalmas, M.: Tolerance to irrelevance: A user-effort oriented evaluation of retrieval systems without predefined retrieval unit. In: Proceedings of RIAO 2004, pp. 463–473 (2004)

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

Michel Beigbeder¹, Mathias G ery², Christine Largeton², and Howard Seck³

¹  cole Nationale Sup erieure des Mines de Saint- tienne
michel.beigbeder@emse.fr

² Universit  de Lyon, Saint- tienne, France

{Mathias.Gery,Christine.Largeton}@univ-st-etienne.fr

³ Universit  Paris-Dauphine, Paris, France
bseck@olaneo.fr

Abstract. This paper presents our participation in the *Relevant in Context task* (ad-hoc track) during the 2010 INEX competition, and a posterior analysis. Two models presented in previous editions of INEX by the authors were merged for our 2010 participation. The first one is based on the proximity of the query terms in the documents [1] and the second one is based on learnt tag weights [2]. The results demonstrate the improvement of focused information retrieval, thanks to the integration of the tag weights in the approach based on proximity.

1 Introduction

INEX Ad-Hoc track aims at evaluating focused XML information retrieval on large collections of structured documents in order to retrieve small units of information, smaller than document. Indeed, the structure allows to divide a document into elements which can be returned to a user instead of the whole document. But, as the tags may be used to emphasize words, the structure can also be used to improve the detection of relevant information. Thus, a word is probably more important if it is marked by certain tags, for instance if it appears in a particular font or if it appears within certain parts of a document (e.g. a title). In order to exploit this hypothesis, we proposed an extension of the BM25 weighting function, called BM25t [2], which takes the tags found in XML documents into account. In this model, a weight is estimated for each tag during a learning stage. This weight measures the capacity of the tag to emphasize terms which appear in relevant passages. This model was evaluated in a previous INEX campaign [3].

However, other approaches than the probabilistic model seemed promising in the context of focused information retrieval, notably those based on the proximity of the query terms in the documents. The use of the term positions first appeared in some implementations of the boolean model [4]. However, this model did not allow ranking of documents but this limitation was removed in subsequent works [5,6,7]. This approach has proved effective in the INEX 2009 campaign [8]. For this reason, the model based on the proximity of the query terms

in structured documents [1] and the BM25t model [2] were merged for our 2010 participation. This led us to study the way to take into account the structure into the model based on the proximity. So, 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 de Lyon - Saint-Étienne*.

The proximity based model uses the positions of the occurrences of the query terms in the documents to score them. More precisely, we define a text area around each occurrence of a query term. The positions belonging to this text area are influenced by this occurrence of the query term. We quantify this influence with a function, called *influence function*. Then, the influence functions of the query terms are combined in order to score the documents. A more formal presentation of these notions appears in section 2. The section 3 will be dedicated to the learning of the tag weights according to their capacity to mark relevant passages on a training collection.

The integration of the structure into the proximity based model was done by modifying the shape of the influence functions according to the weight of the tags. In other words, in our model, the values of the function computed for an occurrence of each query term take into account the weight of the tag of the element in which this occurrence appears. Finally in section 4, we present how elements are scored with these weighted influence functions and how our runs were built with these scores. We also present runs which mix our proximity scores with the INEX Reference run.

Moreover, experiments posterior to the INEX campaign are also presented in this last section. Indeed, as it was pointed out during the INEX workshop, one limit of the evaluation in the INEX campaign of the model based on the proximity lies in the fact that this model requires boolean queries which do not exist in the evaluation framework. In order to avoid manual intervention to build boolean queries, the topic title fields were automatically transformed into boolean queries and the results obtained using these automatically generated queries are presented in this last section.

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 an 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*, e.g. `ss1`, `ss2`, `ss3`, `ss4`);
- B_t : the *title tags*;
- $\overline{B_l \cup B_t}$ (the complement of the set $B_l \cup B_t$): the set of tags that are neither logical tags, nor title tags.

The structure is exploited at two levels. Firstly, the logical tags belonging to B_l are used to determine the elements which can be returned to the user. Secondly,

<p>Document d_1</p> <pre> <article>Document <ss1><st>Caesar in title</st>The section which deals with Caesar</ss1> Following of the document. </article> </pre> <p> $T = \{caesar, deals, document, following, in, of, section, the, title, which, with\}$ $E(d_1) = \{d1/article[1], d1/article[1]/ss1[1], d1/article[1]/ss1[1]/st[1],$ $d1/article[1]/ss1[1]/em[1], d1/article[1]/ss1[1]/em[2]\}$ $B = \{article, em, ss1, st\}$ $B_l = \{article, ss1\}$ $B_t = \{st\}$ $d_1(0) = document$ $d_1(1) = caesar$ $d_1(2) = in$ \dots $d_1(9) = caesar$ \dots $d_1 = 14$ $d_1^{-1}(caesar) = \{1, 9\}$ $x_1(d1/article[1]/ss1[1]) = 1$ $x_2(d1/article[1]/ss1[1]) = 9$ $e(5) = d1/article[1]/ss1[1]/em[1]$ $e_l(5) = d1/article[1]/ss1[1]$ $b(5) = em$ $M_{st}(d1/article[1]) = \{1, 2, 3\}$ $M_{em}(d1/article[1]) = \{5, 7\}$ </p>

Fig. 1. Collection example with one document

the tags belonging to B , including the logical tags, are used to estimate the relevance of an element as detailed in section 3.

When the vector space model considers the number of occurrences of the terms in the documents (through the term frequency or the inverse document frequency), the proximity based model, introduced by [7] takes also into account their positions in the document. Thus, a document is defined as a function which associates a term $t \in T$ to each position in the document:

$$\begin{aligned}
 d: \mathbb{N} &\rightarrow T \\
 x &\mapsto d(x)
 \end{aligned} \tag{1}$$

Given a position x in a document, $e(x)$ is the deepest element (in the XML tree) 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 $e(x)$. Given an element e , $x_1(e)$ (resp. $x_2(e)$) denotes the position of its first (resp. last) term. Figure 1 shows a sample document and illustrates all these notations. For instance, for the fifth position, corresponding to the word *section*, the deepest element is $e(5) = d1/article[1]/ss1[1]/em[1]$ while the deepest logical element is $e_l(5) = d1/article[1]/ss1[1]$.

In order to compute the score $s(q, e)$ of an element e , given a query q , this model introduces the influence function of a term to a position and the influence

of a query to a position. These notions are briefly presented in the following sections. An extended presentation can be found in [1].

2.2 Influence Function of a Term to a Position

Firstly, we modelize the influence of one occurrence of term t at position i on one position x in a document d with an *influence function*. 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 : $x \mapsto \max\left(\frac{k-|x-i|}{k}, 0\right)$ where k is a parameter which controls the size of the influence area. The graphs of such functions have a triangular shape, so we call *triangle functions* these functions. When the distance between x and i is greater than k , the influence is zero – that’s to say that the occurrence of term 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 distance¹.

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

$$p_t^d(x) = \max_{i \in d^{-1}(t)} \left(\mathbf{1}_{e_l(i)} \cdot \max \left(0, \frac{k - |x - i|}{k} \right) \right) \quad (2)$$

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 explained previously, the influence function of a term to a position is used to compute the influence of a query to a position which is used itself to compute the score of an element for this query. This influence function of a query to a position is defined as follows: in the simplest case where a query q contains only one term $t \in T$, the influence of the query to a position x equals the influence of the term t to the position x :

$$p_q^d(x) = p_t^d(x) \quad (3)$$

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

In the other cases, as a boolean query, the query q is a tree with conjunctive and disjunctive nodes. To define the influence on a conjunctive node q_1 AND q_2 the minimum is taken over the influence functions of its children:

$$p_{q_1 \text{ AND } q_2}^d(x) = \min(p_{q_1}^d(x), p_{q_2}^d(x)) \quad (4)$$

Similarly, the influence on a disjunctive node q_1 OR q_2 is defined as the maximum over the influence functions of its children:

$$p_{q_1 \text{ OR } q_2}^d(x) = \max(p_{q_1}^d(x), p_{q_2}^d(x)) \quad (5)$$

These formulas are recursively used during a post order traversal of the query tree to compute the influence function at the root of the tree, that's to say the influence function of the query itself.

2.4 Score 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 $s(q, e)$ of an element e is computed with the following formula:

$$s(q, e) = \frac{\sum_{x_1(e) \leq x \leq x_2(e)} p_q^d(x)}{x_2(e) - x_1(e) + 1} \quad (6)$$

where $x_1(e)$ (resp. $x_2(e)$) is the first position (resp. the last position) of the textual content of the element e .

3 Weighting Tags and Modulating Influence Function Shapes

As explained previously, we suppose that the tags may be used to emphasize words. So, the structure can be used to improve the detection of relevant information. A weight is estimated for each tag using a training set. This weight measures the capacity of the tag to emphasize terms in relevant or in non relevant passages.

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 afterwards used to modulate the influence function of the term occurrences that appear in the elements of type b .

The set of assessments from INEX 2009 is used as a learning set. In the contingency table of Table 1, $R_q(e)$ is the set of the relevant positions in the element $e \in E$ for the topic $q \in Q$, and $M_b(e)$ is the set of the positions of e marked by the tag $b \in B$.

Table 1. Contingency table for the query q and for the tag b

	$R_q(e)$	$\overline{R_q(e)}$
$M_b(e)$	$t_{rm}(b, q)$	$t_{\overline{rm}}(b, q)$
$\overline{M_b(e)}$	$t_{r\overline{m}}(b, q)$	$t_{\overline{r\overline{m}}}(b, q)$
Total	$t_r^{coll}(q)$	$t_{\overline{r}}^{coll}(q)$

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

$$w_b(q) = \frac{\frac{t_{rm}(b, q) + s}{t_{rm}(b, q) + t_{r\overline{m}}(b, q) + s}}{\frac{t_{\overline{rm}}(b, q) + s}{t_{\overline{rm}}(b, q) + t_{\overline{r\overline{m}}}(b, q) + s}} \quad (7)$$

with:

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

In fact, we believe that the capacity of a tag to highlight relevant terms (or on the contrary those that are not relevant) is intrinsic to the tag itself and is not dependant on the query. Thus, we estimate the weight w_b for each tag b instead of a weight for each pair (tag b , query q). The weight w_b of a tag b is averaged using the set of 68 evaluated queries from INEX 2009, using the formula:

$$w_b = \frac{1}{|Q|} \sum_{q \in Q} w_b(q) \quad (8)$$

3.2 Modulating Influence Function Shapes

Then the weights of the tags are integrated into the score of an element. More precisely, the weights of the tags are used to modulate the influence function of the query term occurrences with two methods. In the first one, the height of the triangle is modified and the resulting influence function of a term is:

$$ph_t^d(x) = \max_{i \in d^{-1}(t)} \left(\mathbf{1}_{e_l(i)} \cdot \max \left(0, w_{b(i)} \cdot \frac{k - |x - i|}{k} \right) \right) \quad (9)$$

and in the second one, both the height and the width of the triangle are modified and the resulting influence function of a term is:

$$phw_t^d(x) = \max_{i \in d^{-1}(t)} \left(\mathbf{1}_{e_l(i)} \cdot \max \left(0, \frac{w_{b(i)} \cdot k - |x - i|}{k} \right) \right) \quad (10)$$

4 Experiments

4.1 Building Runs

For the experiments, we used the following sets of logical tags and title tags:

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

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

We submitted four official runs (Emse301, Emse301R, Emse303 and Emse303R) at the *Relevance in Context* task. For the runs Emse301 and Emse301R, the influence function of a query term is *phw*, and for the runs Emse303 and Emse303R, the influence function is *ph*.

As the proximity based model requires boolean queries, for these runs the topic title fields were manually transformed into boolean queries during the competition. We call *Extended queries* this set of queries. After the competition, we conducted posterior experiments in order to obtain a system which is completely automatic in regards to the data currently available in the topics. In this posterior analysis, the following rules were applied to transform the title field into boolean queries:

- removing of the '+' operator
- replacement of the '-' operator by the NOT (!) operator
- the remaining items (either simple terms or phrases) are connected by the AND operator.

We call *Title queries* this set of queries. Then we used the same settings used in our official runs to build another four runs which we named with the same name completed with an 'A'. Thus, for instance, the sole difference between our official run Emse301 and the run Emse301A is the set of queries used.

Table 2 recaps the settings for the runs. Letter 'R' means that the Reference run was used as described latter.

Table 2. Settings for our four official runs and for the four subsequent runs

	Extended queries (INEX 2010)	Title queries (post-INEX)
Height modulation <i>ph</i>	Emse303, Emse303R	Emse303A, Emse303RA
Height and width modulation <i>phw</i>	Emse301, Emse301R	Emse301A, Emse301RA

As each document is analyzed, a score is computed for each logical element according to formula 6. 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 until the ranked list is empty.

For the runs Emse301 and Emse303 and their automatic versions Emse301A and Emse303A, the elements are sorted:

1. firstly, by document score;
2. then, by document id;
3. and finally, by element score.

For the 'R' versions (Emse301R, Emse303R, Emse301RA and Emse303RA) the same sorting keys are used but the Reference run is also used. The elements are returned using the following method: the element of the documents that appear both in our results list and in the Reference run are firstly returned in the order of the Reference run, then the elements of the documents that appear only in our list and finally, the documents that only appear in the Reference run.

4.2 Results

The results obtained by our model during the 2010 INEX competition are presented in Table 3, together with the results obtained during our posterior analysis experiments. The results were computed using the INEX software: *inex_eval* 3.0.

Table 3. *MAgP* results of our four official runs and the four subsequent runs

	Extended queries (INEX 2010)		Title queries (post-INEX)	
	without R	with R	without R	with R
Height modulation <i>ph</i>	Emse303 0.1163	Emse303R 0.1977	Emse303A 0.0760	Emse303RA 0.1591
Height and width modulation <i>phw</i>	Emse301 0.1207	Emse301R 0.1967	Emse301A 0.0751	Emse301RA 0.1596
Baseline	Reference run 0.1436			

The first conclusion concerning our experiments is that both the Reference run and our method get benefits from the other one: use of the Reference Run is very beneficial to every methods and reciprocally all the methods that use the Reference run are significantly better than the Reference run itself.

Furthermore, the experiments permit to compare the methods used to modulate the influence functions with the tag weights. Both strategies *ph* and *phw* improve the Reference run results, but it is not clear if modifying both the height and the width of the triangles is better than only modifying the height.

Finally, the results of the subsequent runs with title queries are not as good as those obtained with extended queries. However, they stay very good when the Reference run is used. Indeed, Table 4 shows that the runs Emse301RA and Emse303RA are ranked just after the runs from "Peking University" which were ranked from 3rd to 6th during the INEX competition.

Table 4. INEX 2010 results (Relevant in Context task)

Rank	<i>MAGP</i>	Institute	Run
1	0.1977	ENSM-SE	Emse303R
2	0.1967	ENSM-SE	Emse301R
3	0.1615	Peking University	32p167
4	0.1615	Peking University	36p167
5	0.1598	Peking University	31p167
6	0.1598	Peking University	37p167
new	0.1596	ENSM-SE	Emse301RA
new	0.1591	ENSM-SE	Emse303RA
7	0.1588	LIA - U. of Avignon	I10LIA1FTri
8	0.1587	LIA - U. of Avignon	I10LIA1FUni
9	0.1521	Queensland U. of Technology	Reference
10	0.1519	Peking University	22p167

5 Conclusion

This article reports the results of our experiments in the *Relevance in Context* task during the 2010 INEX competition and a posterior analysis. Our official INEX 2010 runs used manually built boolean queries. The posterior experiments use automatically built queries which are a conjunctive interpretation of the title topic fields.

Two models presented in previous editions of INEX by the authors were merged for our 2010 participation. The first one is based on the proximity of the query terms in the documents through the use of influence functions around each occurrence of the query terms in the documents [1] and the second one is based on learnt tags weights [2]. The results demonstrate that the proximity model which already proved effective in the previous INEX campaigns is enhanced by modulating the shape of the influence functions of the query terms by the tag weights. It is also shown that the two phase retrieval process with Fetch and Browse gets much benefits from the use of the BM25 based Reference run. Though the best results are obtained with actual boolean queries rather than with the conjunctive interpretation of the title topic field.

Acknowledgements. This work has been partly funded by the Web Intelligence project (région Rhône-Alpes, cf. <http://www.web-intelligence-rhone-alpes.org>) and the Conseil Général de la Loire.

References

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

3. Géry, M., Langeron, C., Thollard, F.: UJM at INEX 2008: Pre-impacting of tags weights. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 46–53. Springer, Heidelberg (2009)
4. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval, ch. 2. McGraw-Hill, New York (1983)
5. Hawking, D., Thistlewaite, P.: Proximity operators - so near and yet so far. In: [9]
6. Clarke, C.L.A., Cormack, G.V., Burkowski, F.J.: Shortest substring ranking (multitext experiments for TREC-4) In: [9]
7. Beigbeder, M., Mercier, A.: An information retrieval model using the fuzzy proximity degree of term occurrences. In: Proceedings of the 2005 ACM Symposium on Applied Computing, SAC 2005, pp. 1018–1022. ACM, New York (2005)
8. Beigbeder, M., Imafouo, A., Mercier, A.: ENSM-SE at INEX 2009: Scoring with proximity and semantic tag information. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 49–58. Springer, Heidelberg (2010)
9. Harman, D.K. (ed.): The Fourth Text REtrieval Conference (TREC-4), Department of Commerce, National Institute of Standards and Technology (1995)

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. This year, we complement the work presented the previous year by enhancing 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 preference relations in the meta search [6], [7]. 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], [14], and [12].

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 [12] which optimizes loss functions focused on the top of the list.

This year, we concentrated our attention on the 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 Wikipedia where the original Wiki syntax has been converted into XML [10]. 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 (such 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 to the process described in [3].

Now, after annotating a set of queries, we can select the elements we will submit 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 following the suggestion made by [2] to improve efficiency and robustness of learning to rank algorithms. Therefore we decided to build our learning base as a pool of the top k 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 for a given 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 :

a 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 [9], TF-IDF or Language Models [15] can be placed in this class.

an 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. For instance, if the retrieved element is a paragraph (i.e. $\langle p \rangle$) the indicator function returns 1 and the other indicator functions return 0.

We sum up in Table 1 all the features used in our experiments for the Restricted Focused and the Efficiency/Thorough tasks¹. 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 . In the end, $BM25(t, x)$ stands for the BM25 model applied to the term t in the element x and $LM_*(t, x)$ is a Language Model where $*$ \in $\{Dirichlet, Jelinek - Mercer, AbsoluteDiscount\}$ is the smoothing function.

2.2 Context Features

The aim of this work is to identify features that are informative to help a learning to rank algorithm perform well on a set of elements. In traditional search engines,

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

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

ID	Feature Description
1	$[x]$
2	# of unique words of x
3	$\sum_{t \in q \cap x} tf(t, x)$
4	$\sum_{t \in q \cap x} \log(1 + tf(t, x))$
5	$\sum_{t \in q \cap x} \frac{tf(t, x)}{[x]}$
6	$\sum_{t \in q \cap x} \log(1 + \frac{tf(t, x)}{[x]})$
7	$\sum_{t \in q \cap x} \log(\frac{[Col]+1}{df(t)+0.5})$ in x
8	$\sum_{t \in q \cap x} \log(1 + \log(\frac{[Col]+1}{df(t)+0.5}))$ in x
9	$\sum_{t \in q \cap x} \log(1 + \frac{[Col]+1}{tf(t, x)})$ in x
10	$\sum_{t \in q \cap x} \log(1 + \frac{tf(t, x)}{[x]} \times \frac{[Col]+1}{df(t)+0.5})$
11	$\sum_{t \in q \cap x} \log(1 + \frac{tf(t, x)}{[x]} \times \frac{[Col]}{tf(t, Col)})$
12	$\sum_{t \in q \cap x} \log(1 + tf(t, x)) \times \log(\frac{[Col]+1}{0.5+df(t)})$
13	$\sum_{t \in q \cap x} tf(t, x) * \log(\frac{[Col]+1}{0.5+df(t)})$
14	$\sum_{t \in q \cap x} BM25(t, x)$ with $k1 = 2$ and $b = 0.2$
15	$\sum_{t \in q \cap x} \log(1 + BM25(t, x))$ with $k1 = 2$ and $b = 0.2$
16	$\sum_{t \in q \cap x} LM_{Jelinek-Mercer}(t, x)$ with $\lambda = 0.5$
17	$\sum_{t \in q \cap x} LM_{Dirichlet}(t, x)$ with $\mu = 2500$
18	$\sum_{t \in q \cap x} LM_{AbsoluteDiscount}(t, x)$ with $\delta = 0.88$
19,20,...,25	family tag of the element

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 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 18 first features described in Table 1 for our INEX submissions. Then, we added independently features 19 to 25.

The description of the feature representation of an element in the retrieved pool for a given query is as follows:

Reference Run Set: In this set, an element x is described through 18 features (exactly as presented in Table 1) where x is the whole document (i.e. $\langle article \rangle$ element). Submissions based on this set are used as baselines and called **LIP6-OWPCRefRun***².

Element-Document and Ratio Set: In this case, we concat extracted features of Table 1 where x is a XML element retrieved in the pool and then the document ($doc(x)$) which x belongs to. In addition, for each feature

² The * corresponds to the task, in our case, “Th” for Efficiency/Thorough and “Fo” for Restricted Focused.

$feat(x)$ computed on x of Table 1, we add the ratio $r_i(x) = \frac{feat_i(x)}{feat_i(doc(x))}$. This ratio should compare the proportion of the information included in the element in 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 relevant 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 Set: In this case, we compute features when x is the retrieved XML element, then we add features of the father of x ($father(x)$) and finally, we expand by computed features on the document where x belongs to. As before, we add two ratios: $r1_i(x) = \frac{feat_i(x)}{feat_i(father(x))}$ and $r2_i(x) = \frac{feat_i(father(x))}{feat_i(doc(x))}$. and we add 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***.

Once we obtain the learning datasets thus described, we can apply a learning to rank algorithm which minimizes the ranking error focused to the top of the retrieved list.

3 OWPC, Our Learning to Rank Model

We outline here the learning to rank model described more in detail in [12]. We consider a standard setting for learning to rank. The ranking system receives as input a query q , which defines a sequence of XML elements denoted $\mathbf{X}(q) \stackrel{def}{=} (\mathbf{X}_1(q), \dots, \mathbf{X}_{[q]}(q))$ (where $[q]$ is used as a shortcut for $|\mathbf{X}(q)|$). In our case, the sequence is a subset of the whole collection filtered by the pooling technique as described in Section 2. $\mathbf{X}_j(q)$ corresponds to the feature representation, i.e the vector of features, of the j -th element. The score function f takes as input the feature representation of an element, thus $f(\mathbf{X}_j(q))$, denoted $f_j(q)$ for simplicity, and returns a real-valued score of the j -th element. In the end, the output of the ranking system is the list of the elements for a given query q sorted by decreasing scores.

The aim of a learning to rank algorithm is to learn a scoring function while minimizing the measured error on the predicted list for a given query. This ranking error must be optimizable and related to an IR evaluation measure which gives more importance at the errors made at the top of the list. In the rest of the section, we describe our model which learns a ranking function by minimizing errors made at the top of the list.

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 for a given query q ($\bar{\mathbf{y}}$ contains the indexes of the *irrelevant* ones).

Given a training set $S = (q_i, \mathbf{y}_i)_{i=1}^m$ of m examples (in our case $m = 40$, see Section 2), learning to rank consists in choosing a score function f that will minimize a given *ranking error function* $\hat{R}^\Phi(f, S)$:

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

$\hat{\mathbb{E}}_{(q, \mathbf{y}) \sim S}$ is the mean on S of ranking errors and \mathbf{err} is the number of misranked relevant XML elements 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:

$$\forall y \in \mathbf{y}, \text{rank}_y(f, q, \mathbf{y}) \stackrel{\text{def}}{=} \sum_{\bar{y} \in \bar{\mathbf{y}}} \mathbf{I}(f_y(q) \leq f_{\bar{y}}(q)) \quad (1)$$

where $\mathbf{I}(f_y(q) \leq f_{\bar{y}}(q))$ is an indicator function (returns 1 if the score of a relevant element is lower than the score of an irrelevant one and 0 otherwise). 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 $\mathbf{I}(f_y(q) \leq f_{\bar{y}}(q))$ of $\text{rank}_y(f, q, \mathbf{y})$. To solve this problem we generally take a convex upper bound of the indicator function which is differentiable and admits only one minimum. This bound is denoted $\ell(f_y(q) - f_{\bar{y}}(q))$ and is set to the hinge loss function³ $\ell : t \mapsto [1 - t]_+$ (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 \mathbf{err} of a real valued function f on (q, \mathbf{y}) as:

$$\mathbf{err}(f, q, \mathbf{y}) \stackrel{\text{def}}{=} \frac{1}{|\mathbf{y}|} \sum_{y \in \mathbf{y}} \Phi_{[\bar{\mathbf{y}}]}(\text{rank}_y(f, q, \mathbf{y}))$$

where $\Phi_{[\bar{\mathbf{y}}]}$ is an aggregation operator over the position of each relevant element in the predicted list. Traditionally, this aggregation operator $\Phi_{[\bar{\mathbf{y}}]}$ was set to the mean in learning to rank algorithms as in [8, 5]. Yet, optimizing the mean of the rank of relevant element 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, we showed that fixing $\Phi_{[\bar{\mathbf{y}}]}$ by the convex Ordered Weighted Aggregation (OWA) operators [13] 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:

³ Instead of the hinge loss function, all standard convex loss function can be used.

Definition 1 (OWA Operator [13]). Let $\alpha = (\alpha_1, \dots, \alpha_n)$ be a sequence of n non-negative numbers with $\sum_{j=1}^n \alpha_j = 1$. The Ordered Weighted Averaging (OWA) Operator associated to α , is the function $\text{owa}^\alpha : \mathbb{R}^n \rightarrow \mathbb{R}$ defined as follows:

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

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

Then we can rewrite the ranking error function as follows:

$$\text{err}(f, q, \mathbf{y}) \stackrel{\text{def}}{=} \frac{1}{[\mathbf{y}]} \sum_{y \in \mathbf{y}} \text{owa}(\text{rank}_y(f, q, \mathbf{y})) \quad (2)$$

SVM Formulation. Thus, 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 [11].

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

where the learning algorithm according to the training set S and the ranking error function $\text{err}(f, q, \mathbf{y})$ 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 $\text{err}(f, q, \mathbf{y})$.

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 $\langle \cdot, \cdot \rangle$ 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.

For this purpose, for each query, we use a fetch and browse strategy to build a pool of elements given to OWPC algorithm. As a reminder, the pool is built in two steps according the three models described in Section 2, that is BM25, LogTF and a Language Model with Absolute Discount smoothing function. First we retrieve the top k (with $k = 1,500$) articles of each model for the fetch step and then for the browse step, we extract the list of all overlapping elements (top $k' = All$) which contained at least one of the search terms. We strive to collect only small elements to enhance the precision of our system by limiting the domain to types $\in \{sec, ss, ss1, ss2, ss3, ss4, p\}$. Thus, the side effects due to waste labeling⁴ are reduced.

We fixed the parameters of our learning to rank algorithm on a validation set composed of Inex'09's queries. We set the weights of the OWA operator to be linearly decreasing as suggested by [12]. We fixed the C parameter of the equation (3) among $\{1, 10, 100\}$ using the $iP[0.01]$ score on the validation set (Inex'09's queries).

4.1 Efficiency/thorough Task

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

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

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

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

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 1,500 elements for each query are evaluated and in the case

⁴ The algorithm gives same importance to two elements with the same label (our labels are binary) as opposed to the INEX evaluation measure which depends on the number of relevant characters in them. This difference between our approach and the evaluation function results in undesirable effects.

of the Efficiency/Thorough task, where the overlap is permitted, this penalizes runs returning a lot of small elements rather than one article.

In the 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. This is encouraging for our Restricted Focused runs.

4.2 Restricted Focused Task

We present in this paragraph the performances of our models for the Restricted Focused task. For this task, our algorithm produces a list containing only small elements due to the limitation of retrieving 1,000 characters per query.

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

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

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

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 carry 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 it belongs to.

We can conclude 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 can be seen 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 significant.

5 Conclusion

In conclusion, we built a training set for our learning to rank 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 to its recall.

References

- [1] Amini, M.R., Usunier, N., Gallinari, P.: Automatic Text Summarization Based on Word-Clusters and Ranking Algorithms. In: Losada, D.E., Fernández-Luna, J.M. (eds.) ECIR 2005. LNCS, vol. 3408, pp. 142–156. Springer, Heidelberg (2005)
- [2] Aslam, J.A., Kanoulas, E., Pavlu, V., Savev, S., Yilmaz, E.: Document selection methodologies for efficient and effective learning-to-rank. In: SIGIR 2009: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 468–475. ACM, New York (2009)
- [3] Buffoni, D., Usunier, N., Gallinari, P.: LIP6 at INEX’09: OWPC for Ad Hoc Track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 59–69. Springer, Heidelberg (2010)
- [4] Burges, C.J.C., Ragno, R., Le, Q.V.: Learning to rank with nonsmooth cost functions. In: NIPS, pp. 193–200 (2006)
- [5] Cao, Y., Xu, J., Liu, T.-Y., Hang, L., Huang, Y., Hon, H.-W.: Adapting ranking SVM to document retrieval. In: Proceedings of the 29th Annual International Conference on Research and Development in Information Retrieval SIGIR 2006 (2006)
- [6] Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. In: NIPS (1997)
- [7] Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. JMLR 4, 933–969 (2003)
- [8] Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002)
- [9] Robertson, S.E., Walker, S., Hancock-Beaulieu, M., Gull, A., Lau, M.: Okapi at trec. In: TREC, pp. 21–30 (1992)
- [10] 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, LNI, vol. 103, pp. 277–291, GI (2007)
- [11] Tsochantaridis, L., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research 6, 1453–1484 (2005)
- [12] Usunier, N., Buffoni, D., Gallinari, P.: Ranking with ordered weighted pairwise classification. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) ICML. ACM International Conference Proceeding Series, vol. 382, p. 133. ACM, New York (2009)
- [13] Yager, R.R.: On ordered weighted averaging aggregation operators in multi-criteria decision making. IEEE Transactions on Systems, Man and Cybernetics (1988)
- [14] Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: SIGIR, pp. 271–278 (2007)
- [15] Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. ACM Trans. Inf. Syst. (2004)

A Useful Method for Producing Competitive Ad Hoc Task Results

Carolyn J. Crouch, Donald B. Crouch,
Sandeep Vadlamudi, Ramakrishna Cherukuri, and 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 [5] 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(s) so identified) and the subsequent application of a specific focusing technique for the Focused and Relevant-in-Context tasks (to select the focused elements). The system is based on the Vector Space Model [10]; basic functions are performed using the Smart experimental retrieval system [9]. In this paper, we report the final results of these experiments, applied to the INEX 2009 Thorough, Focused and Relevant-in-Context tasks. Results show that this 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, show in which cases statistically significant results are obtained. Our 2010 work is ongoing at present.

1 Introduction

In 2009, our INEX investigations centered on establishing a methodology for producing competitive results for the Ad Hoc Focused Task, applied in the environment of the INEX 2009 collection (the larger, scaled-up version of Wikipedia). (Results reported in [5] show that our basic approach had produced competitive focused results when applied to the smaller, 2008 Wiki collection.) As our experiments progressed, it became clear that the same basic methodology, applied in general to the Thorough Task and extended with the production of focused elements to the Focused and Relevant-in-Context Tasks, was capable of producing competitive results for all three tasks. In this paper, we report the results of these experiments, along with the

corresponding significance tests which establish that results produced by this method are competitive with those of the top-ranked participants for each task.

Dynamic element retrieval—i.e., the dynamic retrieval of elements at the desired degree of granularity—has been the focus of our INEX investigations for some years. The incorporation of dynamic element retrieval with traditional document retrieval enables us to generate, with respect to the query, a rank-ordered set of all its elements for each document of interest. Thorough elements (i.e., all the [overlapping] elements along a path) are extracted from this set. These elements are then filtered to produce the corresponding set of focused (i.e., non-overlapping) elements, thus yielding results for the Focused and Relevant-in-Context Tasks. A more detailed view is given below.

2 Dynamic Element Retrieval and the Ad Hoc Tasks

In this section, we describe our methodology for producing, for each query, a set of elements of interest with respect to the three major INEX Ad Hoc Tasks. The experiments performed using this approach are detailed and their results reported in Section 3. Basic functions are performed using Smart. The queries are taken from the *title* field.

2.1 Producing the Element Set

Given any one of the three major Ad Hoc tasks, our concern is to produce, for each query, a set of elements as required for that specific task. The approach described here incorporates article retrieval (to identify the articles of interest) with dynamic element retrieval (to produce the set of elements associated with each article). These results are then either reported or further processed as required by the task. A lower-level view of the process follows.

For each query, we retrieve n articles or documents. We then use dynamic element retrieval to produce the elements themselves. Dynamic element retrieval [3, 7] builds the document tree at execution time, based on (1) a stored schema representing a pre-order traversal of the document, created when it is parsed, and (2) a terminal node index of the collection. Given a specific article, our document-building routine, Flex, identifies its document tree and then *seeds* the tree by connecting each of its terminal nodes to the vector representing that node in the terminal node index. (In this context, the terminal node represents a leaf of the document tree; the content of this node contributes, with that of its siblings, to form the element vector of the parent node.) Since the articles are semi-structured, all untagged text is collected and becomes a new [artificial] child node of its parent. (These nodes are included in the terminal node index.) Given all the terminal nodes associated with a parent, Flex then builds the parent node and the process continues until each element vector in the document tree has been generated, bottom up, from the terminal node level to the body node of the article.

All of the element vectors thus generated, along with the query vector in question, are term-frequency weighted. To determine the correlation between the query and the element vectors, we use inner product with *Lnu-ltu* term weighting [11], designed by

Singhal to deal with differences in vector length (which would otherwise bias results in favor of long vectors [e.g., those representing articles or sections] over short ones [e.g., paragraphs]). *Lnu-ltu* weighting requires the use of two collection-dependent parameters, slope and pivot. (All-element values of slope and pivot are used here.) Correlation of the *ltu*-weighted query with the *Lnu*-weighted element vectors of the document tree produces a rank-ordered list of elements from the document.

2.2 Producing Task-Specific Output

Thorough results are now available. For the top-ranked article identified by document retrieval, we output the set of all Thorough elements (all overlapping elements along a path), do the same for the second-ranked document, and continue until either all n documents are processed or the window is full. Focused results require additional filtering to select the most relevant element along a path. We use one of the three focusing strategies described below to remove overlap. A set of focused elements, $\leq m$ in size, is then reported for each of the top n documents. (Here m represents the number of focused elements reported for each query.) These same results, i.e., focused elements output by document, are reported for the Relevant-in-Context Task.

Three focusing or overlap removal strategies were investigated in these experiments. The *section strategy* chooses the highest correlating non-body element along a path as the focused element. (Most of these elements turn out in fact to be sections. A body element appears in this list of focused elements if and only if none of its child elements are ranked within the top m .) The *correlation strategy* chooses the highest correlating element along a path as the focused element, without restriction on element type. And the *child strategy* chooses the terminal element along a path as the focused element (i.e., ignores correlation and always gives preference to the child rather than the parent).

2.3 Early Results

Our earlier work in 2008 centered on the Focused Task. The best results achieved in those experiments (reported in [1, 4]) were competitive with those in the upper ranges of the official ranking. In 2009, our goal was to confirm that the same methodology could produce competitive results when applied to the new (2009) Wikipedia collection. But problems with *xpaths* (a result of an early decision to generate simplified versions of the document trees—i.e., reduce the tag set) meant that tags present in the new Wiki Collection were not always present in the element path, thus making the proper evaluation of these elements impossible. In this paper, having corrected this *xpath* problem, we present the results of experiments applying our methodology to the 2009 Wikipedia Collection for the Thorough, Focused, and Relevant-in-Context Tasks and compare the results to those in the official rankings. Significance testing of these results is addressed in Section 4.

3 Experiments and Results

The description of our experiments when applied to the Thorough, Focused, and Relevant-in-Context Ad Hoc Tasks and the results produced are described here in Sections

3.1, 3.2, and 3.3, respectively. All of these experiments were performed using the INEX 2009 collection. Significance testing, observations and analysis follow in Section 4.

In these experiments, n represents the number of articles retrieved and m is the number of elements reported. In all cases, n , the number of articles, ranges from 25 to 500, and m , the number of elements, ranges from 50 to 1500. Slope and pivot values are 0.11 and 44, respectively.

3.1 Thorough Task

Recall the basic approach described above. Given a query, the top-ranked n articles are identified based on Smart retrieval. Dynamic element retrieval is then used to build the document trees. As the trees are built, bottom up, each Lnu -weighted element vector is correlated with the ltu -weighted query using inner product. For each tree, a rank-ordered list of elements is produced. This list includes elements representing untagged text in the document, which must be present in order to generate the trees properly but which do not physically exist as elements per se and so are removed from the element list before either thorough or focused elements are reported.

Table 1 displays the results of Thorough Task evaluation for the 2009 INEX collection.

Table 1. MAiP 2009 Thorough Task Evaluation

	50	100	150	200	250	500	1000	1500
25	0.2062	0.2069	0.2081	0.2081	0.2081	0.2081	0.2081	0.2081
50	0.2062	0.2071	0.2081	0.2085	0.2085	0.2085	0.2085	0.2085
100	0.2063	0.2071	0.2082	0.2086	0.2095	0.2095	0.2095	0.2095
150	0.2063	0.2073	0.2084	0.2087	0.2096	0.2098	0.2098	0.2098
200	0.2064	0.2074	0.2083	0.2088	0.2098	0.2099	0.2105	0.2105
250	0.2064	0.2075	0.2085	0.2091	0.2097	0.2101	0.2109	0.2109
500	0.2065	0.2077	0.2085	0.2091	0.2098	0.2103	0.2109	0.2120

The best result is obtained here as expected at $n=500$ and $m=1500$. The MAiP score of 0.2120 places this run at rank 8 with respect to the official results.

3.2 Focused Task

By definition, overlapping elements must be removed from the element set identified in Section 3.1 (above) to produce the set of focused elements for each document tree. The focused elements from each tree are then reported, in article order, for evaluation. The approach used here guarantees that if m focused elements are available (i.e., retrieved by the query), the top-ranked m are reported.

The results of these experiments for each of the three focusing strategies (described in Section 2.2) are given below in Tables 2, 3, and 4, respectively. The best result for each focusing strategy is again obtained at $n=500$ and $m=1500$. The best $iP[0.01]$ value is 0.6594 for the Section strategy. The best result from each of these strategies ranks above the best value reported in the official ranking (Waterloo at 0.6333). Statistical significance is addressed below. (See Section 4.)

Table 2. $iP[0.01]$ 2009 Focused Task – Section Strategy

	50	100	150	200	250	500	1000	1500
25	0.6462	0.6489	0.6539	0.6539	0.6539	0.6539	0.6539	0.6539
50	0.6465	0.6493	0.6539	0.6546	0.6557	0.6557	0.6557	0.6557
100	0.6465	0.6492	0.6537	0.6544	0.6556	0.6573	0.6573	0.6573
150	0.6463	0.6494	0.6539	0.6545	0.6558	0.6582	0.6585	0.6585
200	0.6459	0.6493	0.6538	0.6546	0.6556	0.6581	0.6589	0.6593
250	0.6464	0.6493	0.6538	0.6546	0.6556	0.6581	0.6587	0.6594
500	0.6465	0.6490	0.6539	0.6546	0.6557	0.6580	0.6588	0.6594

Table 3. $iP[0.01]$ 2009 Focused Task – Correlation Strategy

	50	100	150	200	250	500	1000	1500
25	0.6374	0.6392	0.6418	0.6431	0.6431	0.6431	0.6431	0.6431
50	0.6375	0.6394	0.6421	0.6442	0.6442	0.6442	0.6442	0.6442
100	0.6375	0.6394	0.6423	0.6445	0.6459	0.6459	0.6459	0.6459
150	0.6374	0.6396	0.6423	0.6445	0.646	0.6469	0.6469	0.6469
200	0.6374	0.6396	0.6423	0.6445	0.6461	0.647	0.6484	0.6484
250	0.6474	0.6396	0.6423	0.6445	0.6461	0.647	0.6487	0.6488
500	0.6475	0.6395	0.6422	0.6445	0.6459	0.6471	0.6487	0.6488

Table 4. $iP[0.01]$ 2009 Focused Task – Child Strategy

	50	100	150	200	250	500	1000	1500
25	0.6351	0.6377	0.6428	0.6428	0.6428	0.6428	0.6428	0.6428
50	0.6352	0.6379	0.6430	0.6452	0.6464	0.6464	0.6464	0.6464
100	0.6351	0.6379	0.6430	0.6454	0.6464	0.6472	0.6472	0.6472
150	0.6351	0.6379	0.6431	0.6454	0.6464	0.6476	0.6478	0.6478
200	0.6353	0.6378	0.6431	0.6454	0.6467	0.6476	0.6479	0.6481
250	0.6462	0.6379	0.6430	0.6453	0.6467	0.6476	0.6479	0.6482
500	0.6352	0.6379	0.6430	0.6453	0.6465	0.6475	0.6480	0.6482

3.3 Relevant-in-Context Task

For this task, we use the same methodology described in Section 3.2 (above) to produce and then report the focused elements. The resulting list is then evaluated using MAgP. Results for each of the three focusing strategies is given in Tables 5, 6, and 7 below.

Table 5. MAgP 2009 Relevant-in-Context Task – Section Strategy

	50	100	150	200	250	500	1000	1500
25	0.1076	0.1104	0.1129	0.1129	0.1129	0.1129	0.1129	0.1129
50	0.1098	0.1138	0.1152	0.1179	0.1179	0.1179	0.1179	0.1179
100	0.1102	0.1224	0.1268	0.1292	0.1331	0.1358	0.1358	0.1358
150	0.1103	0.1341	0.1392	0.1427	0.1448	0.1496	0.1501	0.1501
200	0.1103	0.1462	0.1444	0.1494	0.1505	0.1509	0.1523	0.1529
250	0.1103	0.1573	0.1521	0.1542	0.1536	0.1552	0.1571	0.1589
500	0.1103	0.1619	0.1593	0.1587	0.1594	0.1606	0.1627	0.1636

Table 6. MAgP 2009 Relevant-in-Context Task – Correlation Strategy

	50	100	150	200	250	500	1000	1500
25	0.1124	0.1154	0.1186	0.1154	0.1154	0.1154	0.1154	0.1154
50	0.1165	0.1201	0.1245	0.1256	0.1256	0.1256	0.1256	0.1256
100	0.1177	0.1386	0.1404	0.1445	0.151	0.1472	0.1472	0.1472
150	0.1178	0.1492	0.1511	0.1562	0.1595	0.1522	0.1531	0.1531
200	0.1179	0.1577	0.1548	0.1588	0.1627	0.1614	0.1643	0.1654
250	0.1179	0.1656	0.1594	0.1621	0.1665	0.1651	0.1695	0.1696
500	0.1179	0.1699	0.1613	0.1645	0.1692	0.1687	0.1722	0.1731

Table 7. MAgP 2009 Relevant-in-Context Task – Child Strategy

	50	100	150	200	250	500	1000	1500
25	0.1094	0.1126	0.1146	0.1146	0.1146	0.1146	0.1146	0.1146
50	0.1123	0.1178	0.1195	0.1221	0.1221	0.1221	0.1221	0.1221
100	0.1131	0.1295	0.1328	0.1395	0.1461	0.1469	0.1469	0.1469
150	0.1131	0.1386	0.1471	0.1476	0.1521	0.1504	0.1513	0.1513
200	0.1131	0.1503	0.1496	0.1501	0.1543	0.1514	0.1552	0.1558
250	0.1131	0.1625	0.1542	0.1565	0.1579	0.1605	0.1633	0.1651
500	0.1131	0.1657	0.1602	0.1616	0.1630	0.1654	0.1675	0.1689

Each strategy produces a best value that would place in the top 10 of the official ranking (i.e., Correlation at 5, Child at 6, and Section at 7). The best result, with an MAGP value of 0.1731, is produced by the Correlation strategy.

With respect to the data reported in Tables 1-7, we note that (for each task) m represents the total number of elements reported with respect to n documents. At 1500, the window is filled (if enough elements of the required type have been produced by the n top-ranked documents to fill it). Indications are that guaranteeing, for each query, a value of n sufficiently large to fill the reporting window may produce slightly improved results, but we have not done that here.

4 Analysis and Observations

We compared our experimental results for the 2009 Thorough, Focused, and Relevant-in-Context tasks with the baseline runs of the ten top-ranked participants. We utilized a t-test, one-tailed, at 95% - the approach used by Kamps for significance testing of participant scores in the Ad Hoc Track of INEX 2008 [6]. The same testing, applied to our 2008 Ad Hoc results [4], is detailed in [2] and reveals a very similar picture to that of 2009, which is summarized below.

Significance testing of 2009 Thorough Task results reveals the following: Our results are significantly better than those of participants ranked 9-10. There is no statistically significant difference between our results and those of participants in the mid-range, and only the top-ranked participant, LIG, produced a significantly better result for this task.

With respect to the 2009 Focused Task, although our scores for all three focusing methods rank above those of the baseline (i.e., participant scores), there is no statistically significant difference between (1) results produced by the three focusing methods or (2) results produced by our methods versus those of the baseline participants.

Consider now the 2009 Relevant-in-Context Task. Testing shows that our results are significantly better than those of participants ranked 6-10. There is no difference between our results and those of participants ranked 2-5, but the result produced by the top-ranked participant, Queensland, is significantly better than ours.

In evaluating the results of our methodology, we find that it produces results that are (1) statistically equivalent to those of the top ten participants for the Focused Task and (2) either superior or equivalent to 9 of the top 10 participants for the Thorough and Relevant-in-Context Tasks. Two participants produced superior results: LIG with the top-ranked Thorough run and Queensland with the top-ranked Relevance-in-Content run.

We began our Focused experiments several years ago with the aim of producing superior elements, by which we meant producing results which, when evaluated, would rank in the top 10 (or, from our viewpoint, could claim to be in good company). Better expressed, our objective has been to produce competitive elements. Results from both 2008 and 2009 show that this objective has now been met. (Details of these experiments for both 2008 and 2009 may be found in [2, 8, 12].) Furthermore, a single methodology has been applied to yield such results for all three Ad Hoc tasks. Our current ongoing work is directed at extracting good snippets from these elements.

References

- [1] Bapat, S.: Improving the results for focused and relevant-in-context tasks. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth (2008), <http://www.d.umn.edu/cs/thesis/bapat.pdf>
- [2] Cherukuri, R.: Significance testing for the INEX 2008-2009 Ad Hoc Tracks. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth (2010), <http://www.d.umn.edu/cs/thesis/cherukuri.pdf>
- [3] Crouch, C.: Dynamic element retrieval in a structured environment. *ACM TOIS* 24(4), 437–454 (2006)
- [4] Crouch, C.J., Crouch, D.B., Bapat, S., Mehta, S., Paranjape, D.: Finding Good Elements for Focused Retrieval. In: Geva, S., Kamps, J., Trotman, A. (eds.) *INEX 2008*. LNCS, vol. 5631, pp. 33–38. Springer, Heidelberg (2009)
- [5] Crouch, C.J., Crouch, D.B., Bhirud, D., Poluri, P., Polumetla, C., Sudhakar, V.: A Methodology for Producing Improved Focused Elements. In: Geva, S., Kamps, J., Trotman, A. (eds.) *INEX 2009*. LNCS, vol. 6203, pp. 70–80. Springer, Heidelberg (2010)
- [6] Kamps, J., Geva, S., Trotman, A.: Analysis of the INEX 2009 Ad Hoc Track Results. In: Geva, S., Kamps, J., Trotman, A. (eds.) *INEX 2009*. LNCS, vol. 6203, pp. 26–48. Springer, Heidelberg (2010)
- [7] Khanna, S.: Design and implementation of a flexible retrieval system. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth (2005), <http://www.d.umn.edu/cs/thesis/khanna.pdf>
- [8] Mahule, A.: Improving results of the INEX Thorough Task. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth (2010), <http://www.d.umn.edu/cs/thesis/mahule.pdf>
- [9] Salton, G. (ed.): *The Smart Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Englewood Cliffs (1971)
- [10] Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. *Comm. ACM* 18(11), 613–620 (1975)
- [11] Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: *Proc. Of the 19th Annual International ACM SIGIR Conference*, pp. 21–29 (1996)
- [12] Vadlamudi, S.: Producing improved results for the INEX Focused and Relevant-in-Context Tasks. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth (2009), <http://www.d.umn.edu/cs/thesis/vadlamudi.pdf>

Relaxed Global Term Weights for XML Element Search

Atsushi Keyaki^{1,*}, Kenji Hatano², and Jun Miyazaki³

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. XML element search engines return XML elements which are part of XML documents as search results. Existing studies related to XML element search are brought from the information retrieval techniques for document search. There are some ways to calculate global weights of each term from statistics of XML elements with 1) the same path expression or 2) the same tag. In the first approach, the more complex a path expression is, the less the number of XML elements with the path expression becomes. This is a problem that global term weights may be calculated using statistics of a few XML elements. Such global weights are never *global*. The second approach also has a problem that it does not consider document structures of XML elements. To resolve the problems, we propose a method for calculating accurate global weights. In our method, we regard a path expression as an array of tags. We relax the restriction of appearance order and appearance frequency of tags in a path expression to gather similar path expressions into the same class. Therefore, we try to decrease the number of classes which hardly contain elements. Our experimental results show that our method can integrate path expressions without decreasing search accuracy with a certain test collection.

Keywords: XML element search, accurate global term weights, classification of similar path expressions.

1 Introduction

The XML¹ is a markup language for structured documents that has become the de facto format for data exchange. A large number of XML documents are now available on the Web, so that it continues to be used in the future.

* Current affiliation: Graduate School of Information Science, Nara Institute of Science and Technology.

¹ <http://www.w3.org/TR/REC-xml/>

An XML element is usually defined as a part of an XML document. The XML element is identified by the surrounding start and end tags. The key goal of XML search is to obtain relevant XML elements to a query instead of just returning the entire XML documents, in particular, with document-centric XML documents [2]. Therefore, XML search engines can generate a ranked list composed of a set of relevant XML elements, while several Web search engines return a set of entire Web documents. By searching XML elements, users do not need to identify relevant descriptions from entire XML documents.

In the field of XML element search, existing term weighting schemes are often brought from the information retrieval techniques of document search. There are two approaches which are often used to calculate global weights of each term. One uses statistics of XML element with the same path expression, while the other uses the statistics of the same tag. In the first approach, the more complex a path expression is, the less the number of XML elements with the path expression becomes. This is one of the problems that global weights may be calculated using statistics of a few elements. Such global weights are not *global*. The second approach also has a problem that it does not consider document structure of XML elements.

To resolve these problems, we should calculate global weights for each term from statistics of XML elements which are *global* enough with considering structures of the XML elements. Therefore, we integrate similar path expressions into the same class and calculate global weights based on the classes. In this paper, we define a *class* as classification with a certain property. To integrate them, we regard a path expression as an array of tags and identify path expressions which are similar to each other in terms of appearance order or appearance frequency of tags. As a result of the integration, we try to decrease the classes which do not contain enough XML elements to calculate accurate global weights. In other words, we propose a method to calculate the relaxed global weights by integrating similar path expressions.

This paper is organized as follows. In Section 2, we explain about information retrieval technique for XML element search and related studies. In Section 3, we propose a new method to calculate relaxed global weights. Experiments results are shown in Section 4, followed by concluding remarks and future work in Section 5.

2 XML Element Search Techniques and Related Studies

In this section, we describe existing XML element search techniques and their ways of calculating global weights. In addition, we also discuss on a related study which relaxes the query restriction related to document structures of XML elements.

2.1 An Expansion from Document Search to Element Search

Information retrieval techniques of XML element search often make use of the ones used in document search. In general, term weighting schemes of document

search utilize some kinds of the statistics, for example, local weights of each term, global weights of each term, normalization by document length, and statistics given by entire document collection [8]. Term weighting schemes of element search also utilize these statistics; however, we should treat global weights carefully. In the term weighting schemes of document search, we use all documents when we calculate global weights. This is because document is a unit for calculating global weights, that is, all documents are in a class. In element search, on the other hand, each element belongs to one of the classes; therefore, most term weighting schemes of element search are not based on elements but on the classes to which the elements with the same and/or similar path expressions belong. Now, the question is how we identify the “same class.” In the next subsection, we refer to some of the most popular XML element search techniques and their standard classification of the same class.

2.2 Existing Methods of Calculating Global Weights

TF-IPF [3], a popular term weighting scheme of XML element search, is a path-based term weighting scheme that extends the well-known TF-IDF [12] approach for document search with the vector space model. In TF-IDF, local weights are calculated by the term frequency (TF) of each term in each document and global weights use inverse document frequency (IDF) of each term in all documents. Finally, a weight of TF-IDF is derived by the product of TF and IDF. In TF-IPF, global weights use inverse path frequency (IPF) which is the inverse of element frequency with the same path expression, while local weights are calculated in the same manner as TF-IDF. A weight of TF-IPF is, then, calculated by the product of TF and IPF. Furthermore, there are some studies which refine TF-IPF into more accurate one. Normalized TF-IPF [6] is one of them and it normalizes each element in its length. These term weighting schemes treat that the XML elements with the same path expression belong to the same class. In other words, if there are the elements which have different path expressions, these elements are classified to different classes. Hence, if the path expressions of elements become more complex, the number of classes increases. In this case, appropriate global weights cannot be calculated, because each class does not have enough XML elements. Therefore, we should consider how to decrease the classes which do not contain enough XML elements to calculate global weights.

Other popular approaches for XML element search are BM25E [7], which is based on Okapi’s BM25 [11] term weighting scheme of document search with probabilistic model, and BM25F [10] for field search. Information retrieval techniques for field search are based on a document as a search unit, and used for structured document. BM25F gives a weight to each tag to consider the field that it appears by adjusting important degrees of each tag².

Meanwhile, BM25E is an information retrieval technique of element search like TF-IPF. There are two ways to calculate global weights where 1) all elements belong to the same class (inverse element frequency, IEF), and 2) the elements

² For example, the weight of `title` tag is given high value, because the element which includes `title` tag tends to be relevant if query keywords appear in such tags.

with the same tag belong to the same class (inverse tag frequency, ITF) [9]. We overview the problems of these approaches. IEF is calculated by counting text nodes repeatedly because there are overlaps among elements in the ancestor-descendant relationship. This means that documents containing numerous tags become influential too much. Even if the feature of each tag is considered in ITF, it does not consider the document structures of XML elements. Since both keywords and structures are used as a query in XML search, it is not appropriate to ignore document structures. Therefore, IEF and ITF are not suitable for global weights.

2.3 Relaxing Restriction of Document Structure in a Query

Keeping above points in mind, we should integrate path expressions if they are supposed to be in the same class. There is an existing study which relaxes query restrictions in terms of document structures [4]. This approach first calculates the global weights in advance based on the path-based classification, i.e., IPF, and then, calculates the term weights of query keywords based on the elements that satisfy query restrictions when the query is posted. However, we cannot identify which class the path expressions included in a query are categorized to before the query is given. Since the class classification is determined by a query, it takes longer search time because the term weights are calculated after the query is posted.

3 Classification of Similar Path Expressions

Concerning the problems of the existing methods of calculating global weights and the related studies of relaxing query restriction introduced in Section 2, we should develop new strategies to decrease the classes each of which contains enough XML elements for calculating global weights. It is better if the classes are classified before a query is processed. To satisfy these requirements, we integrate the path expressions which are similar in terms of their document structures in XML elements. Therefore, we propose a method to calculate relaxed global weights by integrating path expressions.

We define a path expression as an array of tags whose appearance order and appearance frequency of tags in the location steps are considered. Hence, we can relax the restriction of document structures in the order and frequency by our proposed classification, while existing approaches are based on path-based ones. Here, we propose the following three ways for calculating global weights by integrating path expressions; 1) relaxing appearance order of tags, 2) relaxing appearance frequency of tags, and 3) both relaxing appearance order and frequency.

3.1 Inverse Combination Frequency

Concrete examples of path integration with path expressions are shown in Fig. 1. First, we explain inverse combination frequency (ICF) which relaxes the appearance order of tags in path expressions. Tags in structured documents are

1: /article/sec
 2: /article/sec/sec
 3: /article/sec/person/sec
 4: /article/sec/p/
 5: /article/person/sec
 6: /article/sec/sec/person
 7: /article/person/sec/sec
 8: /article/sec/sec/p

Fig. 1. Examples of path expressions

article: 1, sec: 1	1: /article/sec
article: 1, sec: 2	2: /article/sec/sec
article: 1, sec: 2, person: 1	3: /article/sec/person/sec 6: /article/sec/sec/person 7: /article/person/sec/sec
article: 1, sec: 1, sep: 1	4: /article/sec/p/
article: 1, sec: 1, person: 1	5: /article/person/sec
article: 1, sec: 2, sep: 1	8: /article/sec/sec/p

Fig. 2. An example of classification in ICF

/article+/sec+	1: /article/sec 2: /article/sec/sec
/article+/sec+/person+/sec+	3: /article/sec/person/sec
/article+/sec+/p+	4: /article/sec/p/ 8: /article/sec/sec/p
/article+/person+/sec+	5: /article/person/sec 7: /article/person/sec/sec
/article+/sec+/person+	6: /article/sec/sec/person

Fig. 3. An example of classification in IAF

categorized into two types of tags. One represents structural classification, and the other indicates something ideas, attributes, specific contents, etc. These two types of tags are supposed to be independent in their appearance. This suggests that a combination of tags can consist of two or more path expressions. However, it is not appropriate that these path expressions are classified to different classes. In ICF, we, therefore, do not consider the order of tags strictly but the combination of tags, i.e., names and frequency of tags.

To illustrate a concrete example of classification by ICF, a classification result of path expressions in Fig. 1 by ICF is shown in Fig. 2. We preliminarily enumerate the names and frequency of tags in each path expression to integrate path expressions classified as the same class. As a consequence, we integrate path expressions 3, 6, and 7 because all of them have one `article` tag, two `sec` tags, and one `person` tag. The global weights are calculated by all XML elements in path expression 3, 6, and 7 because they are in the same class.

3.2 Inverse Aggregated-Path Frequency

We explain inverse aggregated-path frequency (IAF) which relaxes appearance frequency of tags in a path expression. In some path expressions, a tag appears consecutively twice or more times, for example, `col` tags in `table` tag of HTML. In this case, even if the frequencies of the same tag appearing consecutively are different, we suppose that the features of a path expression are not so different because the semantics of each tag is fixed. Therefore, if consecutive tags are the same, such tags can be aggregated into one.

Consequently, we do not strictly consider the frequency of tags but their order, i.e., names and order of tags in IAF. In the same manner as ICF explained in Section 3.1, a classification result of path expressions in Fig. 1 by IAF is shown in Fig. 3. When the same tag consecutively appears twice or more times, such a tag is aggregated preliminarily. For example, since `sec` tags appear in path expression 2, 6, 7, and 8, path expressions 1 and 2, 4 and 8, 5 and 7 are integrated. This is because path expressions 1 and 2 have one or more `article` tags followed by one or more `sec` tags while expressions 4 and 8 have one or more `article` tags followed by one or more `sec` tags, and one or more `p` tags. Path expressions 5 and 7 also have one or more `article` tags followed by one or more `person` tags, and one or more `sec` tags.

3.3 Inverse Set Frequency

We described the way of integrating path expressions in terms of either appearance order or appearance frequency of tags in Section 3.1 and 3.2. In contrast, inverse set frequency (ISF) relaxes both appearance order and frequency of tags in path expressions. Accordingly, we do not consider the order and frequency of tags but the names of tag in ISF. Therefore, we classify the path expressions which are composed of the same tag name as the same class.

A classification result of path expressions in Fig. 1 by ISF is shown in Fig. 4. Path expressions 1 and 2 are in the same class because they are both composed of `article` and `sec` tags. Path expressions 3, 5, 6, and 7 are composed of `article`,

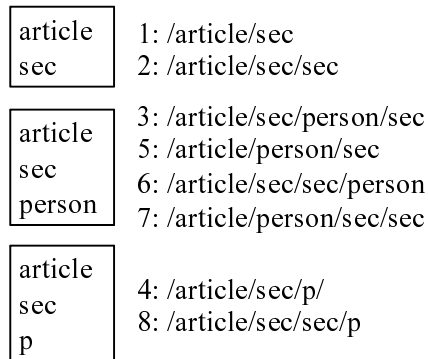


Fig. 4. An example of classification in ISF

Table 1. Effects of integrating path expressions in the INEX 2008 test collection

method	number of classes	the ratio compared to IPF
ICF	56,369	.85
IAF	32,421	.49
ISF	16,007	.24
IPF	66,210	1.0
ITF	495	.0075

Table 2. Effects of integrating path expressions in the INEX 2010 test collection

method	number of classes	the ratio compared to IPF
ICF	22,743,778	.97
IAF	23,383,388	.99
ISF	19,587,224	.83
IPF	23,502,448	1.0
ITF	22,110	.000094

`sec`, and `person` tags, while path expressions 4 and 8 are composed of `article`, `sec`, and `p` tags.

4 Experiments

In this section, we show some experimental results with the INEX 2008 test collection [5] and the INEX 2010 test collection [1], so as to confirm the usefulness of our proposed methods. In more detail, we conducted three kinds of experiments; 1) assessing the effectiveness of integrating path expressions, 2) analyzing the number of XML elements in each class, and 3) evaluating search accuracy.

4.1 Experiments for Integrating Path Expressions

Table 1 and 2 explain that how many path expressions are integrated. Each table shows the number of classes of each method and their ratios compared to IPF.

First, we discuss on Table 1. The result shows that the proposed methods could appropriately integrate the path expressions in the INEX 2008 test collection. IAF which relaxes appearance frequency could integrate more effectively than ICF which relaxes appearance order. The number of classes of ISF decreased to a quarter compared to IPF.

Table 3. The number of XML elements in a class and the ratio of the classes in the INEX 2008 test collection

numbers of elements	IPF	ITF	ICF	IAF	ISF
1	.57	.58	.53	.42	.37
2	.13	.14	.14	.16	.13
3	.062	.055	.067	.080	.063
4	.041	.026	.042	.057	.052
5	.027	.018	.028	.031	.036
6	.021	.0080	.022	.027	.032
7	.014	.010	.017	.017	.019
8	.014	.0040	.014	.019	.017
9	.011	.010	.012	.012	.013
10	.090	.0060	.010	.011	.010
11 or more	.10	.14	.12	.17	.25

In contrast, the proposed methods did not integrate path expressions effectively in the INEX 2010 test collection as shown in Table 2. Though both ICF and IAF did not work well, it seemed to be effective for ISF comparatively. This indicates that our methods did not seem to be sufficient. In particular, we need to consider that whether IAF which hardly integrates path expressions works well or not.

The reason of the result obtained was due to growth in the number of kinds of tags included in the INEX 2010 test collection. This causes an increase in the number of the combination of path expressions. As a result, we could not integrate path expressions effectively. Therefore, we should consider another condition for relaxing path expressions. For example, we need to screen valid tags, while we did not consider in the proposed methods. Because we supposed that the classes of path expressions are mainly based on the *structural tags*, it might be reasonable that we do not use *content tags* but the structural tags only. Otherwise, it might also be reasonable to integrate the tags with the same semantics. We should further consider a new relaxation method because the proposed relaxing approaches are not well sufficient.

We conducted some more experiments in the next section to investigate the results.

4.2 Analyzing the Number of XML Elements in Classes

The experimental result in Section 4.1 indicated that the usefulness of the proposed methods depends on the test collection. We, then, analyze whether the proposed methods can reduce the number of classes which do not contain enough XML elements to calculate global weights. Table 3 and 4 represent the number of XML elements in each class and the ratios of the number of the classes to that of all classes.

Table 3 explains that all of the proposed method reduce the ratio of the classes which contain only an element compared to IPF in the INEX 2008 test

Table 4. The numbers of XML elements in a class and the ratio of classes in the INEX 2010 test collection

numbers of elements	IPF	ITF	ICF	IAF	ISF
1	.65	.61	.66	.65	.65
2	.13	.14	.11	.13	.12
3	.047	.053	.050	.048	.052
4	.028	.031	.029	.028	.031
5	.017	.018	.018	.017	.020
6	.013	.016	.015	.013	.017
7	.015	.012	.013	.015	.0084
8	.011	.012	.0088	.011	.0070
9	.0061	.0052	.0059	.006	.0033
10	.0057	.0078	.0059	.006	.010
11 or more	.081	.10	.081	.081	.086

collection. In addition, the ratio of the classes which contain eleven or more XML elements increases in all our methods. For these reasons, we conclude that we could reduce the ratio of the classes which contain few elements by relaxing restrictions of appearance order and appearance frequency of tags in path expressions. Therefore, it is natural that ISF which relaxes both order and frequency is the most effective as our goal. Although it is arguable how many XML elements are enough to appropriately calculate global weights, we verified some effects on reducing the ratio of the classes containing few XML elements when path expressions are integrated.

Moreover, it is notable that a lot of classes in ITF have only a few elements. Though the INEX 2008 test collection has 495 kinds of tags, 298 kinds of tags (58%) appear only once. In other words, only a few tags are used repeatedly in all documents. It suggests that we should focus on the low-frequent tags to improve our approach.

As expected, we could not observe difference between our methods and IPF when using the INEX 2010 test collection, as long as we see Table 4. We should take another ways to achieve our goal, as mentioned in Section 4.1.

4.3 Evaluation on Search Accuracies

We examined search accuracy of three term weighting schemes by using the global weights obtained by our proposed methods.

More precisely, we used BM25E³ by varying global weights. In addition, we also examined search accuracy of four more methods, three existing methods and a method without global weight, to compare to our proposed methods. Note that we evaluated only with the INEX 2008 test collection⁴ because the proposed methods did not work well with the INEX 2010 test collection.

³ The weight of term j in an XML element i is calculated as follows:

$$\frac{(k_1+1)t_{f_{i,j}}}{k_1((1-b)+b\frac{c_l}{\text{ave}l})+t_{f_{i,j}}} \log \frac{N-df_i+0.5}{df_i+0.5}$$
 [7]D.

⁴ We used 68 queries in the experiment.

Table 5. Search accuracy of our proposed methods

method	iP[.01]	MAiP
ICF	.6169	.1713
IAF	.6178	.1724
ISF	.6166	.1716
IPF	.6146	.1723
IEF	.6107	.1321
ITF	.6135	.1719
no global weights	.2364	.04689

We show search accuracy of these methods in Table 5. All of the proposed method slightly improved search accuracy. In other words, they do not affect the accuracy. Since the search accuracies of IEF and ITF are comparatively lower, we need to consider the document structures of XML elements. Furthermore, we should give proper global weights to each term, because the method without global weights decreased its search accuracy significantly.

Both ICF and IAF could improve search accuracy. However, ISF reduced its search accuracy. It suggests that the deterioration of search accuracy might be caused by excessive relaxation.

We summarize the experiments in Section 4.1 through 4.3. Our methods do not work well with the INEX 2010 test collection but are effective with the INEX 2008 test collection. In the INEX 2008 test collection, ISF is the most effective approach in terms of the integration of path expressions, while IAF is the most effective one in terms of the search accuracy.

5 Conclusion

In this paper, we proposed methods to calculate relaxed global weights for XML element search.

In these methods, we integrate path expressions which are similar in terms of the order and frequency of tags to reduce the number of classes containing only a few elements. Our methods could reduce the ratio of such classes with the INEX 2008 test collection, whereas they do not work well with the INEX 2010 test collection. The experimental evaluations with the INEX 2008 test collection indicated that we could attain more accurate search. However, the results also showed that it might cause deterioration of search accuracy by excessive relaxation.

As future works, we should consider how we treat low-frequent tags, and more precisely explore to reveal why the proposed methods did not work well with the INEX 2010 test collection.

Acknowledgements. This work was supported in part by MEXT (Grant-in-Aid for Scientific Research on Priority Areas #21013035, #22240005, and #22700248).

References

1. Arvola, P., Geva, S., Kamps, J., Schenkel, R., Trotman, A., Vainio, J.: Overview of the INEX 2010 Ad Hoc Track. In: INEX 2010 Workshop Pre-proceedings, pp. 11–40 (December 2010)
2. Blanken, H., Grabs, T., Schek, H.-J., Schenkel, R., Weikum, G.: Intelligent Search on XML Data: Applications, Languages, Models, Implementations, and Benchmarks. LNCS, vol. 2818. Springer, Heidelberg (2003)
3. Grabs, T., Schek, H.-J.: PowerDB-XML: A Platform for Data-Centric and Document-Centric XML Processing. In: Bellahsene, Z., Chaudhri, A.B., Rahm, E., Rys, M., Unland, R. (eds.) XSym 2003. LNCS, vol. 2824, pp. 100–117. Springer, Heidelberg (2003)
4. Hatano, K., Amer-yahia, S., Srivastava, D.: Document-Scoring, for XML Information Retrieval using Structural Condition of XML Queries. In: IEICE technical report, pp. 13–18, DE2007-117 (October 2007)
5. Kamps, J., Geva, S., Trotman, A., Woodley, A., Koolen, M.: Overview of the INEX 2008 Ad Hoc Track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 1–28. Springer, Heidelberg (2009)
6. Liu, F., Yu, C., Meng, W., Chowdhury, A.: Effective Keyword search in Relational Databases. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 563–574. ACM, New York (2006)
7. Lu, W., Robertson, S., MacFarlane, A.: Field-Weighted XML Retrieval Based on BM25. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 161–171. Springer, Heidelberg (2006)
8. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, pp. 157–159. Cambridge University Press, Cambridge (2008)
9. Piwowarski, B., Gallinari, P.: A Bayesian Framework for XML Information Retrieval: Searching and Learning with the INEX Collection. *Journal of Information Retrieval* 8(4), 655–681 (2005)
10. Robertson, S., Zaragoza, H., Taylor, M.: Simple BM25 Extension to Multiple Weighted Fields. In: Proceedings of the 13 ACM International Conference on Information and Knowledge Management, pp. 42–49 (November 2004)
11. Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M.: Okapi at TREC-3. In: The Third Text Retrieval Conference (TREC-3), pp. 109–126 (1995)
12. Salton, G., Buckley, C.: Term-Weighting Approaches in Automatic Text Retrieval. *Journal of Information Processing and Management* 24(5), 513–523 (1988)

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. Of the public search engines, Google and Yahoo perform marginally better than Bing and significantly better than the on-site Wikipedia search.

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 not been reported before, to the best of my knowledge. 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 the metrics of INEX. For example, novelty and diversity^[3] are not rewarded; nor is wasted user

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

effort penalized. Nevertheless, the results should be indicative of the true performance, given the variety of different metrics and appropriate interpretation of the results.

It is commonly known that web search engines rely on information outside the Wikipedia, including incoming links from other web pages as well as click data specific to each query and search engine. This might give them an advantage over the academic search engines that only rely on the INEX version of the Wikipedia. However, as the Web is available to all the INEX participants, we cannot assume that the presumed advantage is unfair.

The 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. Google and Yahoo are highly competitive when looking at the first page of results.

2 Related Work

Commercial online search engines have been compared in the past in various ways and various metrics. Measuring the popularity and profitability is without a doubt an interesting way to rank the search businesses by their success. A more technical aspect that is often measured is the size of the index or web coverage [5]. However, there is no search business without a search engine with an effective search algorithm. Previously, not many Cranfield-style evaluations of this scale on commercial search engines have ever been published. For example, Tümer et al. compared Google, Yahoo, MSN, and Hakia with 10 queries on the live WWW [6], whereas Bar-Ilan et al. compared the result lists of Google, Yahoo, and Teoma and measured overlap and statistical correlation between the lists [2].

A recent study worth noting was conducted by Ganjisaffar et al. [4] who pooled the top 10 results of Google, Yahoo, and Live search on the domain of *en.wikipedia.org*. In their experiment, seven assessors labelled a total of 240 queries resulting in a finding that Live search outperforms both Google and Yahoo.

3 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. Quotation marks, plus and minus signs were all included as such.

3.1 Google

Of all the different Google search sites, the one that is not specific to any country was chosen² 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.

3.2 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 bing³ was chosen for except 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

http://en.wikipedia.org/wiki/Marilyn_Monroe.

The contents of these two pages are equivalent, but, if the page is relevant to a query, only Google scores whereas Bing hits a missing page. None of the search engines retrieve both pages because they try to avoid including duplicate pages in the results. This might be a case where Bing is being unfairly penalized for retrieving “wrong” pages where the content is relevant but the title is misspelled.

3.3 Yahoo!

Yahoo⁴ 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 problems is the same: Yahoo too retrieves a number of redirected pages which are not included in the INEX version of Wikipedia.

3.4 Wikisearch

The on site Wikipedia search engine⁵ is the only search engine in this experiment that returns focused results — in addition to generating snippets for each

² www.google.com/ncr

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

⁴ search.yahoo.com

⁵ <http://en.wikipedia.org/w/index.php?title=Special:Search&search=...>

article. 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 version of the Wikipedia.

4 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.

4.1 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 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.

4.2 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 are 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.

4.3 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.

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

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

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 than the number of submitted results because of new pages and redirecting pages.

5 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 has the highest *character precision*, retrieving the highest ratio of relevant content to non-relevant content. However, limiting the results to 500 characters per article was merely an artificial post-search procedure to satisfy the task requirements, and therefore, the search engines should not be rewarded or penalised for it. As this comparison only considers the top two results for each query, it is fair to compare how many relevant articles the search engines ranked in the top two ranks of the result list. Google tops the chart as the only search engine that returns at least two results for each of the 52 queries included in the evaluation. Google also has the highest total number of relevant articles found (79) and the highest precision (75.96%).

Table 2. Evaluation of the top two results for the 52 topics submitted for the restricted focused task

Search engine	articles	relevant	art_prec	char_prec	iP0.01
Google	104	79	0.7596	0.3276	0.1040
Yahoo	102	77	0.7404	0.3435	0.1186
Bing	102	75	0.7212	0.3354	0.1062
Wiki	100	68	0.6538	0.2670	0.0713

Comparing the public search engines with the best academic ones, we note that Yahoo (0.3435) ranks the 3rd in the Restricted Focused task where the official measure was character precision [1]. Bing (0.3354) and Google (0.3276) are not far behind whereas Wikisearch has the lowest score — obviously due to the largest number of results per query.

Whether there is any significant difference between the results is tested in Table 3. According to the topic-wise t-test on article precision, both Google and Yahoo perform significantly better than the default Wikipedia search but the differences between other search engines are not statistically significant.

Table 3. P-values of the t-test (one-tailed)

T-test	Yahoo	Bing	Wiki
Google	0.2424	0.1260	0.0074
Yahoo		0.2989	0.0138
Bing			0.0818

The RRIC runs consist of the first page of results for each query before restricting the submission to the first 500 characters of the article. The performance of the official RRIC runs are shown in Table 4. The number of results returned on the first page varies a lot, which makes a direct comparison of absolute precision or recall unjustified. As we are interested in article precision at fairly low ranks, we order the runs according to interpolated precision at 0.01 which has been considered a rather stable measure in the past INEX evaluations.

Table 4. Evaluation of runs submitted for the restricted relevant in context task

Search engine	articles (avg)	relevant	art_prec	char_prec	iP0.01
Yahoo	2946 (56.7)	1075	0.3463	0.1290	0.2848
Google	3537 (68.0)	1305	0.3629	0.1264	0.2658
Wiki	13688 (263.2)	2344	0.1819	0.0463	0.1995
Bing	549 (10.6)	301	0.5363	0.0115	0.1975

As a side note, Bing and Yahoo results are slightly compromised because of the mismatch between titles returned and titles in the INEX collection. Moreover, all the search engines did slightly better live than in this experiment as they retrieved relevant articles which were more recent than the INEX 2009 document collection.

6 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.

What we learn from this experiment is that searching the Wikipedia with Google or Yahoo is worth a try when the Wikipedia search does not find any relevant articles. However, the only search engine that returned focused results was the Wikipedia search — a feature that will hopefully be appreciated by future users.

References

1. Arvola, P., Geva, S., Kamps, J., Schenkel, R., Trotman, A., Vainio, J.: Overview of the inex 2010 ad hoc track. In: INEX 2010. LNCS, vol. 6932, pp. 1–32. Springer, Heidelberg (2011)
2. Bar-Ilan, J., Mat-Hassan, M., Levene, M.: Methods for comparing rankings of search engine results. *Comput. Netw.* 50, 1448–1463 (2006)
3. Clarke, C.L., Kolla, M., Cormack, G.V., Vechtomova, O., Ashkan, A., Büttcher, S., MacKinnon, I.: Novelty and diversity in information retrieval evaluation. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, pp. 659–666. ACM, New York (2008)
4. Ganjisaffar, Y., Javanmardi, S., Lopes, C.: Leveraging crowdsourcing heuristics to improve search in wikipedia. In: Proceedings of the 5th International Symposium on Wikis and Open Collaboration, WikiSym 2009, pp. 27:1–27:2. ACM, New York (2009)
5. Kim, Y.S., Kang, B.H., Compton, P., Motoda, H.: Search engine retrieval of changing information. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, pp. 1195–1196. ACM, New York (2007)
6. Tumer, D., Shah, M.A., Bitirim, Y.: An empirical evaluation on semantic search performance of keyword-based and semantic search engines: Google, yahoo, msn and hakia. In: Proceedings of the 2009 Fourth International Conference on Internet Monitoring and Protection, pp. 51–55. IEEE Computer Society, Washington, DC, USA (2009)

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 than on relevant paragraphs. Our findings are: 1) the simplest language model for document retrieval performs relatively well 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; 3) stemming and stopword removal can further boost performance.

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 have been created by the INEX participants to represent real life information needs. As in 2009, each topic consists of five fields. The `<title>` field (Content Only, or CO query) is the same as the standard keyword query. The `<castitle>` field (Content And Structure, or 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.

In our last year's work [1], we investigated how a standard IR engine performs in document and XML element retrieval using the simplest language model. We

found 1) the full document retrieval outperforms the XML element retrieval using language model based on Dirichlet priors; 2) the element relevance score itself can be used to remove overlapping element results effectively.

This paper documents our primary, official, and unofficial 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 (e.g. stemming and stopping). We adopt the language modeling approach [2] and tailor the estimate of query term generation from a document to query term 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 described in section 2. Section 3 explains how to convert a user query to an Indri structured query. The retrieval model and strategies that we used are summarized in section 4. We present our results and their analysis in section 5 and conclude this paper in section 6.

2 Pre-processing and Indexing

The 2010 INEX Wikipedia collection has 2,666,190 English XML documents that were taken on 8 October 2008. It is annotated with the 2008-w40-2 version of YAGO ([3]), which is a knowledge base developed at the Max-Planck-Institute Saarbrücken.

We build up two indices. One is from the original Wikipedia version and the other from the stemmed and stopped version of this Wikipedia. We only index the following XML elements:

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 as full document retrieval while ignoring boolean operators (e.g. “-” or “+”) in their respective <title> and <castitle> fields. We extract all <castitle> terms within “about”. Both types of queries use the Indri belief operator #combine [4].

Additionally, we remove terms like “of”, “or”, “and”, “in”, “the”, “from”, “on”, “by”, and “for” from both CO and CAS queries when retrieving articles on the original Wikipedia. For the stemmed and stopped Wikipedia, we also

stem and stop CO and CAS queries using Porter’s stemmer and the stopword list of the Indri search engine.

CO or CAS queries for full document retrieval: For instance, for INEX query (id=2010003) we have:

```
<title>Monuments of India</title>
<castitle>
//article[about(., Monuments) and about(., India)]
//sec[about(., Monuments of India)]
</castitle>
```

After the described operation, the formulated Indri queries look like:

- **CO query:**
#combine[article](monuments india)
- **CAS query:**
#combine[article](monuments india monuments india)
- **Stemmed and stopped CO query:**
#combine[article](monument india)
- **Stemmed and stopped CAS query:**
#combine[article](monument india monument india)

4 Retrieval Models and Strategies

We first retrieve full articles using either CO or CAS (stemmed and stopped) 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. To improve the performance, additional runs are made on stemmed and stopped CO or CAS queries. Based on last year’s experience that aggregating good result of document retrieval can further improve performance, the provided reference run is used as the final rank of retrieved documents for all our results. For instance, one of our runs has result document ranking: 1, 2, 3, 4. The reference run has document ranking: 3, 1, 5, 4. The reference-based

result is then 3, 1, 4. Document 2 from our run is then missed while document 5 from the reference run is not considered.

The reference run uses BM25 ranking function:

$$score(D|Q) = \sum_{i=1}^l idf(q_i) \frac{tf(q_i, D) \cdot (k_1 + 1)}{tf(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \quad (2)$$

$$idf(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \quad (3)$$

with $k_1 = 1.2$ 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.

In our full document retrieval experiment, we submitted 9 results for official evaluation of three tasks. The focus of these experiments is on the retrieval model only. Consequently, for these experiments we did not consider system efficiency. In this paper, we provide more unofficial results when stemming and stopping techniques are used.

4.2 Snippet Retrieval

Within the ad-hoc retrieval track three sub-tasks have the form of snippet retrieval:

- The **Restricted Focused (ResFocus) task** is a variant of the Focused Task where only results with at most 1,000 characters per topic are allowed. This year we generate snippet retrieval from full document retrieval. We adopt a fixed offset for this task. The relevant characters are offset to 1000 characters or offset to the length of a document that contains less than 1000 characters.
- The **Relevant in Context (RiC) task** requires the system to return relevant elements or passages clustered per article. Within each article, reading order of the retrieved element matters. This year we submit our full document retrieval for this task.
- The **Restricted Relevant in Context (ResRiC) task** is similar to RiC but only allow 500 characters per article may be retrieved. Our runs are built in the similar way as for the restricted focused task except the relevant characters are only 500.

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 have two document retrieval results for CO and CAS (with and without stemming and stopping) queries and two reference-based results. On the whole, we have 9 official runs, of which 5 are qualified in the

ad-hoc track. For unofficial runs (noted as *), we remove the duplicated articles in the disqualified official results and produce 12 more runs for (restricted) relevant in context tasks.

5.1 Runs

Our official and unofficial runs are named in the following way:

- first field:** abbreviation of sub-tasks
- second field:** query type (e.g. CO or CAS)
- third field:** reference-based run if it is used (e.g. Ref)
- last field:** stemmed and stopped
- unofficial runs are noted as *

Our runs are referred as baseline runs when not using stemming and stopping or the reference run. We also convert the provided reference run to the required format of sub-tasks using the same strategy described in sub-section 4.2. The pure reference runs are named with “Ref” and abbreviation of sub-tasks (e.g. RefRiC).

5.2 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 participating groups, our best run ranks 9th of 19 groups 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 summarized in Table 4. The total evaluated runs from participating groups are 34, 65, and 39 following the top-down order of sub-tasks in this table. Our simple approach for restricted focused document retrieval is relatively effective, even when there is no help of the reference run.

5.3 Measured as Snippet Retrieval

When measured as thorough task, our results for the RiC task rank higher than our other results, followed by the 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 ranks 3rd in 17 participating groups and 14th in all 217 evaluated runs with MAiP of 0.2230 when measured as thorough retrieval. The performance overview of sub-tasks of snippet retrieval is represented separately in the following sub-sections.

Restricted Focused. The ResFocus task is to return a 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 and return the following 1000 characters or the rest of the document that has less than 1000 characters. The

Table 1. Measured as document retrieval

performance metrics	official runs for restricted focused		
	ResFocusCOPref	ResFocusCO	ResFocusCAS
MAP	0.2593	0.0243	0.0226
our rank / all official runs	6/34	7/34	8/34
	official runs for relevant in context		
	RiCCOPref	RiCCO	RiCCAS
MAP	0.2593	0.0243	0.0226
our rank / all official runs	32/65	61/65	62/65
	official runs for restricted relevant in context		
	ResRiCCOPref	ResRiCCO	ResRiCCAS
MAP	0.2593	0.0243	0.0226
our rank / all official runs	14/39	35/39	36/39

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 5th among the participating groups and ranks 8th in all 34 evaluated runs in terms of the score of character precision (char_prec). Our other two runs rank 12th and 13th 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
ResFocusCOPref	0.3361	0.0964	0.0435	0.0000	0.0067	0.3361
ResFocusCAS	0.3241	0.0820	0.0357	0.0000	0.0061	0.3241
ResFocusCO	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. All official and unofficial (noted as *) results are presented in Table 3. Our reference-based run ranks 6th among 15 participating groups and 20th in all 47 evaluated runs with the score of MAgP 0.1377 (see Table 3).

The result of CAS queries is better than that of CO queries for original and stemmed-stopped indices. The reference run boosts the performance of CO queries more compared to CAS queries. The technique of stemming and stopping brings more performance gain. The best performance reaches MAgP 0.1395 when reference run, stemming, and stopping are applied.

Restricted Relevant in Context. The ResRiC task limits the result of the RiC task to be 500 characters at most. Similar to the restricted focused task, our

Table 3. Measured as focused retrieval

runs for relevant in context	performance metrics				
	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
RiCCORef	0.2642	0.2310	0.1694	0.1431	0.1377
RiCCASRef*	0.2641	0.2318	0.1686	0.1445	0.1377
RiCCAS*	0.2360	0.2081	0.1734	0.1408	0.1232
RiCCO*	0.2333	0.1985	0.1656	0.1305	0.1212
RiCCORef_ss*	0.2618	0.2325	0.1706	0.1439	0.1395
RiCCASRef_ss*	0.2618	0.2325	0.1702	0.1439	0.1393
RiCCAS_ss*	0.2787	0.2309	0.1769	0.1326	0.1316
RiCCO_ss*	0.2836	0.2215	0.1694	0.1269	0.1304
RefRiC*	0.2684	0.2322	0.1714	0.1442	0.1436

results start from the 21-th character of the result of document retrieval. Again only the reference-based run is officially evaluated and its performance with others (noted as *) is summarized in Table 4. It ranks 9th among 9 participating groups and 12th in all 24 evaluated runs with the score of MAgP 0.1375 (see Table 4).

Similar to the findings in the RiC task, the result of CAS queries outperforms that of CO queries without reference information. The best result is MAgP 0.1392 for the reference-based, stemmed, and stopped run in case of CO queries.

Table 4. Measured as focused retrieval

runs for restricted relevant in context	performance metrics				
	gP[5]	gP[10]	gP[25]	gP[50]	MAgP
ResRiCCORef	0.2641	0.2313	0.1686	0.1428	0.1375
ResRiCCASRef*	0.2641	0.2313	0.1682	0.1435	0.1374
ResRiCCAS*	0.2346	0.2067	0.1727	0.1404	0.1229
ResRiCCO*	0.2319	0.1972	0.1650	0.1300	0.1209
ResRiCCORef_ss*	0.2618	0.2319	0.1701	0.1430	0.1392
ResRiCCASRef_ss*	0.2618	0.2319	0.1697	0.1430	0.1390
ResRiCCAS_ss*	0.2786	0.2316	0.1767	0.1326	0.1317
ResRiCCO_ss*	0.2823	0.2224	0.1693	0.1265	0.1304
RefResRiC*	0.2685	0.2325	0.1708	0.1439	0.1434

5.4 Analysis

In general, BM25 ranking function generates better results than our simplest language model, which has no stopping and stemming. When stemming and stopping techniques are applied, improvement by reference run is marginal. 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.

Table 5. Number of queries with measure score less than average

	all query type	non-article element	any type	performance measure
ResFocusCOfRef	32	13	13	char_prec
ResFocusCAS	33	12	14	char_prec
ResFocusCO	32	11	14	char_prec
RiCCOfRef	31	14	13	MAgP
ResRiCCOfRef	32	14	13	MAgP
RiCCOfRef _{ss} *	31	14	13	MAgP
RefRiC*	31	14	13	MAgP
ResRiCCOfRef _{ss} *	31	14	14	MAgP
RefResRiC*	32	14	13	MAgP

There are 52 queries (topics) used for evaluation in all snippet retrieval. 18 out of 52 is to return XML elements such as “sec”, “p”, “person”, “flower”, and “*/sec” rather than “article”. The rest is the document retrieval with restriction on a fixed number of characters. When a query is about any XML element type (noted as * or “article/*”), we may also accept the whole document. There are 15 queries on the “article” element and 19 queries on any XML element.

For possible performance improvement, it is important to identify the difficult queries that bring down the average measure score. The number of queries that have lower measure score than the average is presented in column 2 of the Table 5. The number of queries that retrieve XML elements other than “article” and any type is in column 3 while the total number of such queries is 18. Though our retrieval model also fails on queries for “article”, there is slightly more percentage loss on other types of XML elements. The column 4 is the number of queries that retrieve any XML elements. The column 5 is the performance metric used for identifying difficult queries. We only examine the official results and the best unofficial runs.

The number of queries that have zero measure score at average are 27 for all restricted focused runs, 3 for the RiC run, and 4 for the restricted RiC run.

Particularly, the most difficult queries that gain the least measure score in all tasks are topic 19, topic 31, and topic 107. Their `title` and `castitle` fields 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>
```

These queries retrieve either non-article elements or elements of any type. It is not surprised that non-article element retrieval has worse performance than article and any type element retrieval because our runs favors document retrieval rather than element retrieval. For the retrieval of any type element, more investigation is needed on the relevance judgement (qrels) such as if there is preference on a certain element and which element it is.

6 Conclusion

In this paper, we present our results for the ad-hoc track of INEX 2010. Our official 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 baseline runs perform relatively good among participating groups in the restricted focused retrieval. Our performance loss happens more on retrieval of non-article elements than on retrieval of article element. The reference run does provide better document ranking, which in turn improve the performance of our baselines. When stemming and stopping are applied, the reference-based run can be further improved. At the meantime, the baseline runs gain more performance improvement that reduces the difference between baselines and reference-based runs.

References

1. Li, R.M., van der Weide, T.P.: 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 für 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: Scaling Up the Evaluation Using Crowdsourcing

Gabriella Kazai¹, Marijn Koolen², Jaap Kamps²,
Antoine Doucet³, and Monica Landoni⁴

¹ Microsoft Research, United Kingdom
v-gabkaz@microsoft.com

² University of Amsterdam, Netherlands
{m.h.a.koolen,kamps}@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 Book Track is to evaluate approaches for supporting users in searching, navigating and reading the full texts of digitized books. The investigation is focused around four tasks: 1) Best Books to Reference, 2) Prove It, 3) Structure Extraction, and 4) Active Reading. In this paper, we report on the setup and the results of these tasks in 2010. The main outcome of the track lies in the changes to the methodology for constructing the test collection for the evaluation of the Best Books and Prove It search tasks. In an effort to scale up the evaluation, we explored the use of crowdsourcing both to create the test topics and then to gather the relevance labels for the topics over a corpus of 50k digitized books. The resulting test collection construction methodology combines editorial judgments contributed by INEX participants with crowdsourced relevance labels. We provide an analysis of the crowdsourced data and conclude that – with appropriate task design – crowdsourcing does provide a suitable framework for the evaluation of book search approaches.

1 Introduction

Prompted by the availability of large collections of digitized books, e.g., the Million Book project¹ and the Google Books Library project² the Book Track was launched in 2007 with the aim to promote research into techniques for supporting users in searching, navigating and reading the full texts of digitized books. Toward this goal, the track provides opportunities to explore research questions around three areas:

- Information retrieval (IR) methods for searching collections of digitized books,
- Mechanisms to increase accessibility to the contents of digitized books, and
- Users' interactions with eBooks and collections of digitized books.

¹ <http://www.ulib.org/>

² <http://books.google.com/>

Based around the three main themes above, the following four tasks were investigated in 2010:

1. The *Best Books to Reference* (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 focused retrieval approaches on collections of books, where users expect to be pointed directly at relevant book parts that may help to confirm or refute a factual claim;
3. The *Structure Extraction* (SE) task aims at evaluating automatic techniques for deriving structure from OCR and building hyperlinked table of contents;
4. The *Active Reading task* (ART) aims to explore suitable user interfaces to read, annotate, review, and summarize multiple books.

In this paper, we report on the setup and the results of each of these tasks at INEX 2010. However, the main focus of the paper is on the challenge of constructing a test collection for the evaluation of the BB and PI tasks. This challenge has so far remained the main bottleneck of the Book Track, which in the past three years has struggled to build a suitably large scale test collection relying on its participants' efforts alone. Indeed, the effort required to contribute to the building of such a test collection is daunting. For example, the estimated effort that would have been required of a participant of the INEX 2008 Book Track to judge a single topic was to spend 95 minutes a day for 33.3 days [11]. This level of demand is clearly unattainable. At the same time, as a requirement of participation, it poses a huge burden and is likely to be one of the causes of the low levels of active participation that follows the high number of registrations.

To address this issue, this year we explored the use of crowdsourcing methods to contribute both the topics and the relevance labels to the test collection. This follows the recent emergence of human computing or crowdsourcing [8] as a feasible alternative to editorial judgments [2,11,7,13]. Similarly to our case, such efforts are motivated by the need to scale up the Cranfield method for constructing test collections where the most significant effort and cost is associated with the collection of relevance judgments. By harnessing the collective work of the crowds, crowdsourcing offers an increasingly popular alternative for gathering large amounts of relevance data feasibly at a relatively low cost and in a relatively short time.

Our goal this year was to establish if crowdsourcing could indeed be relied upon for creating a suitable test collection for the Book Track. To this end, we combined editorial judgments contributed by 'trusted' INEX participants with crowdsourced data, using the editorial labels as a gold set to measure the quality of the crowdsourced labels. In addition, we also explored the possibility to crowdsource not only the relevance labels, but the test topics too. Our analysis shows that with the appropriate task design, crowdsourcing does indeed offer a solution to the scalability challenge of test collection building [10].

Table 1. Active participants of the INEX 2010 Book Track, contributing topics, runs, and/or relevance assessments (BB = Best Books, PI = Prove It, SE = Structure Extraction, ART = Active Reading Task)

ID Institute	Created topics	Runs	Judged topics
6 University of Amsterdam	19-20, 22	2 BB, 4 PI	05, 10, 18-19, 42, 64, 82
7 Oslo University College	02-06	5 PI	02-03
14 Uni. of California, Berkeley	-	4 BB	-
41 University of Caen	-	SE	SE
54 Microsoft Research Cambridge	00-01, 07-09, 24-25	-	00-01, 05-09, 12, 15, 18, 23-25, 31, 33, 42-43, 63-63, 70, 78, 81-82
86 University of Lugano	15-18, 21, 23	-	-
98 University of Avignon	-	9 BB, 1 PI	00, 24, 77
339 University of Firenze	-	-	SE
386 University of Tokyo	-	SE	-
663 IIT-H	10-14	-	-
732 Wuhan University	-	SE	-

In the following, we first we give a brief summary of the actively participating organizations (Section 2). In Section 3, we describe the book corpus that forms the basis of the test collection. The following three sections discuss our test collection building efforts using crowdsourcing: Section 4 details the two search tasks: BB and PI; Section 5 details our topic creation efforts; and Section 6 details the gathering of relevance labels. Then, in Section 7 we present the results of the BB and PI tasks, while Sections 8 and 9 summarize the SE and ART tasks. We close in Section 10 with a summary and plans for INEX 2011.

2 Participating Organizations

A total of 93 organizations registered for the track (compared with 84 in 2009, 54 in 2008, and 27 in 2007). However, of those registered only 11 groups took an active role (compared with 16 in 2009, 15 in 2008, and 9 in 2007), see Table 1.

2.1 Summary of Participants' Approaches

The University of Avignon (ID=98, [3]) contributed runs to the BB and PI tasks. They experimented with a method for correcting hyphenations in the books and applying different query expansion techniques. For retrieval they used the language modeling approach of the Lemur toolkit. In total, they corrected over 37 million lines (about 6%) in the corpus that contained hyphenated words, leading to around 1% improvement in MAP. No improvements were observed as a result of query expansion.

Oslo University College (ID=7, [14]) took part in the PI task and explored semantics-aware retrieval techniques where the weights of verbs that reflect

confirmation were increased in the index. Using language modeling as their retrieval approach, they show that the new index can improve precision at the top ranks.

The University of California, Berkeley (ID=14, [12]) experimented with page level retrieval in the BB task. They derived book level scores by summing the page level scores within the books. Page level scores were generated in two ways: using a probabilistic approach based on logistic regression, and using coordination-level match (CML). They found that simple methods, e.g., CML do not work for book retrieval. Their page level logistic regression based method yielded the best results overall.

The University of Amsterdam (ID=6, [9]) looked at the effects of pseudo relevance feedback (RF) in both the BB and PI tasks, and also investigated the impact of varying the units of retrieval, e.g., books, individual pages, and multiple pages as units in the PI task. In the BB task, they found that their book level retrieval method benefited from RF. In the PI task, they achieved best performance with individual page level index and using RF. With larger units, RF was found to hurt performance.

The University of Caen (ID=41, [6]) participated in the SE task, continuing their approach of last year that uses a top-down document representation with two levels, part and chapter, to build a model describing relationships for elements in the document structure. They found that their approach is simple, fast, and generic—using no lexicon or special language dependent heuristics—but is also outperformed by methods tuned to the corpus and task at hand.

3 The Book Corpus

The Book Track builds on a collection of 50,239 out-of-copyright books³, digitized by Microsoft. The corpus contains 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 bookID, which is also the name of the directory that contains the book's OCR file, e.g., A1CD363253B0F403.

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 a set of labels (as attributes) and additional marker elements for more complex structures, like table of contents. For example, the first label attribute in the XML extract below 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

³ Also available from the Internet Archive (although in a different XML format).

(SEC_TOC). Marker elements provide detailed markup, e.g., indicating entry titles (TOC_TITLE) or page numbers (TOC_CH_PN) in a table of contents.

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>[...]
```

The full corpus, totaling around 400GB, is available on USB HDDs. A reduced version (50GB, or 13GB compressed) is available via 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 Search Tasks

Focusing on IR challenges, two search tasks were investigated in 2010: 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 5).

4.1 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 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 often found in academic publications or Wikipedia articles. The reading list may be for educational purposes or for entertainment, etc.

The task was defined as: “*The task is to return, for each test topic, a ranked list of 100 (one hundred) books estimated relevant to the general subject area of the factual statement expressed within the test topics, ranked in order of estimated relevance.*”

Participants were invited to submit either single or pairs of runs. Each run had to include the search results for all the 83 topics of the 2010 test collection (see Section 5). A single run could be the result of either a generic (non-book-specific) or a book-specific IR approach. A pair of runs had to contain a non-book-specific

run as a baseline and a book-specific run that extended upon the baseline by exploiting book-specific features (e.g., back-of-book index, citation statistics, book reviews, etc.) or specifically tuned methods. A run could contain, for each topic, a maximum of only 100 books, 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 Prove It (PI) Task

The goal of this task is 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 refute a given factual statement (topic). Users are assumed to view the ranked list of book parts, moving from the top of the list down, examining each result.

In the guidelines distributed to participants, the task was defined as: *“The task is to return a ranked list of 1000 (one thousand) book pages (given by their XPath), containing relevant information that can be used to either confirm or reject the factual statement expressed in the topic, ranked in order of estimated relevance.”*

Participants could submit up to 12 runs, each containing a maximum of 1,000 book pages per topic for each of the 83 topics (see Section [5](#)), ranked in order of estimated 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](#).

5 Test Topics for the Search Tasks

In an effort to focus the search intentions to more specific (narrow) topics, this year we defined the test topics around one-sentence factual statements. Unlike previous years, we also solicited test topics both from INEX participants and from workers on Amazon’s Mechanical Turk (AMT) service,^{[4](#)} a popular crowdsourcing platform and labour market. Our aim was to compare the two sources and to assess the feasibility of crowdsourcing the topics (and the relevance judgments later on) of the test collection.

5.1 INEX Topics

We asked the INEX Book Track participants to create 5 topics each, 2 of which had to contain factual statements that appear both in the book corpus and in Wikipedia. Participants were asked to fill in a web form for each of their topics, specifying the factual statement they found in a book, the query they would

⁴ <https://www.mturk.com/mturk/>

use to search for this information, the URL of the book containing the fact, the exact page number, the URL of the related Wikipedia article, the version of the fact as it appears in the Wikipedia page, and a narrative detailing the task and what information is regarded by the topic author as relevant. A total of 25 topics were submitted by 5 groups. Of these, 16 facts appear both in books and in Wikipedia.

5.2 Crowdsourced Topics

To crowdsource topics, we created two different human intelligence tasks (HITs) on AMT which asked workers to find general knowledge facts either in the books of the INEX corpus or both in the books and in Wikipedia:

Fact Finding Both in Books and in Wikipedia (Wiki HIT): To gather factual statements that appear both in the book corpus and in Wikipedia, we created a HIT with the following instructions: “Your task is to find a general knowledge fact that appears BOTH in a Wikipedia article AND in a book that is available at <http://www.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. Once you found a fact both in Wikipedia and in the book collection, fill in the form below. HITs with correct matching facts will be paid \$0.25. Only facts that appear both in Wikipedia and in the booksearch.org’s book collection will be paid.” We provided an example fact and instructed workers to record the following data for the factual statement they found: the Wikipedia article’s URL, the fact as it appeared in the Wikipedia article, the URL of the book that states the same fact and the exact page number.

We set payment at \$0.25 per HIT and published 10 assignments. All 10 assignments were completed within 4 hours and 18 minutes. On average, workers spent 11 minutes on the task, resulting in an effective hourly rate of \$1.31.

Fact Finding in Books (Book HIT): To gather factual statements that appear in the book corpus, we created a simple HIT with the following instructions: “Your task is to find a general knowledge fact that you believe is true in a book available at <http://www.booksearch.org.uk>. Both the fact and the book must be in English. The fact should not be longer than a sentence. Only facts that appear in the book collection at <http://www.booksearch.org.uk> will be paid.” As with the Wiki HIT, we provided an example fact and instructed workers to fill in a form, recording the factual statement they found, the URL of the book containing the fact and the exact page number.

Given the response we got for the Wiki HIT and the simpler task of the Book HIT, we first set payment at \$0.10 per HIT and published 50 assignments. However, only 32 of the 50 assignments were completed in 13 days. We then cancelled the batch and published a second set of 50 assignments at \$0.20 per HIT, this time pre-selecting to workers by requiring at least 95% HIT approval rate. This time, all 50 assignments were completed in 14 days. The average time workers spent on the task was 8 minutes in the first batch and 7 minutes in the second batch (hourly rate of \$0.73 and \$1.63, respectively).

5.3 Topic Selection

All collected topics were carefully reviewed and those judged suitable were selected into the final test collection. All topics contributed by INEX participants were acceptable, while filtering was necessary for topics created by AMT workers. Out of the 10 Wiki HITs, only 4 topics were selected (40%). Of the 32 Book HITs in the first batch, 18 were acceptable (56%), while 36 were selected from the 50 Book HITs in the second batch (72%). Topics from AMT workers were rejected for a number of reasons:

- 19 topics were rejected as the information given was a (random) extract from a book, rather than a fact, e.g., “logical history of principle of natural selection”, “This is a picture diagram of fall pipes with imperfect joints being carried through the basement of house into drain”, “At the age of twenty five he married a widow forty years old; and for five-and-twenty years he was a faithful husband to her alone”, “A comparison of all the known specimens shows the material to be of slate and exhibits general uniformity in shape, the most noticeable differences being in the handle and the connecting neck.”;
- 5 topics were nonsensical, e.g., “As a result of this experience I became tremendously interested in the theater can be found on page 63 of the book titled Printing and book designing by Wilson, Adrian.”, “dance”, “I want a woman with a soul”, “the objective facts, in nature or in the life of man”;
- 2 topics had missing data, i.e., book URL, fact or page number;
- 2 topics referred to facts outside the book corpus, e.g., CBS news;
- 5 topics had incorrect book URLs or page references;
- 1 topic was the example fact included in the HIT.

Comparing the average time workers took to complete their task in the acceptable and not-acceptable sets of topics, we only found a small difference of 522 seconds vs. 427 seconds, respectively (st.dev. 676 and 503 seconds, min. 13 and 22 seconds, and max. 3389 and 2437 seconds), thus proving of little use in our case to automate filtering in the future.

In addition, out of the total 58 selected AMT topics, 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 however high quality (even more diverse and creative than the topics created by the INEX participants) and seemingly reflecting real interest or information need.

From the above, it is clear that crowdsourcing provides an attractive and promising means to scale up test topic creation: AMT 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. This may be improved upon by pre-selection workers through qualification tasks [21] or by adopting more defensive task design [15]. Indeed, we found that selecting workers based on their approval rate had a positive effect on quality: batch 2 of the Book HITs which required workers to have a HIT approval rate of 95% had the highest rate of acceptable topics (72%). In addition, paying workers more (per hour) also shows correlation with the resulting quality.

6 Relevance Assessments

Like the topics, the relevance assessments were also collected from two sources: INEX participants and workers on AMT.

6.1 Gathering Relevance Labels from INEX Participants

INEX participants used the assessment system module of the Book Search System⁵ developed at Microsoft Research Cambridge. This 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. Screenshots of the relevance assessment module are shown in Figures 1 and 2.

The assessment pools were created by taking the top 100 books from the BB and PI runs and ordering them by minimum rank and by popularity, i.e., the number of runs in which a book was retrieved. The book ranking of PI runs was based on the top ranked page of each book.

Relevance labels were contributed to a total of 30 topics. Of these, 21 topics were selected, those with the most judged pages at the start of January 2011, which were then consequently used for the AMT experiments (see next section). Relevance data gathering from INEX participant was frozen on the 22nd of February 2011.

6.2 Crowdsourcing Relevance Labels

We collected further relevance labels from workers on AMT for the selected 21 topics. We created 21 separate HITs, one for each topic, so that the title of the HITs could reflect the subject area of the topic in the hope of attracting workers with interest in the subject.

Each HIT consisted of 10 pages to judge, where at least one page was already labeled as *confirm* or *refute* by an INEX participant. This was done to ensure that a worker encountered at least one relevant page and that we had at least one label per HIT to check the quality of the worker's work. In each batch of HITs, we published 10 HITs per topic and thus collected labels for 100 pages per topic from 3 workers, obtaining a total of 6,300 labels (3 labels per page).

Pooling Strategy. When constructing the AMT assessment pools (100 pages per topic), we combined three different pooling strategies with the aim to get i) a good coverage of the top results of the official PI runs, ii) a large overlap with the pages judged by INEX assessors (so that labels can be compared), and iii) to maximise the number of possibly relevant pages in the pool:

- Top- n pool: we pool the top n pages of the official PI runs using a round-robin strategy.

⁵ <http://www.booksearch.org.uk/>

Books to Explain Back to Topic List User manual and rules Home About Logout

Your task is to find book pages that confirm or refute the factual statement below, or pages that are relevant to the general subject of the statement:
According to a Philippine legend the smoke coming out Carlaon volcano is caused by an old man called Harisaboqued smoking tobacco.

BOOK SEARCH

Books 1 - 10 of 83 Page 1 ➔

Book title, author, etc.	Table of contents	A page snippet
<p>Philippine folklore stories. Author: Miller, John Maurice, Published:1904, by Ginn, Pages:128 p. anting-anting bawit bungtao captan dilagan dumapuet guidala harisaboqued lerna lalabutan toku nagayarin mangta manulito onlog perico polibulac quicoy sinaga tullane</p>		
<p>Seneca fiction, legends, and myths ... Author: Curtin, Jeremiah, Published:1918 by N/A Pages:cp. 37-819. agwaq dagwanonyent dahanahyovans diq dionongas gaayvendat ganyaligova gonongwa gowa gwa hapodyovogowa haksa hinon hodadonon -khu otkatdon onowa onwa onanda</p> <p>Seneca Indians Religion and mythology. Indiagwadan</p>	<p>PART 1. MATERIAL COLLE...</p> <ol style="list-style-type: none"> 1. The sister and her ... 2. The child and his u... 3. Dogepon and his unc... 4. The woman who marri... 5. The ghost woman and... 6. Hartrousa and his fo... 7. The old man's grand... 	
<p>Seneca Indian myths. Author: Curtin, Jeremiah, Published:1922, by E.P. Dutton & company Pages:xxi, 516p. dawpawne flying-squirrel caha gowa hagsawne halendona hodadilo luhaworn</p> <p>Indians of North America Folklore. mam-water -marked nyagwaha otkatdon poyehao stageel stwa smek-wkole thousand-lags wolf-marked yellowbird yant</p>	<p>TWO YOUNG MEN WHO WENT... TREE WORK AND HIS MOT... COLD AND FROST, OR STO... THE GAYDO GOWA WOLF AN... THE HUNTER WHO BECAME... TWO SENECA WOMEN ESCAP... A DEAD MAN SPEAKS THRO...</p>	
<p>The encyclopaedia britannica; a dictionary of arts, sciences... Author: N/A Published:1878-89 by C. Scribner's sons, Pages:25 v. branchal castil condilaran deflexion electrovive Encyclopedias and dictionaries eyr-peace kirghis leuckart object-glass phrasies phratry rabbers semi-durnal strike-out suleyman tarim taranhan tide-generating tomak tong-king</p>	<p>CHAP. I. ON THE NATURE... 1. Definition of tide... 2. General description... 3. General explanation... 4. Historical sketch, &c... CHAP. II. TIDE-GENERAT... 5. Investigation of S... 6. Form of equilibrium...</p>	
<p>Volcanoes and earthquakes / Author: Thomas F. Fisher, Published:1908 by E. P. Dutton, Pages:xxi, 325 p.</p>	<p>I VERULUS. First Erupt... II ETNA. Ancient Erupt...</p>	

Fig. 1. Relevance assessment module of the Book Search System, showing the list of books in the assessment pool for a selected topic

Book Viewer Back to Book List User manual and rules Home About Logout


Find evidence to confirm/refute the fact that: **According to a Philippine legend the smoke coming out Carlaon volcano is caused by an old man called Harisaboqued smoking tobacco.**

Philippine folklore stories.
 Miller, John Maurice.
 Harisaboqued Search

Book	TOC	To judge	Search	MyNotes
Page 13	One day Harisaboqued called the people together and told them that he was going away for a long time. He asked them again not to plant over the line, and told them that if they disregarded this wish he would carry all the tobacco away and per mit no more			
Page 17	The people fled and did not stop until they were far away. Harisaboqued had kept his word.			
Page 16	At last one man planted in the for bidden ground, and, as nothing happened, others did the same, until soon the moun tain was entirely covered with the waving plants. The people were verry happy and soon forget about Harisaboqued and their promise to him.			
Page 13	HARISABOQUED AND HIS LITTLE MEN			
Page 18	rolls out of the mountain top. Villages have sprung up along the sides, but no tobacco is grown on the mountain. The people remember the tales of the former great crops and turn longing eyes to the heights above them, but they will have to wait. Harisaboq			
Page 9				

16 PHILIPPINE FOLKLORE STORIES

rolls out of the mountain top. Villages have sprung up along the sides, but no tobacco is grown on the mountain. The people remember the tales of the former great crops and turn longing eyes to the heights above them, but they will have to wait. **Harisaboqued** is still smoking his tobacco.



18 Page 18 of 134

This page contains information that:

- 3. CONFIRMS the factual statement (at the top)
- 2. REFUTES the factual statement
- 1. RELEVANT to the subject of: Harisaboqued's legend
- 0. IRRELEVANT
- 1. Don't know

Add Tags:
Notes:

Fig. 2. Relevance assessment module of the Book Search System, showing the Book Viewer window. Relevance options are listed below the book page image.

Table 2. Statistics on the INEX and AMT relevance labels for the selected 21 topics

Source	INEX	AMT	INEX+AMT
Unknown	2	805	807
Irrelevant (0)	4,792	3,913	8,705
Relevant (1)	814	148	962
Refute (2)	18	113	131
Confirm (3)	349	1,321	1,670
Total	5,975	6,300	12,275

- Rank-boosted pool: in this pool the pages from the PI runs are reranked using a favorable book ranking. This book ranking is based on both the official BB and PI runs, and was used to create the pools for the INEX assessors to judge. The resulting page ranking has potentially more relevant pages in the top ranks and has a large coverage of the pages judged by the INEX assessors.
- Answer-boosted pool: we use a heuristic similarity function to increase the number of potentially relevant pages in the pool. We take all keywords (removing stopwords) from the factual statement of the topic that does not appear in the query and subject part of the topic, and rank the pages submitted to the PI task using coordination level matching.

As a result of the mixed pooling methods, in each 100 page assessment pool we have roughly the top 30 pages per pooling method plus the known relevant pages. Pages can occur only once in each HIT, but the known relevant pages could occur in multiple HITs, leading to 1,918 query/page pairs.

6.3 Collected Relevance Data

Statistics of the collected relevance labels are presented in Table 2. The Unknown category is used for when assessors could not judge a page (because the page was not properly displayed, or the text was written a language the assessor could not read). This happened more often in the crowdsourcing phase than in the INEX assessment phase.

For the 21 topics, a total of 5,975 page-level relevance labels were collected from INEX participant and 6,300 labels from workers on AMT. However, the AMT set contains 3 judgments per page, while the INEX data contains only one label per page (mostly). Due to missing participant IDs in the user accounts of two INEX assessors, 430 pages ended up being judged by multiple assessors. As a rule, only one label was required from the set of INEX participants, so when a page was judged by an INEX participant, it was removed from the pool. On the other hand, three labels were required by non-INEX users of the Book Search System. Interestingly, out of the 430 pages with multiple judgments, there are only 39 pages with disagreements (agreement is 91%).

A noticeable difference between the INEX and AMT labels is the relative high volume of *relevant* labels at INEX and *confirm* labels in the AMT set. The latter

Table 3. Agreement and consensus among the AMT workers and agreement between AMT majority vote and INEX labels, over different classes of labels

	All Binary proof		
AMT agreement	0.71	0.78	0.89
AMT consensus	0.90	0.92	0.91
AMT-INEX agreement	0.72	0.77	0.78
AMT-INEX consensus	0.87	0.89	0.91

is at least partly due to the fact that each HIT had at least one page labeled as *confirm* or *refute* (but mostly *confirm*). In the next section, we look at the agreement between INEX and AMT labels.

6.4 Analysis of Crowdsourced Relevance Labels

In this section, we look at agreement and consensus among the AMT workers. For agreement we look at average pairwise agreement per topic and page (so over pairs of judgments). We have, in principle, judgments from three AMT workers, resulting in three pairs of different workers, whose average pairwise agreement may range from 0 (all three pick a different label) to 1 (all three pick the same label). Consensus is the percentage of labels that form the majority vote. That is, the page label that gets the majority vote of m workers out of the total of n workers labelling that page leads to a consensus of $\frac{m}{n}$. A higher consensus means agreement is more concentrated among a single label. This is useful when there are more than 2 possible labels. Again we have, in principle, judgments from three AMT workers, whose consensus may range from 0.3333 (all three pick a different label) to 1 (all three pick the same label). We also look at agreement between AMT majority vote labels and INEX labels. If this agreement is high, AMT labels might reliably be used to complement or replace editorial judgments from INEX participants.

We look at agreement and consensus among the AMT labels using a number of label classes:

- All classes: no conflation of labels, giving four classes: *irrelevant*, *relevant*, *refute* and *confirm*
- Binary: the *relevant*, *refute* and *confirm* labels are conflated, leading to only two classes: *irrelevant* and *relevant/confirm/refute*.
- Proof: we ignore the irrelevant labels and conflate the refute and confirm labels, leading to two classes: *relevant* and *confirm/refute*.

In Table 3 we see the agreement and consensus among the AMT labels. If we differentiate between all 4 labels, agreement is 0.71. Consensus is 0.90, which means that, on average, the majority vote for a label forms 90% of all worker votes. If we consider only binary labels, the percentage agreement is higher. Also the agreement among the different degrees of relevance is high with 0.78. Due to the relatively strong percentage agreement, consensus is high among all sets.

Table 4. Statistics on the official Prove It relevance assessments based on the INEX and AMT labels

Sets	INEX	AMT	ip2c-set
Judgements	5,537	1,873	6,527
Irrelevant (0)	4,502	1,500	5,319
Relevant (1)	712	17	719
Confirm/Refute (2)	323	356	489

We also look at agreement between the relevance judgments derived from the majority vote of the AMT labels with gold set of INEX labels (bottom half of Table 3). Agreement over all 4 label classes is 0.72. AMT workers are more likely to label a page as *refute* or *confirm* than INEX participants. Without the *irrelevant* labels, the *relevant* labels dominate the INEX judgments and the *refute/confirm* labels dominate the AMT judgments, which leads to a somewhat lower agreement on these labels.

6.5 Official Qrels

Page Level Judgments. From the multiple labels per page, we derived a single judgment for evaluation. First, we discarded judgments in the *unknown* category and conflate the *refute* and *confirm* labels to a single relevance value (=2). We give confirm and refute pages the same relevance value because the PI task requires a system to find pages that either confirm or refute the factual statement of the topic. Thus, both types of pages satisfy this task. We then use majority rule among the AMT labels and keep the lower relevance value in case of ties. For the 39 pages with disagreeing INEX labels, we chose the label with the higher relevance value. We merge the two sets by always keeping the INEX labels over and above an AMT label for the same page. We refer to the resulting set as the *ip2c-set* qrel set (INEX page level judgments and crowdsourced label set) and use this set as the official set for the evaluation of the PI task. Statistics for this set are given in Table 4. In total, we have 489 pages that confirm or refute a factual statement (23 per topic on average) and 719 pages that are relevant to the topic of the factual statement (34 per topic).

Book Level Judgments. In addition to the page level judgments, it was necessary to gather book level judgments to evaluate the BB runs. These labels were provided by the task organizers for the pool of books constructed from the top 10 books of all BB runs. Books were judged on a four-point scale: 0) irrelevant, 1) marginally relevant (i.e., the book contains only a handful of pages related to the subject of the topic, 2) relevant (i.e., the topic is a minor theme), and 3) perfect (the book is dedicated to the topic).

Statistics on the relevance judgements are given in Table 5. A total of 990 books have been judged for 21 topics (47 per topic). Of these, 210 were marginally relevant, 117 relevant and 36 were perfect. The 36 perfect books are spread across 11 topics. That is, for 10 topics no perfect books were pooled. There is 1 topic (2010070) with no relevant or perfect books.

Table 5. Statistics on the official Best Books relevance assessments

Judgements	990
Irrelevant (0)	627
Marginally relevant (1)	210
Relevant (2)	117
Perfect (3)	36

7 Evaluation Measures and Results for the Search Tasks

7.1 Best Books Task Evaluation

For the evaluation of the Best Books task, we use the book level relevance labels given in the `ib-org-set` qrel set and report standard trec-eval measures: Mean Average Precision (MAP), Precision at 10 (P@10) and Normalized Cumulative Gain at 10 (NDCG@10). NDCG@10 uses the graded relevance scores, while for the binary measures the four-point relevance scale was collapsed to binary labels (Irrelevant (0), all other relevant degrees (1)).

Table 6 shows the effectiveness scores for the Best Book runs, where NDCG@10 is regarded as the official measure.

The best BB run (NDCG@10=0.6579) was submitted by the University of California, Berkeley (p14-BOOKS2010_T2_PAGE_SUM_300) who employed page level retrieval methods and derived book level scores by summing the page level scores within the books. Page level scores were generated using a probabilistic approach based on logistic regression. A run by the University of Avignon followed close second with NDCG@10=0.6500. They experimented with a method for correcting hyphenations in the books and used the language modeling approach of the Lemur toolkit.

Table 6. Evaluation results for the INEX 2010 Best Books task

Run ID	MAP	P@10	NDCG@10
p14-BOOKS2010_CLM_PAGE_SUM	0.1507	0.2714	0.2017
p14-BOOKS2010_CLM_PAGE_SUM_300	0.1640	0.2810	0.2156
p14-BOOKS2010_T2FB_BASE_BST	0.3981	0.5048	0.5456
p14-BOOKS2010_T2_PAGE_SUM_300	0.5050	0.6667	0.6579
p6-inex10.book.fb.10.50	0.3087	0.4286	0.3869
p6-inex10.book	0.3286	0.4429	0.4151
p98-baseline_1	0.4374	0.5810	0.5764
p98-baseline_1_wikifact	0.4565	0.5905	0.5960
p98-baseline_2	0.4806	0.6143	0.6302
p98-baseline_2_wikifact	0.5044	0.6381	0.6500
p98-fact_query_10wikibests	0.4328	0.5714	0.5638
p98-fact_query_entropy	0.4250	0.5476	0.5442
p98-fact_query_tfidfwiki	0.3442	0.4667	0.4677
p98-fact_query_tfwiki	0.4706	0.5571	0.5919
p98-fact_stanford_deps	0.4573	0.5857	0.5976

7.2 Prove It Task Evaluation

For the evaluation of the PI task, we use the qrel set of `ip2c-set`, which contains page level judgements contributed both by INEX participants and by the workers on AMT. As detailed in Section 6, the set was created by first applying majority rule to the AMT labels after spam labels have been removed, where in case of ties we kept the lower relevance degree, then merging this set with the INEX labels always taking the INEX label above an AMT label.

As with the BB task, we report standard trec-eval measures: MAP, P@10 and NDCG@10. For NDCG, we used two different weighting options:

- 0-1-2 weighting, which simply reflects the original relevance grades, where pages that confirm/refute the topic statement are twice as important as pages that simply contain related information.
- 01-10 weighting that emphasizes pages that confirm/refute the topic statement, treating them 10 times as important as other relevant pages:
 - Irrelevant (0) \rightarrow 0
 - Relevant (1) \rightarrow 1
 - Confirm/Refute (2) \rightarrow 10

For the binary measures, all classes of relevance were mapped to 1, while irrelevant to 0. We regard the NDCG@10 as the official measure. Table 7 shows the effectiveness scores for the Prove It runs, where only exact page matches are counted as hits.

The best PI run (NDCG@10=0.2946) was submitted by the University of Amsterdam (p6-inex10.page.fb.10.50), who investigated the impact of varying the units of retrieval, e.g., books, individual pages, and multiple pages as units in the PI task. They achieved best performance with their individual page level index and using pseudo relevance feedback.

Accounting for Near-Misses in the PI Task. Figure 3 shows the effectiveness scores for the Prove It runs, calculated over the `ip2c-set`, where near-misses

Table 7. Evaluation results for the INEX 2010 Prove It task (exact match only)

Run ID	MAP	P@10	NDCG@10 (0-1-2 weighting)	NDCG@10 (0-1-10 weighting)
p6-inex10.5page.fb.10.50	0.1163	0.2143	0.1703	0.1371
p6-inex10.5page	0.1209	0.2619	0.2182	0.1714
p6-inex10.page.fb.10.50	0.1521	0.3524	0.2946	0.2322
p6-inex10.page	0.1216	0.3238	0.2795	0.2338
p7-to.g.res	0.0453	0.1714	0.1276	0.0876
p7-to.g_2xover1.res	0.0342	0.1476	0.1225	0.0882
p7-to.g_2xover3.res	0.0288	0.1286	0.1124	0.0827
p7-to.g_5xover1.res	0.0340	0.1476	0.1195	0.0841
p7-to.g_5xover3.res	0.0262	0.1333	0.1119	0.0826
p98-fact_query_10wikibests_focus	0.0097q	0.0429	0.0321	0.0222

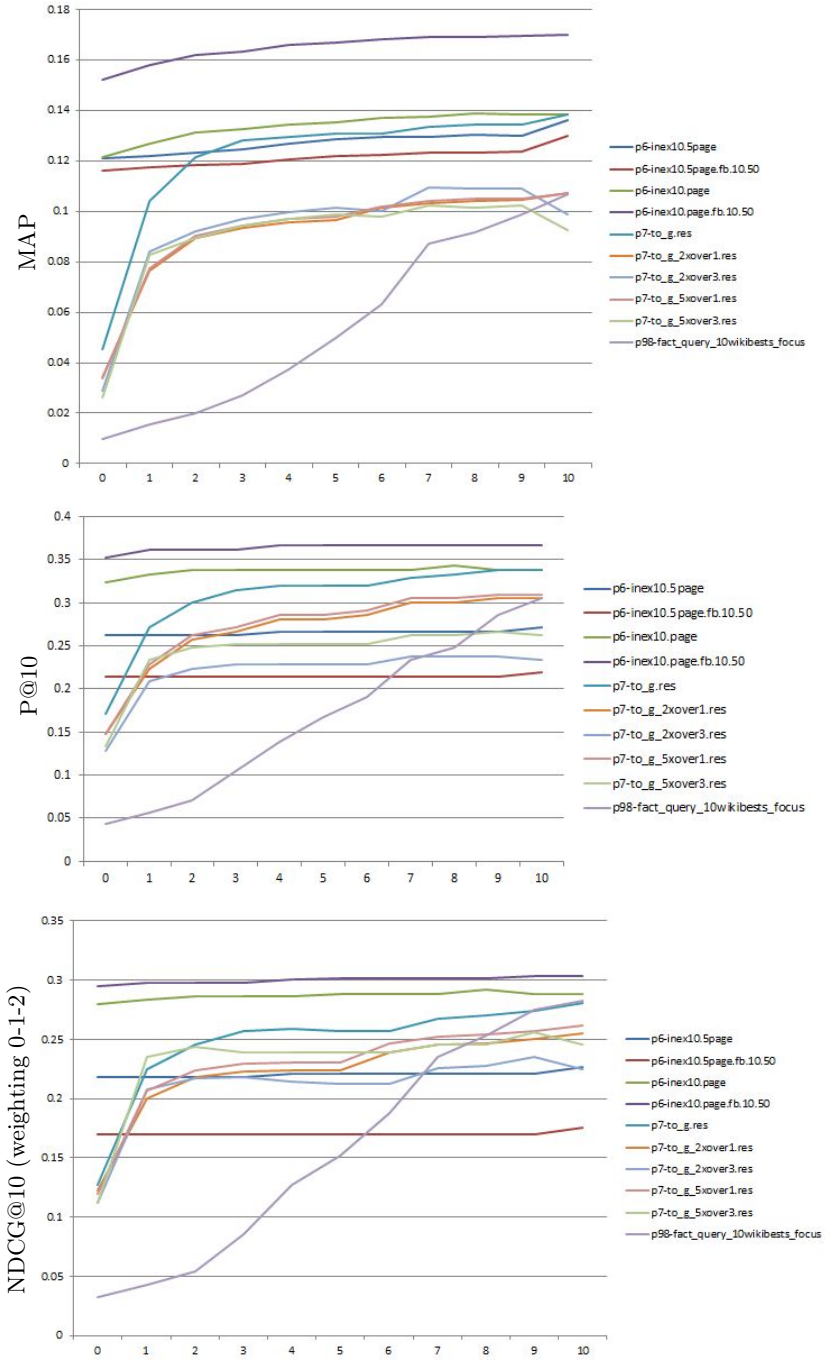


Fig. 3. Evaluation results for the INEX 2010 Prove It task with near-misses of n page distance

are taken into account. This was done by ‘replacing’ an irrelevant page in a run with a relevant page that is within n distance, starting with $n=0$ and increasing to 10 (where a relevant page could only be ‘claimed’ once). Some of the lower scoring submission pick up quickly, showing that they do retrieve pages in books with relevance, even retrieve pages that are in close proximity to the desired relevant page. The better scoring runs are fairly stable, demonstrating clearly that they are effective in locating the precise relevant pages inside the books.

8 The Structure Extraction (SE) Task

The goal of the SE task is to test and compare automatic techniques for extracting structure information from digitized books and building a hyperlinked table of contents (ToC). In 2010, the task was run only as a follow-up of the conjoint INEX and ICDAR 2009 competition [4,5], enabling participants to refine their approaches with the help of the ground-truth built in 2009.

Only one institution, the University of Caen, participated in this rerun of the 2009 task. Both the University of Caen and a new group, the University of Firenze, contributed to the building of the SE ground-truth data, adding 114 new books with annotated ToCs, increasing the total to 641 books.

The performance of the 2010 run is given in Table 8. A summary of the performance of the 2009 runs with the extended 2010 ground-truth data is given in Table 9.

Table 8. Score sheet of the run submitted by the University of Caen during the 2010 rerun of the SE competition 2009

	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 9. Summary of performance scores for the 2009 runs with the extended 2010 ground-truth data; results are for complete entries

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%

9 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.

ART is based on the evaluation experience of EBONI [16], 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.

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

10 Conclusions and Plans

The INEX Book Track promotes the evaluation of modern access methods that support users in searching, navigating and reading the full texts of digitized books, and investigated four tasks: 1) Best Books to Reference, 2) Prove It, 3) Structure Extraction, and 4) Active Reading. In this paper, we reported on the setup and the results of these tasks in 2010.

The main track activity was in the two search tasks, Best Books and Prove It. A total of 15 BB runs were submitted by 3 groups, and a total of 10 PI runs by 3 groups. Best Book submissions were shown to be highly effective, the best BB run obtaining an NDCG@10 score of 0.6579 (University of California, Berkeley, who combine book level and page level scores), and the runner up run a score of 0.6500 (University of Avignon, who used a dedicated tokenizer within the language modeling approach). The Prove It submissions were surprisingly effective, given that they try to solve the genuine needle-in-a-haystack problem of book page retrieval. This was probably aided by the topics being verbose and specific statements of facts to be confirmed or refuted. The best PI run obtained an NDCG@10 score of 0.2946 (University of Amsterdam, using an individual page level index with pseudo relevance feedback). The SE task was run (though not advertised), using the same data set as last year. One institution participated and contributed additional annotations. The final task, ART, attracted the interest of two participants, but no comprehensive experiment was conducted.

The main outcome of the track this year lies in the changes to the methodology for constructing the test collection for the evaluation of the two search tasks. In

an effort to scale up the evaluation, we explored the use of crowdsourcing both to create the test topics and then to gather the relevance labels for the topics over a corpus of 50k digitized books. The resulting test collection construction methodology combines editorial judgments contributed by INEX participants with crowdsourced relevance labels. With our quality control rich crowdsourcing design, we obtained high quality labels showing 78% agreement with INEX gold set data [10]. This has paved the way to completely removing the burden of relevance assessments from the participants in 2011.

In 2011, the track will shift focus onto more social and semantic search scenarios, while also continuing with the ART and SE tasks. The track will build on its current book corpus as well as a new collection from Amazon Books and LibraryThing. The PI task will run with minor changes, also asking systems to differentiate positive and negative evidence for a given factual claim. The BB task will be replaced by the new Social Search for Best Books (SSBB) task which will build on the corpus of 1.5 million records from Amazon Books and LibraryThing. SSBB will investigate the value of user-generated metadata, such as reviews and tags, in addition to publisher-supplied and library catalogue metadata, to aid retrieval systems in finding the best, most relevant books for a set of topics of interest.

Acknowledgments. The crowdsourcing experiments of the track were generously supported by Microsoft Research Cambridge. Marijn Koolen and Jaap Kamps were supported by the Netherlands Organization for Scientific Research (NWO, grants 612.066.513, 639.072.601, and 640.001.501).

References

1. Alonso, O., Mizzaro, S.: Can we get rid of TREC assessors? using Mechanical Turk for relevance assessment. In: Geva, S., Kamps, J., Peters, C., Sakai, T., Trotman, A., Voorhees, E. (eds.) *Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation*, pp. 15–16 (2009)
2. Alonso, O., Rose, D.E., Stewart, B.: Crowdsourcing for relevance evaluation. *SIGIR Forum* 42, 9–15 (2008)
3. Deveaud, R., Boudin, F., Bellot, P.: LIA at INEX 2010 Book Track. In: Geva, S., et al. (eds.) *INEX 2010. LNCS*, vol. 6932, pp. 118–127. Springer, Heidelberg (2010)
4. Doucet, A., Kazai, G., Dresevic, B., Uzelac, A., Radakovic, B., Todic, N.: IC-DAR 2009 Book Structure Extraction Competition. In: *Proceedings of the Tenth International Conference on Document Analysis and Recognition (ICDAR 2009)*, Barcelona, Spain, pp. 1408–1412 (2009)
5. Doucet, A., Kazai, G., Dresevic, B., Uzelac, A., Radakovic, B., Todic, N.: Setting up a competition framework for the evaluation of structure extraction from OCR-ed books. *International Journal on Document Analysis and Recognition*, 1–8 (2010)
6. Giguët, E., Lucas, N.: The Book Structure Extraction Competition with the Resurgence software for part and chapter detection at Caen University. In: Geva, S., et al. (eds.) *INEX 2010. LNCS*, vol. 6932, pp. 128–139. Springer, Heidelberg (2010)

7. Grady, C., Lease, M.: Crowdsourcing document relevance assessment with mechanical turk. In: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, CSLDAMT 2010, pp. 172–179, Association for Computational Linguistics (2010)
8. Howe, J.: Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business, 1st edn. Crown Publishing Group (2008)
9. Kamps, J., Koolen, M.: Focus and Element Length in Book and Wikipedia Retrieval. In: Geva, S., et al. (eds.) INEX 2010. LNCS, vol. 6932, pp. 140–153. Springer, Heidelberg (2010)
10. Kazai, G., Kamps, J., Koolen, M., Milic-Frayling, N.: Crowdsourcing for book search evaluation: Impact of quality on comparative system ranking. In: SIGIR 2011: Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, New York (2011)
11. Kazai, G., Milic-Frayling, N., Costello, J.: Towards methods for the collective gathering and quality control of relevance assessments. In: SIGIR 2009: Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, New York (2009)
12. Larson, R.R.: Combining Page Scores for XML Book Retrieval. In: Geva, S., et al. (eds.) INEX 2010. LNCS, vol. 6932, pp. 154–163. Springer, Heidelberg (2010)
13. Le, J., Edmonds, A., Hester, V., Biewald, L.: Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In: SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation, pp. 21–26 (2010)
14. Preminger, M., Nordlie, R.: OUCs participation in the 2010 INEX Book Track. In: Geva, S., et al. (eds.) INEX 2010. LNCS, vol. 6932, pp. 164–170. Springer, Heidelberg (2010)
15. Quinn, A.J., Bederson, B.B.: Human computation: A survey and taxonomy of a growing field. In: Proceedings of CHI 2011 (2011)
16. Wilson, R., Landoni, M., Gibb, F.: The web experiments in electronic textbook design. *Journal of Documentation* 59(4), 454–477 (2003)

LIA at INEX 2010 Book Track

Romain Deveaud, Florian Boudin, 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 and present our contributions in the INEX 2010 Book Track. Digitized books are now a common source of information on the Web, however OCR sometimes introduces errors that can penalize Information Retrieval. We propose a method for correcting hyphenations in the books and we analyse its impact on the Best Books for Reference task. The observed improvement is around 1%.

This year we also experimented different query expansion techniques. The first one consists of selecting informative words from a Wikipedia page related to the topic. The second one uses a dependency parser to enrich the query with the detected phrases using a Markov Random Field model. We show that there is a significant improvement over the state-of-the-art when using a large weighted list of Wikipedia words, meanwhile hyphenation correction has an impact on their distribution over the book corpus.

1 Introduction

The number of books available in electronic format increases continuously. The mass-digitization of books is creating large digital libraries containing information about a broad range of topics. Well known examples of these digital libraries are the project Gutenberg¹ and Google Books², allowing people to read books for free on different devices (iPhone, iPad, Kindle...). The development of specialized Information Retrieval methods for this kind of documents is a real issue for the community.

The electronic representation of books are obtained using an Optical Character Recognition (OCR) process that automatically generates the machine-encoded text corresponding to the images of the pages. However it generally introduces some errors [10], 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 as two different words at the indexing step. Considering the following lines³:

¹ www.gutenberg.org

² books.google.com

³ Extract from *1984*, Georges Orwell, <http://books.google.com>

On each landing, opposite the lift shaft, the poster with the **enormous** face gazed from the wall.

In this example, the terms “*enor-*” and “*mous*” are indexed instead of “*enormous*”. As far as we know, no previous work has reported experiments on the impact of word hyphenation correction on book retrieval effectiveness. We propose in Section 2 a simple and efficient approach for correcting hyphenated words, which produce an almost errorless version of the corpus. We evaluate its performance on the Best Books for Reference task with the topics and the *qrels* of the 2009 and 2010 Book tracks.

Several studies had been conducted on Information Retrieval (IR) inside books in general, and some of them show that indexing specific parts (e.g. headers, titles or table of contents) is nearly as effective as indexing the entire content of books [6,11]. However, considering these information are not always available, we did not take into account these different parts in our experiments.

This year we tried different query expansion and query enrichment methods for the Best Books for Reference task. We started by using Wikipedia as an external source of knowledge for selecting informative words. A Wikipedia page is associated to each topic, and the most informative words of the pages are selected to expand the queries. This approach was already studied in [4] and we enhanced it with some features, different weighting schemes and several term extraction measures. We finally observed that on previous year *qrels* (2009), the *entropy* measure for selecting the words within the Wikipedia page gives the better results. As a second approach for modifying the query, we used the Stanford Parser [1] to extract phrases from the topics and applied a Markov Random Field model [8] to enrich the query.

The rest of the paper is organized as follows. In Section 2, we present our contribution with the correction of the hyphenated words in the corpus and its evaluation. Then, we detail in Section 3 our retrieval framework, our query expansion approaches and the runs we submitted for the Best Books for Reference task. Finally, we present our results in Section 4.

2 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 lexicon made of 118,221 unique words extracted from the English Gigaword corpus⁴.

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.

⁴ LDC Catalog No. LDC2007T07, Available at www ldc.upenn.edu

We evaluated the correction impact with the official *qrels* and topics from the 2009 and 2010 Book Tracks [23]. This year, 21 topics were validated over the 82 initially proposed, but there were fewer relevance judgements than in 2009, where only 16 topics were validated. Table 1 sums up some information about the collection. We can note that the standard deviation is high, it reveals that the sizes of the books vary a lot within the collection.

Table 1. Some important numbers about the book collection

Number of books	50,239
Size of the collection (words)	5,080,414,177
Size of the largest book	1,659,491
Size of the smallest book	590
Average size	101,125
Standard deviation of the size	102,127

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 configurations of the same retrieval model. This model uses a Language Modeling (LM) approach to IR with different Dirichlet prior smoothing (μ) values, along with the stopword list provided by Lemur and the Porter stemmer. Queries are generated from the <title> fields of the 2009 topics and the <query> fields of the 2010 topics. The number of retrieved books is set to 100. Results are reported in Table 2 for the 2009 topics and in Table 3 for the 2010 ones.

Table 2. Book retrieval results on both initial and corrected Book Track corpus, with the 2009 topics and qrels, 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

Despite the sizeable number of corrected words, the improvement is relatively low, however we note that it is in the same order for these two years ($\approx 1\%$). As said previously, books are larger than traditional web documents and there are very few words that appear only once in a book. Hence, the errors introduced by some misspelled words are greatly reduced.

Apart from the word hyphenation correction, we can see that the scores vary a lot between the 2009 and the 2010 topics. The very high scores achieved by

Table 3. Book retrieval results on both initial and corrected Book Track corpus, with the 2010 topics and qrels, 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, $\mu = 2500$	0.444	0.581	0.447	0.586
LM, $\mu = 1000$	0.435	0.576	0.439	0.581
LM, $\mu = 0$	0.390	0.528	0.393	0.524

this basic model on this year topics results from the small amount of assessments that could be collected. We used the corrected version of the corpus for all our further experiments and all the runs we submitted.

3 Best Books for Reference

3.1 Retrieval Model

All the runs and experiments we will further describe follow the same retrieval model. We use Indri, which is part of the Lemur project⁵ and provide an implementation of a LM approach for retrieval [7]. The embedded stoplist provided by Lemur is used for stopword removal along with the standard Porter stemmer.

Given a sequence of query terms $Q = (q_1, \dots, q_n)$ treated as a bag of words, the scoring function of a document D is defined as follow :

$$s_Q(D) = \prod_{i=1}^n p_D(q_i)^{\frac{1}{n}}$$

$p_D(\cdot)$ is estimated by Maximum Likelihood Estimation with Dirichlet prior smoothing:

$$p_D(q_i) = \frac{tf_{q_i,D} + \mu \times pc(q_i)}{|D| + \mu}$$

where \mathcal{C} is the entire collection, $|D|$ the size of the documents and $tf_{q_i,D}$ the frequency of the query term q_i in the document D . The μ parameter is empirically set to 2500, which is also the default value proposed by Indri.

3.2 Baselines

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. We submitted two other baselines (**baseline_1_wikifact** and

⁵ www.lemurproject.org

baseline_2_wikifact) which are exactly the same as before, except that we add the content of the `<wikifact>` field, when it is available. The text of this field corresponds to the first paragraph of a Wikipedia page identified as related to the topic. Queries are treated as bag of words and retrieval is performed using the model described in Section 3.1. Results are presented in Section 4.

3.3 Contextual Query Expansion Using Wikipedia

Several studies previously investigated the use of Wikipedia as an external corpus for Query Expansion [4,5,9,12]. In their approach, Koolen *et al.* [4] extract useful terms from Wikipedia pages to expand queries and use them for Book Retrieval. A page is selected by querying Wikipedia with the original query and getting the page that matches 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 this work.

We use the `<wikiurl>` field, when available, to get a Wikipedia page closely related to each topic. Otherwise we query the Wikipedia search engine with the `<query>` field and we select the best ranked article. Then we extract terms from this article using different measures described below (*tf*, *tf.idf*, *entropy*...) and we use them to expand the query. We also keep the scores of these term selection measures in order to weigh the words inside the query. Indeed, some terms are more important than other in the Wikipedia page, and a representation of this relative importance is the score of the measure. A weight is therefore associated to each selected term.

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. Given a sequence of query terms $Q = (q_1, \dots, q_k)$, a sequence of fact terms $F = (f_1, \dots, f_m)$ and a list of weighted terms $T_Q = \{(t_1, w_1), \dots, (t_n, w_n)\}$ extracted from related Wikipedia pages, we rank books according to the following scoring function $\Delta_Q(D)$:

$$\Delta_Q(D) = \left(\prod_{i=1}^k p_D(q_i)^{\frac{1}{k}} \right)^{\frac{X}{X+Y+Z}} \times \left(\prod_{i=1}^m p_D(f_i)^{\frac{1}{m}} \right)^{\frac{Y}{X+Y+Z}} \times \left(\prod_{i=1}^n p_D(t_i)^{\frac{w_i}{\sum_{j=1}^n w_j}} \right)^{\frac{Z}{X+Y+Z}}$$

Here, we could not learn an appropriate weighting scheme for the X , Y and Z weights, so they were set empirically. We gave the same weight to the `<query>`

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 Tree-Tagger⁶. 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 within 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 an *entropy* measure to rank the words accordingly to their informativeness, and the 10 words with best score are selected for the expansion. Considering a sequence of words $S = (w_1, \dots, w_n)$, the *entropy* measure we use is defined as follow:

$$E(S) = - \sum_{i=1}^n p(w_i) \log_2(p(w_i))$$

Where the $p(w_i)$ are computed within the whole Wikipedia article.

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.

3.4 Using the Stanford Parser

In this model, 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**.

⁶ www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

3.5 Applying the Book Track’s Contextual Query Expansion to the Ad Hoc Track

In this Section we briefly present the generalization of the approach we tried in the Book Track. This approach consists of expanding the user query with contextual words taken from a Wikipedia page related to the topic. The contextual Wikipedia page is obtained by querying the Wikipedia search engine with the original user query (which is the `<title>` field in the INEX Ad Hoc topics). A page is automatically selected if the query matches its title. Otherwise, we assume that the first result returned by the Wikipedia search engine is relevant enough.

We developed a specific tool named Mirmiri⁷ to achieve this automatic selection. This tool is a Ruby library which provides some utilities related to Information Retrieval and Text Processing in general. It is Open-Source and available for free⁸.

We produced a run which expands the original query with 200 words taken from the related Wikipedia page, and we evaluated it with the official *qrels*. The retrieval is done at the Document level (i.e. Relevance in Context task).

Table 4. Document retrieval results on the Relevance in Context task in terms of Mean Average Precision (MAP) and Precision at 5 and 10 documents (P@5 and P@10)

Run	MAP	P@5	P@10
p22-Emse301R	0.4292	0.6962	0.6423
qe_wiki_entropy	0.3858	0.6500	0.6077
Reference Run	0.3805	0.6423	0.5750
baseline	0.3496	0.6115	0.5596

We can see in Table 4 that it does not beat the best run of this year (p22-Emse301R) for the Relevance in Context Task. However it outperforms the Reference Run given by the INEX organizers which was high this year, due to the tuning on the 2009 topics and judgements. The **baseline** run we used in this experiment is obtained by using only the `<title>` fields as a bag of words (i.e. the same run as `qe_wiki_entropy` but without expansion terms). The results show that expanding the **baseline** queries with Wikipedia terms leads to an improvement of the performance which is situated in the order of 10%. These different results confirm that this approach performs well for the retrieval of whole documents.

4 Results

The official results of the Book Track are presented in Table 5.

⁷ mimir.org

⁸ github.com/romaindeveaud/mimir

Table 5. Evaluation results of all runs submitted for the Best Books for Reference task. Our run identifiers are prefixed with p98.

Runs	MAP	P@10	NDCG@10
p14-BOOKS2010_T2_PAGE_SUM_300.trec	0.5050	0.6667	0.6579
p98-baseline_2_wikifact.trec	0.5044	0.6381	0.6500
p98-baseline_2.trec	0.4806	0.6143	0.6302
p98-fact_query_tfwiki.trec	0.4706	0.5571	0.5919
p98-fact_stanford_deps.trec	0.4573	0.5857	0.5976
p98-baseline_1_wikifact.trec	0.4565	0.5905	0.5960
p98-baseline_1.trec	0.4374	0.5810	0.5764
p98-fact_query_10wikibests.trec	0.4328	0.5714	0.5638
p98-fact_query_entropy.trec	0.4250	0.5476	0.5442
p14-BOOKS2010_T2FB_BASE_BST.trec	0.3981	0.5048	0.5456
p98-fact_query_tfidfwiki.trec	0.3442	0.4667	0.4677
p6-inex10.book.trec	0.3286	0.4429	0.4151
p6-inex10.book.fb.10.50.trec	0.3087	0.4286	0.3869
p14-BOOKS2010_CLM_PAGE_SUM_300.trec	0.1640	0.2810	0.2156
p14-BOOKS2010_CLM_PAGE_SUM.trec	0.1507	0.2714	0.2017

We can see that our baselines that use the `<fact>` fields achieve the best results. This behaviour can be explained by the fact that the `<fact>` are actual full sentences taken from the books for most of the topics. Again, relevance judgements are a bit insufficient considering the number of topics, and it favours the books containing these sentences. Assessments sparsity also explains the high scores achieved by the best runs. The `fact_stanford_deps`, ranked fifth, also performed well considering that no external resources are involved except a dependency parser.

We see that the *term-frequency* measure for selecting expansion words within a Wikipedia page performs better than the *tf.idf* or the *entropy*. It denies our initial intuition which was that the *entropy* would perform better. Indeed the good results presented in Section 3.5 led us to think that query expansion was an efficient approach, and that the *entropy* was an appropriate measure for selecting important and informative words. There are two main reasons that can explain the relative gap between the Ad Hoc and the Book results for the same method. First, the vocabulary of a common encyclopedia and of 19th century books is very different, and can cause word mismatch. Second, the fact that full sentences directly taken from the books appear in the topics highly favours the query words. The small number of assessments collected also plays a major role here.

To highlight this problem we experimented the same method with the 2009 topics and *qrels* and compared it with the best run from last year and the same baseline as `baseline_1`. The results presented in Table 6 show that the *entropy* measure achieves the best results overall when performing the contextual query expansion we presented in Section 3.3. This experiment indicates that this approach is efficient and can achieve high scores even with very different documents.

Table 6. Evaluation results for the 2009 INEX Book Retrieval task in terms of Mean Average Precision (MAP) and Precision at 10 documents (P@10)

Run	MAP	P@10
qe_wiki_entropy	0.363	0.593
BR_linex09.book.fb.10.50 (best 2009 run)	0.347	0.486
baseline_1	0.304	0.507

5 Conclusions

In this paper we presented our contributions to the INEX Book Track. We proposed to enhance Book Search performance by correcting word hyphenations and produced a corrected version of the collection. Although we cannot see a significant improvement on a Book Retrieval task the retrieval accuracy of the models we experimented were all enhanced. We expect that this correction can lead to better better in focused search tasks such as page or extent retrieval.

We also presented the runs we submitted within the Best Books for Reference task. Our baselines achieved the best results mainly because of the topics that were containing unmodified sentences from books, and also because of the small number of relevance judgements collected. However we evaluated the query expansion approach on the 2009 topics and *qrels* and we showed that an appropriated weighting scheme combined to a score normalization between the terms of the expansion leads to better results.

References

1. De Marneffe, M.C., MacCartney, B., Manning, C.D.: Generating typed dependency parses from phrase structure parses. In: Proceedings of LREC 2006 Conference (2006)
2. Kazai, G., Doucet, A., Koolen, M., Landoni, M.: Overview of the INEX 2009 Book Track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 145–159. Springer, Heidelberg (2010)
3. Kazai, G., Koolen, M., Doucet, A., Landoni, M.: Overview of the inex 2010 book track: At the mercy of crowdsourcing. In: Geva, S., et al. (eds.) INEX 2010. LNCS, vol. 6932, pp. 98–117. Springer, Heidelberg (2011)
4. Koolen, M., Kazai, G., Craswell, N.: Wikipedia pages as entry points for book search. In: Proceedings of the Second ACM International Conference on Web Search and Data Mining. WSDM 2009, pp. 45–53. ACM, New York (2009)
5. Li, Y., Luk, W.P.R., Ho, K.S.E., Chung, F.L.K.: Improving weak ad-hoc queries using wikipedia as external corpus. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR 2007, pp. 797–798. ACM, New York (2007)
6. Magdy, W., Darwish, K.: Book search: indexing the valuable parts. In: Proceeding of the 2008 ACM Workshop on Research Advances in Large Digital Book Repositories, Books Online 2008, pp. 53–56. ACM, New York (2008)

7. Metzler, D., Croft, W.B.: Combining the language model and inference network approaches to retrieval. *Inf. Process. Manage.* 40, 735–750 (2004)
8. Metzler, D., Bruce Croft, W.: A markov random field model for term dependencies. In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR 2005*, pp. 472–479. ACM, New York (2005)
9. Milne, D.N., Witten, I.H., Nichols, D.M.: A knowledge-based search engine powered by wikipedia. In: *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM 2007*, pp. 445–454. ACM, New York (2007)
10. Taghva, K., Borsack, J., Condit, A.: Results of applying probabilistic ir to ocr text. In: *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1994*, pp. 202–211. Springer, New York (1994)
11. Wu, H., Kazai, G., Taylor, M.: Book Search Experiments: Investigating IR Methods for the Indexing and Retrieval of Books. In: Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., White, R.W. (eds.) *ECIR 2008. LNCS*, vol. 4956, pp. 234–245. Springer, Heidelberg (2008)
12. Yu, X., Jones, G.J.F., Wang, B.: Query dependent pseudo-relevance feedback based on wikipedia. In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR 2009*, pp. 59–66. ACM, New York (2009)

The Book Structure Extraction Competition with the Resurgence Software for Part and Chapter Detection at Caen University

Emmanuel Giguet and Nadine Lucas

GREYC Cnrs, Caen Basse Normandie University
BP 5186 F-14032 Caen Cedex, France
`name.surname@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, part and chapter. The main idea is to use a model describing relationships for elements in the document structure. Frontiers between high-level units are detected, parts and then chapters. 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 [6]. The INEX Resurgence software used at Caen University to structure voluminous documents was modified to handle book parts, on top of chapters. The original Resurgence software processes academic articles (mainly in pdf format) and news articles (mainly in HTML format) in various text parsing tasks [8].

The experiment was conducted from pdf documents to ensure the control of the entire process. The document content is extracted using the pdf2xml software [2]. In the first experiment, we handled only the chapter level [7]. 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 and we detail the actual results on the INEX book corpus, both the 2010 corpus and the 2009 one. In section 3, we discuss the advantages of our method and make proposals for future competitions. In the last section we sum up our contribution.

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 here 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 post-face with different layout. It ultimately retrieves the page body in a page, surrounded by margins [8].

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.

Chapter Title Detection throughout the document was conducted using a sliding window to detect chapter transitions with two patterns, as explained in [7].

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.

Parts 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:

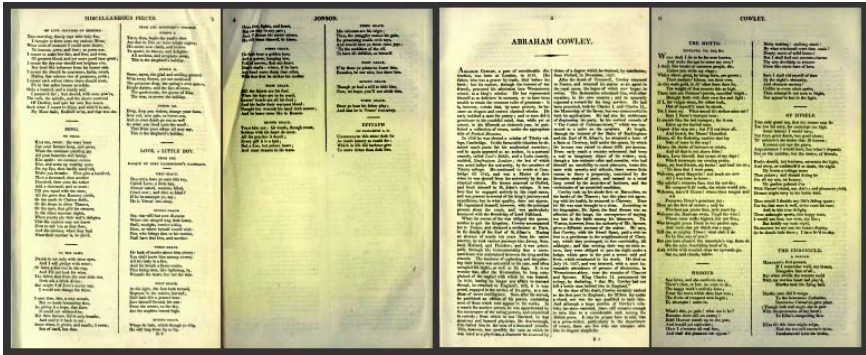


Fig. 1. View of the four-page sliding window to detect chapter beginning. Pattern 1 matches. Excerpt from 2009 book id = 00AF1EE1CC79B277.

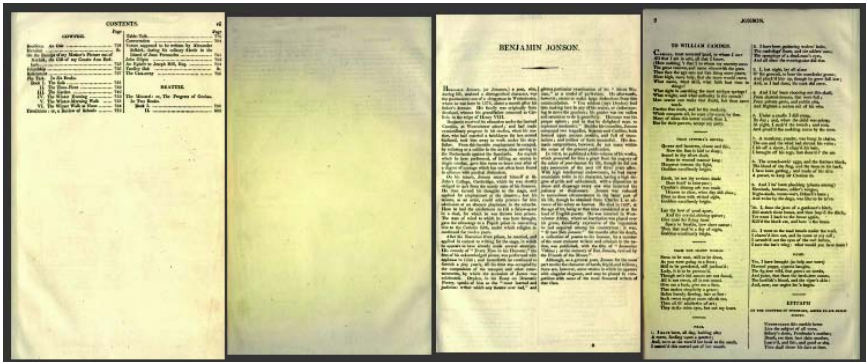


Fig. 2. View of the four-page sliding window to detect chapter announced by a blank page. Pattern 2 matches. Excerpt from 2009 book id= 00AF1EE1CC79B277.

- the text body in the page is mainly blank,
- with a blank line at least 5 times the standard line space height;
- followed by 1 to 3 written lines;
- with a blank line 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 3 illustrates the pattern. It applies on a single page. 3 context pages are given but are not used in the process.

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).

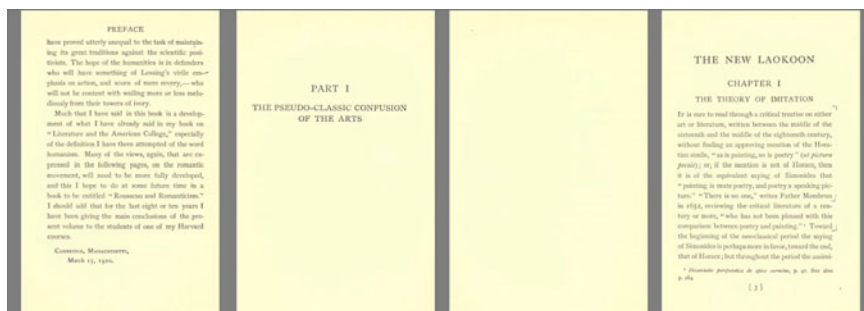


Fig. 3. View of the one-page sliding window to detect parts beginning. 3 context pages (1 page before, two pages after) are given but not used. Excerpt from 2009 book id = 2A5029E027B7427C.

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 [7]. 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.

2.4 Experiment

The corpus provided in 2010 was extended as compared with the 2009 one. It comprised 1114 books instead of 1000 in 2009. The GREYC 2010 program detected only part and chapter titles. No effort was exerted to find the section titles and sub-titles. On the practical side, the team was interested in handling voluminous documents, such as textbooks and cultural heritage books, hence the interest in INEX. The top-down strategy and the highest levels in the book

hierarchy were favoured because this is the most useful step when filtering large book collections, in text mining tasks for instance. Moreover, most if not all techniques start from the lower levels. Reasonable results can be obtained for those levels with existing programs once the relevant parts or chapters have been retrieved.

There was only one run.

2.5 Results

GREYC Results. The entire corpus was handled, but 26 books were not analyzed. This was due to lack of time, since we could not use parallel computing this time, as we did in 2009 for the pdf2XML task.

The official results for 2010 are given in Table 1 and are compared with the first official evaluation in 2009 in Table 2. However, the very bad results in 2009 were due to a bug in page numbers.

Table 3 shows the complementary alternative evaluation based on links and provided by Xerox Research Center Europe (XRCE).

The 2010 results outperform the 2009 results as expected. This is mainly explained by improvements in the system calibration. Little gain is obtained from part detection. This is due to the fact that the number of parts is low (and even often null in individual books), as compared to the total number of sections

Table 1. Official evaluation 2010 on 2010 ground truth (641 books)

Results 2010	Precision	Recall	F-Measure
Titles	18,03%	12,53%	12,33%
Levels	13,29%	9,60%	9,34%
Links	14,89%	10,17%	10,37%
Complete entries	10,89%	7,84%	7,86%
Entries disregarding depth	14,89%	10,17%	10,37%

Table 2. Official evaluation 2009 against 2009 ground truth (527 books)

Results 2009	Precision	Recall	F-Measure
Titles	19,83%	13,60%	13,63%
Levels	16,48%	12,08%	11,85%
Links	1,04%	0,14%	0,23%
Complete entries	0,40%	0,05%	0,08%
Entries disregarding depth	1,04%	0,14%	0,23%

Table 3. Alternative 2010 evaluation with Xerox linked-based metrics

	XRCE Link-based Measure			
	Links			Accuracy (for valid links)
	Precision	Recall	F1	Title
GREYC 2010	63.9	39.5	42.1	47.6

and subsections to be found throughout the collection. The official evaluation does not distinguish false responses from nonresponses, so all sections titles are considered as false.

Comparison. GREYC was the sole participant in 2010 and was evaluated on a slightly extended book collection (1114 books). The ground truth contained 641 books. Hence, results could not be compared directly with the 2009 participants results on the 2009 corpus.

Results were therefore compared against the 2009 ground truth that comprised 527 books using the Python evaluation toolkit provided on sourceforge [3]. GREYC results were compared with results obtained by the 2009 official best run (run 3), 2009 results after correction of the page bug by Xerox (called GREYC-1 in [75] and GREYC-1C 2009 in the tables below to avoid confusion with runs), and the 2010 results. They were also compared with the results obtained on the same ground truth by other participants in 2009 (Table 4).

Table 4. Alternative evaluation comparing all participants against the 2009 ground truth

	XRCE Link-based Measure			
	Links			Accuracy (for valid links)
	Precision	Recall	F1	Title
MDCS	65.9	70.3	66.4	86.7
XRCE-run3	69.7	65.7	64.6	74.4
Noopsis	46.4	38.0	39.9	71.9
GREYC run 3	6.7	0.7	1.2	13.9
GREYC-1C 2009 page bug correction	59.7	34.2	38.0	42.1
GREYC 2010	64.4	38.9	41.5	47.6

Table 5 shows another tentative measure, provided by [43] to check if accuracy measured on title wording (INEX 08 like measure) could be useful, along with a level accuracy measure based on correct title retrieval. Results are given for the official best GREYC run (run 3) and for the results with page number correction GREYC-1C.

Table 5. Alternative accuracy evaluation with INEX 08 like measures for all participants, against the 2009 ground truth

	INEX08 like measure	
	Accuracy	
	Title	Level
MDCS	86.7	75.2
XRCE-run3	74.4	68.8
Noopsis	71.9	68.5
GREYC run 3	0.0	31.4
GREYC-1C 2009	42.1	73.2
GREYC 2010	22.3	64.2

3 Discussion

GREYC was the only candidate this year, but since official results in 2009 suffered from a gross error in page numbers, it was worth re-evaluating results on a comparable corpus. Moreover, the book part level was also tested. The low recall is still due to the fact that the full hierarchy of titles was not addressed as mentioned earlier. This will be addressed in the future.

3.1 Reflections on the Experiment

Despite shortcomings, mostly due to early stage development, the INEX book structure extraction competition is very interesting. The corpus provided for the INEX Book track is the best available corpus offering full books at document level [9]. Although it comprises mostly XIXth century printed books, it is very valuable for it provides various examples of layout. Besides, this meets our requirements for electronic use of patrimonial assets. The ground truth is manually corrected, so the dataset is easier to work with than the dataset provided by [10].

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 body, not on the table of contents;
- It handles books without table of contents (ToC), 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.

The fact that typographical position is very stable in the corpus reflects real-life conventions in book printing.

The advantage of using the book body is clear when comparing two datasets, books without ToC and books with ToC [5,1]. The difference is clearer in the GREYC case with the link-based measure.

Another advantage is robustness. Since no list of expected and memorized forms is used, but position and distribution 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 explicit

Table 6. Comparison of results on two books datasets after [5]

	whole dataset (precision / recall)	no-ToC dataset (precision/ recall)
MDCS	65.9 / 70.3	0.7 / 0.7
XRCE	69.7 / 65.7	30.7 / 17.5
NOOPSIS	46.4 / 38.0	0.0 / 0.0
GREYC-1C 2009	59.7 / 34.2	48.2 / 27.6

mention such as *chapter*, they are found as well, for instance a plain title stating “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. This approach reflects an original breakthrough to improve robustness and proves very useful to generate ToCs to help navigate digitized books when none was provided in the printed version.

3.2 Reflections on Evaluation Measures

Concerning evaluation rules, the very small increment in quantified results did not reflect our qualitative assessment of a significant improvement. Though this is a subjective impression that seems fairly common, we were puzzled.

Generally speaking, the ground truth is still very coarse and it mostly relies on automated results depending on the ToC. If the ToC is the reference, it is an error to extract prefaces, for instance, because they generally do not figure in ToCs. In the same way, most ToCs do not reflect the whole hierarchy of sections and subsections, but skip lower levels. The participants using the book body as main reference are penalized if they extract the whole hierarchy of titles as it appears in the book, when the ToC represents only higher levels.

For all participants, accuracy on titles seems to be a thorny question, because there is a huge difference in title accuracy as calculated by INEX organizers from the retrieval of the wording, and title accuracy as calculated by XRCE from the links [1]. In the INEX08-like measure on accuracy for title and level provided by XRCE, the figures decrease while precision and recall grow.

A test was made to evaluate level accuracy, since proceeding one level at a time allowed a relevance check on this measure. In 2009 GREYC calculated only chapters and the level accuracy was high, 73.2, in the GREYC results, after correction on the page bug. Scores in level accuracy in 2010 were calculated with part and chapter level information and then without part and chapter level information to check consistency (Table 7).

Level accuracy according to title wording was called Inex08 like measure after [4]. The difference in level accuracy raises questions. Results in 2009 (GREYC-1C with page bug correction) were given for one level only, namely chapters. The submitted 2010 GREYC results with level information for part and chapter levels had a level accuracy of 64.2, but when level information was scrapped, it was better, 77.9. It should be the contrary. Another intriguing observation is that linked-based title accuracy and text-based level accuracy did not evolve together. Our guess is that level accuracy is not relevant, for it is calculated from the XML with relative depth for each book, and not against a standard layout scale for the entire collection.

Title accuracy improved in 2010 according to both XRCE link based measure and Inex08-like title wording based measure. This is explained by a better rightward segmentation, already tested in 2009 after the runs and mentioned in [7]. However, title accuracy according to the INEX08 like calculation sharply collapsed between 2009 and 2010.

Table 7. GREYC alternative link-based evaluation with and without level information against the 2009 ground truth

	XRCE Link-based Measure					
	Links			Title accuracy for valid links	Inex08 like Accuracy	
	Precision	Recall	F1		Title	Level
GREYC-1C 2009	59.7	34.2	38.0	13.9	42.1	73.2
GREYC 2010	64.4	38.9	41.5	47.6	22.3	64.2
GREYC 2010 with- out level info	64.4	38.9	41.5	47.6	22.3	77.9

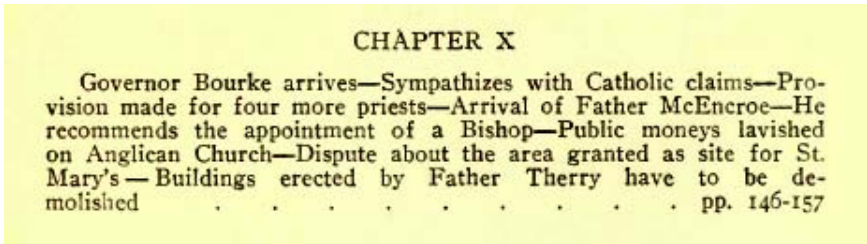


Fig. 4. View of the ToC entry. Excerpt from 2010 book id =A803EBAC7E50C7D0.

Since GREYC was the only candidate working from the actual book body layout and not after the ToC, results suffered from the fact that ToC when present is used as the reference in the groundtruth. However, there is a significant difference between ToC and book titles. Sometimes, the mention *chapter* was not found in the book or was abbreviated as C in the ToC. Although differences in case, such as CHAPTER III in the book and Chapter 3 in the ToC are cared for in the evaluation, by using case insensitive option, differences in numbering and word segmentation penalized our results. The edit distance error margin seems to be wide, but we tried a Levenshtein distance error margin of 20% and found it is not sufficient, confirming other findings as suggested by [51].

In some cases, detailed subentries were included in the chapter title, while they are not present in the ToC, or vice versa, as explained in [5]. All these details explain very low results in title accuracy. Figure 4 compared with Figure 5 shows an example where subentries are included in the ToC and in the reference deriving from it, but not on the corresponding page in the book body. Here is the ground truth entry for the book *Life of Archpriest Therry* (book id A803EBAC7E50C7D0) Chapter X, p. 146.

```
<toc-entry title="CHAPTER X Governor Bourke arrives Sympathizes with Catholic claims Pro vision made for four more priests Arrival of Father McEncroe He recommends the appointment of a Bishop Public moneys lavished on Anglican Church Dispute about the area granted as site for St. Mary s Buildings erected by Father Therry have to be de molished pp." page="206" />
```

4 Proposals

The bias introduced by a semi-automatically constructed ground truth was salient as can be seen in Figure 4 above, where split words or added pp. at the end of the entry illustrate poor quality against human judgment. Manually corrected annotation is still to be checked to improve the ground truth quality. As mentioned in [9,11] quantitative effort is also needed, but it is time-consuming. However, it might not be realistic to expect a clean unique reference for a large book collection. It might be better to handle parameters according to the final aim of the book processing, such as navigation or information filtering. Thus known automatic biases might be countered or valued in the performance measure.

One simple idea would be to consider equally right results for titles matching with either the ToC or the book body. 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,

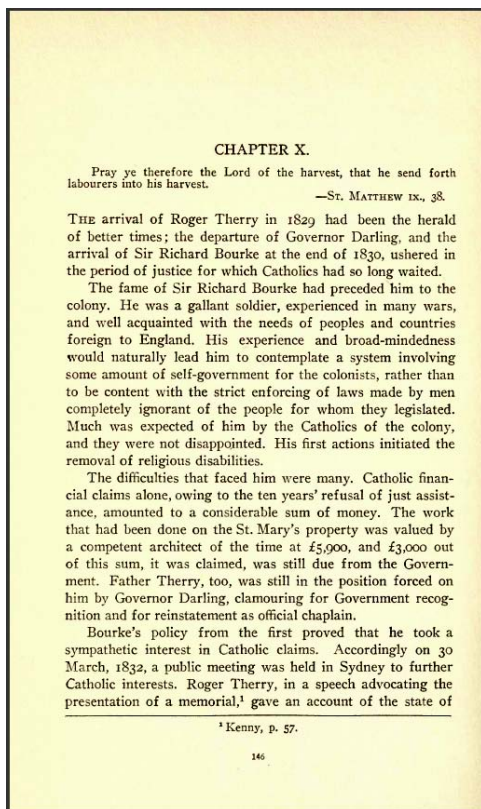


Fig. 5. View of first page of Chapter X (p. 146). Excerpt from 2010 groundtruth, book id = A803EBAC7E50C7D0.

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.

Concerning details, it should be clear whether or when the prefix label 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 [4,5], because providing complete and accurate results for one or more levels 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.

The time is ripe for eliciting effective face-to-face interaction between participants, to stimulate discussion and make the evaluation rules evolve faster. This should entice new participants to enter an important field of development with a lively discussion ahead.

5 Conclusion

The paper presents a strategy of detecting parts and chapters in a book without the use of the table of contents, using only layout features of the scanned pages. The strategy is mostly easy to follow and is reproducible.

Acknowledgments. We thank student Alexandre Baudrillart who implemented new features in the program during his summer job at GREYC, Caen University. We also thank anonymous reviewers for their excellent comments and suggestions.

References

1. Doucet, A., Kazai, G., Drešević, B., Uzelac, A., Radaković, B., Todić, N.: Setting up a competition framework for the evaluation of structure extraction from ocr-ed books. *International Journal of Document Analysis and Recognition (IJ DAR)*, Special Issue on Performance Evaluation of Document Analysis and Recognition Algorithms 14(1), 45–52 (2011)
2. Déjean, H.: pdf2xml open source software, <http://sourceforge.net/projects/pdf2xml/> (last visited March 2010)
3. Déjean, H., Meunier, J.-L.: INEX structure extraction groundtruth, <https://sourceforge.net/projects/inexse/> (last update 2010-09-29, last visited March 2011)
4. Déjean, H., Meunier, J.L.: XRCE Participation to the Book Structure Task, pp. 124–131. Springer, Heidelberg (2009), <http://portal.acm.org/citation.cfm?id=1611913.1611928>

5. Déjean, H., Meunier, J.L.: Reflections on the INEX structure extraction competition. In: Proceedings of the 9th IAPR International Workshop on Document Analysis Systems, DAS 2010, pp. 301–308. ACM, New York (2010), <http://doi.acm.org/10.1145/1815330.1815369>
6. Doucet, A., Kazai, G.: Icdar 2009 book structure extraction competition. In: IEEE (ed.) 10th International Conference on Document Analysis and Recognition IC-DAR 2009, Barcelona, Spain, pp. 1408–1412 (2009)
7. Giguët, E., Lucas, N.: The book structure extraction competition with the Resurgence software at Caen university. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 170–178. Springer, Heidelberg (2010)
8. Giguët, E., Lucas, N., Chircu, C.: Le projet Resurgence: Recouvrement de la structure logique des documents électroniques. In: JEP-TALN-RECITAL 2008, Avignon, France (2008)
9. Kazai, G., Doucet, A., Koolen, M., Landoni, M.: Overview of the INEX 2009 book track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 145–159. Springer, Heidelberg (2010), <http://portal.acm.org/citation.cfm?id=1881065.1881084>
10. Vincent, L.: Google book search: Document understanding on a massive scale. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02, pp. 819–823. IEEE Computer Society, Washington, DC, USA (2007), <http://portal.acm.org/citation.cfm?id=1304596.1304903>

Focus and Element Length for Book and Wikipedia Retrieval

Jaap Kamps^{1,2} and Marijn Koolen¹

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 [10], 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 [16] for indexing and retrieval. Our indexing approach is based on earlier work [1, 3, 5, 13–15].

- *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 [11]. 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 [8] and on the larger INEX 2009 collection [12] with two sets of category labels [11]; one based on the *Wikipedia* category structure and one based on the *WordNet* category labels. [11] 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.

The category distance scores are computed as follows. For each target category we estimate the distances to the categories assigned to retrieved document,

similar to what is done in [17]. The distance between two categories is estimated according to the category titles. In previous experiments we also experimented with a binary distance, and a distance between category contents, but we found the distance estimated using category titles the most efficient and at the same time effective method.

To estimate title distance, we need to calculate the probability of a term occurring in a category title. To avoid a division by zero, we smooth the probabilities of a term occurring in a category title with the background collection:

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

where C is the category title and D is the entire wikipedia document collection, which is used to estimate background probabilities. We estimate $P(t|C)$ with a parsimonious model [2] that uses an iterative EM algorithm as follows:

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

The initial probability $P(t|C)$ is estimated using maximum likelihood estimation. We use KL-divergence to calculate distances, and calculate a category score that is high when the distance is small as follows:

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

where d is a retrieved document, C_t is a target category and C_d a category assigned to a document. The score for a document in relation to a target category $S(d|C_t)$ is the highest score, or shortest distance from any of the document's categories to the target category. So if one of the categories of the document is a target category, the distance and also the category score for that target category is 0, no matter what other categories are assigned to the document. Finally, the score for a document in relation to a query topic $S(d|QT)$ is the sum of the scores of all target categories:

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

Besides the category score, we also need a query score for each document. This score is calculated using a language model with Jelinek-Mercer smoothing with a linear length prior:

$$P(q_1, \dots, q_n|d) = P(d) \sum_{i=1}^n \lambda P(q_i|d) + (1 - \lambda)P(q_i|D)$$

where $P(d)$ is proportional to the document length, computed as:

$$P(d) = \frac{|d|}{|D|}$$

Finally, we combine our query score and the category score through a linear combination. For our official runs both scores are calculated in the log space, and then a weighted addition is made.

$$S(d|QT) = \mu P(q|d) + (1 - \mu)S_{cat}(d|QT)$$

Based on previous experiments [8], we use $\mu = 0.9$.

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 $S(d) = 0.7ArticleRF(d) + 0.3AnchorLen(d)$. 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, systems are restricted to return no more than 1000 characters per topic. 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, systems are restricted to returning no more than 1500 results. 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, systems are restricted to returning no more than 500 characters per result. 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 [3, 6]. 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 [7], 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 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. The Fusion run is

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
<i>Base SecLen</i>	0.0723	0.3308	0.2910	0.2184	0.1944
<i>Fusion Sec</i>	0.0678	0.3027	0.2694	0.2110	0.1906

less effective than the Base run. The anchor text does not help early precision, although we did not range over all possible weighted combinations. Perhaps a lower weight on the anchor text might be beneficial. 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.

2.5 Focused Task

We have no overlapping elements in our indexes, so no overlap filtering is done. However, systems can return no more than 1000 characters per topic. This means the result list needs to be cut-off. Articles tend to be longer than 1000 characters, so for the article-level runs, some part of an article needs to be selected to be returned. Section elements might also be longer, in which case a part of a section needs to be selected. We choose to select the first 1000 characters returned in retrieval order, based on the idea that precision tends to drop over ranks. For example, if the highest ranked section has 500 characters and the second ranked section has 600 characters, we return the highest ranked section and the first 500 characters of the second ranked section. If the highest ranked section or article is longer than 1000 characters, we select the first 1000 characters.

Because relevant text tends to start near the beginning of XML elements [4], we also looked at a method that cuts off results after 100 characters, so that the initial text from multiple high-ranked results is returned.

The mean length of Sec results is 936, although most results have far fewer (median is 397) characters. The first 1000 characters often corresponds to more than one section. The SecLen and Base results are longer, with a median length of 2,227 and 6,098 characters respectively (mean lengths are 5,763 and 13,383). For most topics, the first returned result needs to be cut-off after the first 1000 characters.

How does precision among the first 1000 characters correspond to older Focused evaluation measures, such as interpolated precision at the first percentage of recall (iP[0.01])? The median number of characters per topic is 129,440 (mean

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

Run id	# ret.	# rel.	ret. $P_{article}$	P_{char}	iP[0.00]	iP[0.01]
<i>Base Sec F1000 Topic</i>	1.25	0.38	0.3301	0.1232	0.1694	0.0386
<i>Base Sec F100 Article</i>	10.17	3.17	0.3105	0.1162	0.2468	0.0338
Sec F1000 Topic	3.94	2.13	0.5196	0.2340	0.3372	0.1087
SecLen F1000 Topic	1.56	0.90	0.5667	0.2975	0.3188	0.1261
*Base Sec F1000 Topic	1.29	0.81	0.6250	0.3490	0.4012	0.1376
Base SecLen F1000 Topic	1.29	0.81	0.6186	0.3526	0.3903	0.1518
Base F1000 Topic	1.10	0.69	0.6250	0.2806	0.2828	0.0737
Sec F100 Article	10.63	4.87	0.4576	0.2127	0.4117	0.0842
SecLen F100 Article	10.08	4.85	0.4804	0.1981	0.4403	0.0934
*Base Sec F100 Article	10.06	5.27	0.5229	0.2445	0.4626	0.1140
Base SecLen F100 Article	10.06	5.27	0.5229	0.2677	0.5015	0.1226
Base F100 Article	10.00	5.23	0.5231	0.1415	0.2623	0.0340

339,252). For most topics, precision of the first 1000 retrieved characters is somewhere between iP[0.00] and iP[0.01].

Table 2 shows the results for the Focused Task, where $P_{article}$ is the article-level precision and P_{char} is the character-level precision. Article-level precision is based on binary relevance judgements. If the returned (part of the) article contains relevant text, the article is considered relevant at this level. For the character level precision, the set-based precision of all returned characters is computed as the number of returned highlighted characters divided by the total number of returned characters. As expected, the P_{char} for all the runs is somewhere between iP[0.00] and iP[0.01]. The first 1000 retrieved characters (denoted F1000 Topic) gives higher precision than first 100 per article up to 1000 characters (denoted F100 Article). But by 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.

We note that, although the F100 Article runs have a lower P_{char} than the F1000 Topic runs, they have a higher iP[0.00] score. This is because they return smaller and therefore more results. Relevant text tends to be concentrated near the start of XML elements [4], so it is often beneficial to return text from the start of an element. In the case of the F1000 Topic runs, the first retrieved result is usually 1000 characters long, so iP[0.00], which is determined after the first relevant retrieved result, is based on the first 1000 characters of an element, be it article or section. For the F100 Article runs, iP[0.00] will often be based on

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]
<i>Base Sec Len</i>	0.0597	0.1492	0.1330	0.1080	0.1031
<i>Fusion Sec</i>	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]
<i>Base F500 Article</i>	0.0576	0.1439	0.1191	0.1053	0.0980
<i>Base Sec F500 Article</i>	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

a smaller number of characters. If the first 500 characters of the highest ranked section or article are relevant, taking the first 1000 characters from that section or article gives an $iP[0.00]$ of 0.5, while taking the first 100 characters gives an $iP[0.00]$ of 1.

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:

- 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 [9]. 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. This index contains 50,239 books.

1-Page Index: each individual page is indexed as a retrievable unit. This index contains 16,105,669 1-page units.

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. Note that a book with 53 pages is divided into 11 units: 10 units with 5 pages and 1 unit with 3 pages. Most books have more than 300 pages, so the number of units with less than 5 pages is small; less than 2% of units in the index, in fact. The average number of pages per unit is 4.97. This index contains 3,241,347 5-page units.

For the 2010 Book Track, there are 83 topics in total. Each of these topics consists of a factual statement derived from a book in the collection, as well as a query with the most important words in the factual statement. Of these 83 topics, 21 have relevance assessments for pages pooled from the official runs. For the Best Book task, the aim is to return the best books in the collection on the general topic of the statement. For the Prove It task, the aim is to return pages that either refute or confirm 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.

Table 5. Page level statistics for the 2010 Prove It runs

Run	# pages	# books	# pages/book
1-page	985	464	2.1
1-page RF	1000	444	2.3
5-page	988	112	8.8
5-page RF	1000	105	9.5

Table 6. Results for the INEX 2010 Best Book Task

Run	MAP	P@5	ndcg@5	P@10	ndcg@10
Book	0.3286	0.4952	0.4436	0.4429	0.4151
Book-RF	0.3087	0.4667	0.3948	0.4286	0.3869

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. Because the returned results need to be individual pages, each 5-page unit is segmented into its individual pages. The highest ranked 5-page unit is split into 5 pages with each page receiving the same score as the 5-page unit. Thus, the first N retrieved units are presented as $5N$ results.

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.

3.1 Prove It Run Analysis

With the single page index, we get an average of 2.1 pages per book, while the 5-page index gives us 8.8 pages per book. With an average 4.97 pages per unit, that is 1.77 units per book. The 5-page index gives us less units per book than the single page index, but we get more pages from a single book. The impact of relevance feedback is a slight increase in the number of pages from the same book. Expanding the query with terms from the top ranked book pages probably results in retrieving more pages from those books because similar terms are used throughout the same book.

3.2 Best Book Results

In Table 6 we see the results for the Best Book Task. The standard Book retrieval system performs better than the system with pseudo relevance feedback. PRF expands the topic of the query somewhat, which is unnecessary for the task of finding only the best books on a topic.

Books are have graded relevance judgements, with labels *excellent*, *good*, *marginal*, *irrelevant* and *cannot judge*. The latter is used for instance when the scanned image of a book is unreadable. For the Best Book task we are mainly interested in books labelled *good* or *excellent*. In Figure 1 we see the success rates

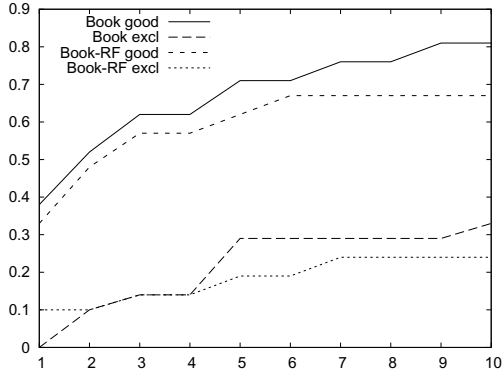


Fig. 1. Success rate over ranks 1 to 10 in the 2010 Best Books task

Table 7. Mapping of the Prove It relevance labels to graded relevance values

Label	Official Extra	
	Official	Extra
Irrelevant (0)	0	0
Relevant (1)	1	1
Refute (2) or Confirm (3)	2	10

over the top 10 for retrieving at least one book labelled *good* or higher, and for retrieving at least one *excellent* book. We see that the Book run has a success rate for books labelled *good* or higher of almost 0.4 at rank 1 going up to 0.81 at rank 10. The Book-RF runs has a slightly lower success rate. For the books labelled *excellent*, the success rates are much lower. The Book run returns an excellent book in the top 10 for 33% of the topics, the Book-RF run for 24% of the topics.

3.3 Prove It Results

For the Prove It task, the goal is to return individual book pages that either confirm or refute a factual statement. The same set of topics is used as for the Best Book task, but here, participants could use the query field, as well as the factual statement. For our runs we only use the query field.

The assessed book pages could be assessed as either *irrelevant*, *relevant*, *refute* or *confirm*. Pages are labelled relevant when they discuss the topic of the statement, but neither refute or confirm that statement. These labels can be mapped to relevance values in multiple ways. We show results for the official mapping, and a mapping in which the refute/confirm pages are weighted extra. These mappings to graded relevance values are shown in Table 7. For the Extra mappings, the refute/confirm pages weight 10 times as much as pages labelled relevant.

The nDCG@10 measure uses the graded relevance scores, and we regard the nDCG@10 as the official measure. The results are given in Table 8. The 1-page index gives better results than the 5-page index in terms of early precision. For

Table 8. Evaluation results for the INEX 2010 Prove It task, nDCG@10 with official weights 0-1-2 (4th column) and Extra weights 0-1-10 (5th column)

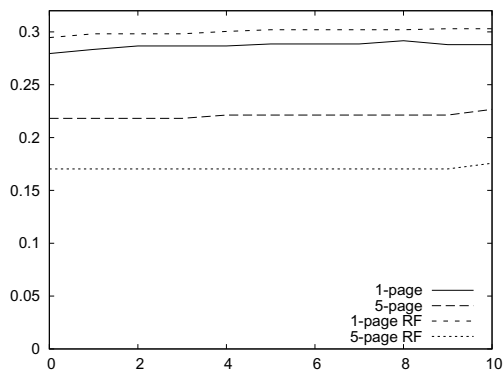
Run ID	MAP	P@10	nDCG@10	
			Official	Extra
1-page	0.1216	0.3238	0.2795	0.2338
1-page RF	0.1521	0.3524	0.2946	0.2322
5-page	0.1209	0.2619	0.2182	0.1714
5-page RF	0.1163	0.2143	0.1703	0.1371

the 1-page index, feedback improves precision, while for the 5-page index, feedback hurts performance. The top single page results are probably more focussed on the topic, so the expansion terms are also more related to the topic. The 5-page units might have only one or two pages on the topic, with the surrounding pages slowly drifting to other topics, which causes further topic drift through pseudo relevance feedback.

If we put extra weight on the confirm/refute pages, the nDCG@10 scores of all four runs drop. However, they drop slightly more for the 1-page RF run than for the other runs. Although feedback improves precision of the 1-page run, it apparently finds more relevant pages, but fewer confirm/refute pages. This could again be caused by topic drift introduced through pseudo relevance feedback.

Longer units seem to provide no benefit to finding confirm/refute pages or relevant pages. At least not when treating each page within the 5-page units as individual results. Of course, one could still score a whole 5-page unit if any one of the 5 pages is relevant or confirms/refutes the factual statement. However, the relevance judgements treat each page as a separate document, so scoring larger units based on these page level judgements would give systems returning larger units an unfair advantage, as they can return more pages within the same number of ranks as systems returning only individual pages.

We have also looked at *near-misses*, that is, retrieved pages that are not relevant themselves, but are next to a relevant pages in the book. Because pages typically

**Fig. 2.** Impact of near misses on nDCG@10 performance in the 2010 Prove It task

do not coincide with the logical of a book—section and paragraphs can start or end anywhere on the page and be split over multiple pages—a topic might be discussed over multiple pages. Although the information requested by a user might be on one page, the page preceding or succeeding it might still be closely related, and contain clues for the reader that the actual relevant information is nearby.

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, we found that pseudo relevance feedback is effective in combination with an index of individual pages as units. With larger units, feedback hurts precision. Given that the topics consist of specific factual statements, feedback only works when the top ranked results are highly focused on the topic of these statements. Although larger units might make it easier to find relevant material—early precision of the book-level runs is higher than that of the page-level runs—they also provide more off-topic text with which feedback terms introduce topic drift.

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] Fachry, K.N., Kamps, J., Koolen, M., Zhang, J.: Using and detecting links in wikipedia. In: Fuhr, N., Kamps, J., Lalmas, M., Trotman, A. (eds.) INEX 2007. LNCS, vol. 4862, pp. 388–403. Springer, Heidelberg (2008)
- [2] Hiemstra, D., Robertson, S., Zaragoza, H.: Parsimonious language models for information retrieval. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 178–185. ACM Press, New York (2004)
- [3] Kamps, J., Koolen, M.: The impact of document level ranking on focused retrieval. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 140–151. Springer, Heidelberg (2009)
- [4] Kamps, J., Koolen, M.: On the relation between relevant passages and XML document structure. In: Trotman, A., Geva, S., Kamps, J. (eds.) SIGIR 2007 Workshop on Focused Retrieval, pp. 28–32. University of Otago, Dunedin (2007)

- [5] Kamps, J., Koolen, M., Sigurbjörnsson, B.: Filtering and clustering XML retrieval results. In: Fuhr, N., Lalmas, M., Trotman, A. (eds.) INEX 2006. LNCS, vol. 4518, pp. 121–136. Springer, Heidelberg (2007)
- [6] Kamps, J., Koolen, M., Lalmas, M.: Locating relevant text within XML documents. In: Proceedings of the 31th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 847–849. ACM Press, New York (2008)
- [7] Kamps, J., Geva, S., Trotman, A., Woodley, A., Koolen, M.: Overview of the INEX 2008 ad hoc track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 1–28. Springer, Heidelberg (2009)
- [8] Kaptein, R., Koolen, M., Kamps, J.: 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 (2009)
- [9] Kazai, G., Milic-Frayling, N., Costello, J.: Towards methods for the collective gathering and quality control of relevance assessments. In: SIGIR 2009: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 452–459. ACM, New York (2009), doi: <http://doi.acm.org/10.1145/1571941.1572019> ISBN 978-1-60558-483-6
- [10] Koolen, M., Kamps, J.: The importance of anchor-text for ad hoc search revisited. In: Chen, H.-H., Efthimiadis, E.N., Savoy, J., Crestani, F., Marchand-Maillet, S. (eds.) Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 122–129. ACM Press, New York (2010)
- [11] Koolen, M., Kaptein, R., Kamps, J.: Focused search in books and wikipedia: Categories, links and relevance feedback. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 273–291. Springer, Heidelberg (2010)
- [12] Schenkel, R., Suchanek, F., Kasneci, G.: Yawn: A semantically annotated wikipedia xml corpus (2007), <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.140.5501>
- [13] Sigurbjörnsson, B., Kamps, J.: The effect of structured queries and selective indexing on XML retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 104–118. Springer, Heidelberg (2006)
- [14] Sigurbjörnsson, B., Kamps, J., de Rijke, M.: An Element-Based Approach to XML Retrieval. In: INEX 2003 Workshop Proceedings, pp. 19–26 (2004)
- [15] Sigurbjörnsson, B., Kamps, J., de Rijke, M.: Mixture models, overlap, and structural hints in XML element retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 196–210. Springer, Heidelberg (2005)
- [16] Strohman, T., Metzler, D., Turtle, H., Croft, W.B.: Indri: a language-model based search engine for complex queries. In: Proceedings of the International Conference on Intelligent Analysis (2005)
- [17] Vercoustre, A.-M., Pehcevski, J., Thom, J.A.: Using Wikipedia categories and links in entity ranking. In: Focused Access to XML Documents, pp. 321–335 (2007)

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. 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. We tested a number of combinations for this approach, and were able to obtain the top score for the “Best Books” task.

1 Introduction

In our 2009 work on the INEX Book Track, 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. 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, and none of our approaches were as effective as those reported by some other groups. 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 referring 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 $P(R | Q, D)$ is calculated and the documents or components are presented to the user ranked in order of decreasing values of that probability. To avoid invalid probability values, the usual calculation of $P(R | Q, D)$ uses the “log odds” of relevance given a set of S statistics derived from the query and database, such that:

$$\begin{aligned} \log O(R|C, Q) &= \log \frac{p(R|C, Q)}{1 - p(R|C, Q)} = \log \frac{p(R|C, Q)}{p(\bar{R}|C, Q)} \\ &= c_0 + c_1 * \frac{1}{\sqrt{|Q_c|} + 1} \sum_{i=1}^{|Q_c|} \frac{qt f_i}{ql + 35} \\ &+ c_2 * \frac{1}{\sqrt{|Q_c|} + 1} \sum_{i=1}^{|Q_c|} \log \frac{t f_i}{cl + 80} \\ &- c_3 * \frac{1}{\sqrt{|Q_c|} + 1} \sum_{i=1}^{|Q_c|} \log \frac{ct f_i}{N_t} \\ &+ c_4 * |Q_c| \end{aligned}$$

where C denotes a document component and Q a query, R is a relevance variable, and

$p(R|C, Q)$ is the probability that document component C is relevant to query Q ,

$p(\overline{R}|C, Q)$ 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 i th matching term,

tf_i is the within-document frequency of the i th matching term,

ctf_i is the occurrence frequency in a collection of the i th matching term,

ql is query length (i.e., number of terms in a query like $|Q|$ for non-feedback situations),

cl is component length (i.e., number of terms in a component), and

N_t is collection length (i.e., number of terms in a test collection).

c_k are the k coefficients obtained through the regression analysis.

Assuming that stopwords are removed during index creation, then ql , cl , and N_t are the query length, document length, and collection length, respectively. If the query terms are re-weighted (in feedback, for example), then qtf_i is no longer the original term frequency, but the new weight, and ql is the sum of the new weight values for the query terms. Note that, unlike the document and collection lengths, query length is the relative frequency without first taking the log over the matching terms.

The coefficients were determined by fitting the logistic regression model specified in $\log O(R|C, Q)$ to TREC training data using a statistical software package. The coefficients, c_k , used for our official runs are the same as those described by Chen [1]. These were: $c_0 = -3.51$, $c_1 = 37.4$, $c_2 = 0.330$, $c_3 = 0.1937$ and $c_4 = 0.0929$. Further details on the TREC2 version of the Logistic Regression algorithm may be found in Cooper et al. [4].

2.2 Blind Relevance Feedback

It is well known that blind (also called pseudo) relevance feedback can substantially improve retrieval effectiveness in tasks such as TREC and CLEF. (See for example the papers of the groups who participated in the Ad Hoc tasks in TREC-7 (Voorhees and Harman 1998) [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 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:

$$CLM_c = \sum_{i=1}^{|Q_c|} tf_i \quad (2)$$

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.

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 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

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

Table 4. 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 identifiers are included in each object, and these identifiers are simple extensions of the document (book) level identifier, we were able to use the page-level identifier to determine where in a given book-level document a particular page or paragraph occurs, and generate an appropriate XPath for it.

Indexes were created to allow searching of full page contents, and component indexes for the full content of each of individual paragraphs on a page. Because of the physical layout based structure used by the Books collection, paragraphs split across pages are marked up (and therefore indexed) as two paragraphs. Indexes were also created to permit searching by object id, allowing search for specific individual pages, or ranges of pages.

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

4 INEX 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 than 16 million pages, response time was very good for the basic search operations, but slowed dramatically whenever data from records was needed.

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

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

Table 6. Evaluation results for the INEX 2010 Best Books task

Run ID	MAP	P@10	NDCG@10
p14-BOOKS2010_CLM_PAGE_SUM.trec	0.1507	0.2714	0.2017
p14-BOOKS2010_CLM_PAGE_SUM_300.trec	0.1640	0.2810	0.2156
p14-BOOKS2010_T2FB_BASE_BST.trec	0.3981	0.5048	0.5456
p14-BOOKS2010_T2_PAGE_SUM_300.trec	0.5050	0.6667	0.6579
p6-inex10.book.fb.10.50.trec	0.3087	0.4286	0.3869
p6-inex10.book.trec	0.3286	0.4429	0.4151
p98-baseline_1.trec	0.4374	0.5810	0.5764
p98-baseline_1_wikifact.trec	0.4565	0.5905	0.5960
p98-baseline_2.trec	0.4806	0.6143	0.6302
p98-baseline_2_wikifact.trec	0.5044	0.6381	0.6500
p98-fact_query_10_wikibests.trec	0.4328	0.5714	0.5638
p98-fact_query_entropy.trec	0.4250	0.5476	0.5442
p98-fact_query_tfidfwiki.trec	0.3442	0.4667	0.4677
p98-fact_query_tfwiki.trec	0.4706	0.5571	0.5919
p98-fact_stanford_deps.trec	0.4573	0.5857	0.5976

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.

Table 6 shows the official results for the Best Books task in the book retrieval tasks. The first four lines of Table 6 show the results for our official submitted runs described in Table 5 (as Mean Average Precision, Precision at 10, and Normalized Discounted Cumulative Gain at 10, the official measures used for the task). As these results show, the simplest methods are definitely not the best methods in the best book retrieval task. Our runs using simple coordination level matching both showed significantly worse results (note that the full result data for each participant was not available, and so we could not calculate true statistical significance. They certainly appear to be considerably worse than any of the other submitted runs). On the other hand, our baseline run (T2FB_BASE_BST), appears in the middle range of the score distribution, and our test run using the TREC 2 algorithm at the page level and simple summation of scores obtained the best results of any official run.

5 Conclusions and Future Directions

The results of the Books track were only recently made available to participants, but a few observations can be made about the runs. Our belief that the simple CLM on pages might outperform the TREC2 approach on pages (as suggested in our notebook paper) was wildly wrong, given how the CLM approaches were outperformed by every other approach tried this year. Perhaps this can be seen as an object lesson of the unreliability of “eyeballing” evaluation results in the absence of actual result data. We have known for some time that the base TREC 2 algorithm usually provides fairly high precision (and usually good recall too when combined with the blind feedback approach described above). This appears to be the main factor for the success in the Best Books task, high precision search at the page level, with book ranking scores based on a summation of page scores for the books with highly ranked pages.

Of course, it is quite reasonable that the TREC2 algorithm outperforms the CLM approach. The CLM ranking value, as noted above is purely concerned with term occurrences in the text, while the TREC2 logistic regression algorithm attempts to estimate the probability that a document component is relevant for a given query. While a CLM value can be inflated by multiple occurrences of any given term (even very common terms), the TREC2 approach averages the contributions of the various terms, and therefore tends to favor pages with more query terms with higher IDF values.

As noted by one reviewer, there is a question of why page-level search seems to capture more appropriate information than document (or book-level) search. I would suggest that this may be more an artifact of the types of questions/topics than necessarily a characteristic of book searching. Most of the topics are actually framed as “factoid” types of questions, most of which can be answered in one or a few pages, and which are unlikely to have full book-length treatments (although some of the particular people, places, or events making up parts of the questions might well have full books, which contain the pages that could answer the questions).

Now that the relevance data for this task is available, we plan to test other approaches to merging page-level results, as well as other basic ranking algorithms for the page search component, including, naturally, the use of blind feedback at the page level. The reason that this latter was not included in the official runs was that we underestimated the time it would take to build the vector files needed for each page, and ran out of time to make the official deadline.

References

1. Chen, A.: Multilingual information retrieval using english and chinese queries. In: Peters, C., Braschler, M., Gonzalo, J., Kluck, M. (eds.) CLEF 2001. LNCS, vol. 2406, pp. 44–58. Springer, Heidelberg (2002)
2. Chen, A.: Cross-Language Retrieval Experiments at CLEF 2002. In: Peters, C., Braschler, M., Gonzalo, J. (eds.) CLEF 2002. LNCS, vol. 2785, pp. 28–48. Springer, Heidelberg (2003)

3. Chen, A., Gey, F.C.: Multilingual information retrieval using machine translation, relevance feedback and compounding. *Information Retrieval* 7, 149–182 (2004)
4. Cooper, W.S., Chen, A., Gey, F.C.: Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression. In: *Text Retrieval Conference (TREC-2)*, pp. 57–66 (1994)
5. Cooper, W.S., Gey, F.C., Dabney, D.P.: Probabilistic retrieval based on staged logistic regression. In: *15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Copenhagen, Denmark, June 21–24, pp. 198–210. ACM, New York (1992)
6. Harman, D.: Relevance feedback and other query modification techniques. In: Frakes, W., Baeza-Yates, R. (eds.) *Information Retrieval: Data Structures & Algorithms*, pp. 241–263. Prentice-Hall, Englewood Cliffs (1992)
7. Larson, R.R.: A logistic regression approach to distributed IR. In: *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, August 11–15, pp. 399–400. ACM, New York (2002)
8. Larson, R.R.: A fusion approach to XML structured document retrieval. *Information Retrieval* 8, 601–629 (2005)
9. Larson, R.R.: Probabilistic retrieval, component fusion and blind feedback for XML retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) *INEX 2005*. LNCS, vol. 3977, pp. 225–239. Springer, Heidelberg (2006)
10. Larson, R.R.: Ranking and fusion approaches for XML book retrieval. In: Geva, S., Kamps, J., Trotman, A. (eds.) *INEX 2009*. LNCS, vol. 6203, pp. 179–189. Springer, Heidelberg (2010)
11. Voorhees, E., Harman, D. (eds.): *The Seventh Text Retrieval Conference (TREC-7)*. NIST (1998)
12. Voorhees, E., Harman, D. (eds.): *The Eighth Text Retrieval Conference (TREC-8)*. NIST (1999)

OUC's Participation in the 2010 INEX Book Track

Michael Preminger and Ragnar Nordlie

Oslo University College

Abstract. In this article we describe the Oslo University College's participation in the INEX 2010 Book track. 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 new knowledge, namely the establishment (or refutation) 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 refutation 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. We attempted to divide the pages in the collection into categories of how confirmative they are, and indexed them individually (each page comprising a document). 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. It may be argued that a richer set of confirmation indicators could be applied. Our claim is that the selected words should function as style indicators, not as content indicators (the content catered for by the topic, i.e. the factual claim under scrutiny), and were therefore sufficient. A larger collection could incur noise.

These tags then facilitated weighting pages differently (based on their proportion of confirmatory words) when retrieving candidates of confirming pages. Retrieval was performed using the Indri language model in the default mode, weighting the confirmatory pages differently, as indicated above. As the primary aim here is to try and compare traditional retrieval with "prove it", there was no particular reason to divert from the default.

The 2010 tasks have been featuring a relatively large number of topics (83). As a new method was employed for collecting relevance assessments, only 21 of these queries were ready for evaluation at deadline time, and it is these queries that are used for retrieval.

3 Runs and Results

Based on the index that we were constructing, we weighted relevant pages on two levels: pages that featured 1 percent or more confirmatory words, and pages that featured 3 percents or more confirmatory words (the latter including the former) were weighted double, quintuple (5x) and decuple (10x) the baseline. Our baseline was normal, non-weighted retrieval, as we would do for finding relevant pages. We were using the indri combine operation with no changes to the default setting (regarding smoothing, a.s.o).

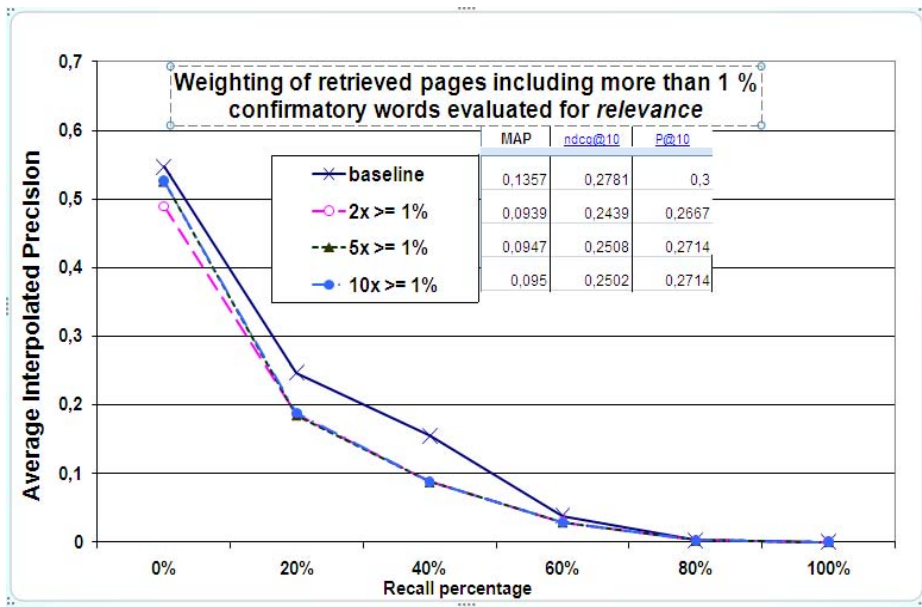
The analysis was carried out twice: once against the entire official qrel file (meaning that all assessed book pages judged either confirming/refuting – or merely relevant to – the statement-query are taken to be relevant (Figure 1)). The second analysis was done against a filtered version of the official qrel file, featuring only the pages assessed as confirming/refuting (Figure 2).

The purpose was to see if the rate of confirmatory words can be used as a "prove it" indicator, given that the relevance assessments properly reveal pages that confirm the factual statement. Weighting retrieved pages that feature 1 percent or more confirmatory words does not seem to outperform the baseline at any weighting level, at any region of the precision recall curve (Subfigures 1(a) and 2(a)). The reason for that may be that quite many pages belong to this category. The weighting thus seems to hit somewhat randomly. An occurrence rate of confirmatory words of one percent seems not to discriminate "proving" pages.

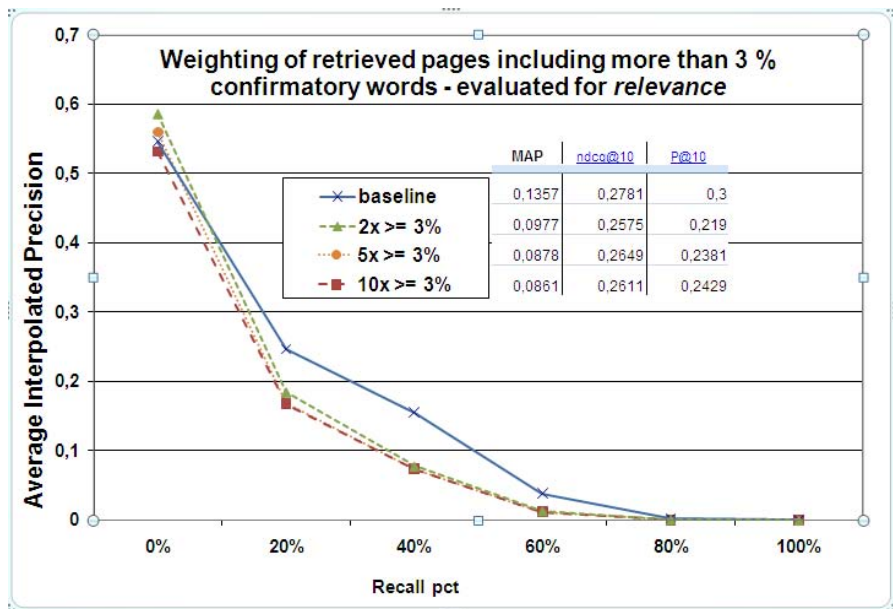
The results are more interesting when restricting the weighting to pages that feature 3 percents or more confirmatory words. Here the results are different at the low recall and the high recall regions. Directing our attention to the low recall region first, we see in Figure 1(b) that both doubling the weight and, to a lesser extent quintupling it, slightly outperform the the baseline. The effect is a bit clearer when evaluating by pages assessed as confirming (Figure 2(b)). Here also decupling the weight given to pages with 3 percent or more of confirmatory words slightly outperforms the baseline.

No treatment seems to outperform the baseline in the higher recall regions. Subject, of course, to a more thorough scrutiny of the result, this could indicate that collecting many books that prove a statement is not likely to be better supported by this approach than by traditional retrieval, whereas only finding very few such books (early hits) might benefit from it. The reason for that may be approached by looking at single relevant pages retrieved at the low and high recall regions. This kind of treatment was beyond the scope of the present paper. The value of pursuing it may be limited in light of the overall results.

Looking at the "prove it" task in terms of traditional retrieval, the temporary conclusion would be that the treatment experimented with here may be in the right direction, and further pursuit of it has some potential of good retrieval, particularly if it is the low recall region that is important (early hits). If the purpose is collecting as many books as possible as evidence for a claim then the approach does not seem as promising.

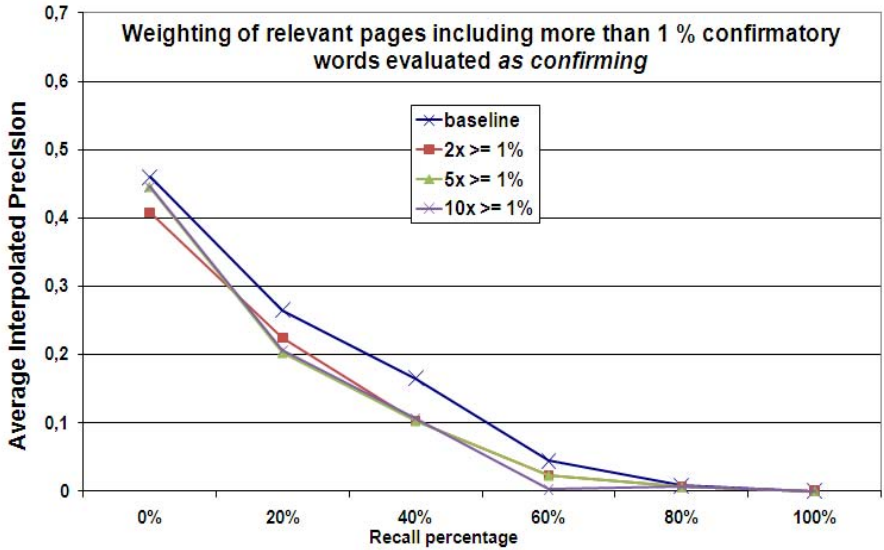


(a) Weighting relevant pages with 1 percent or more confirmatory words

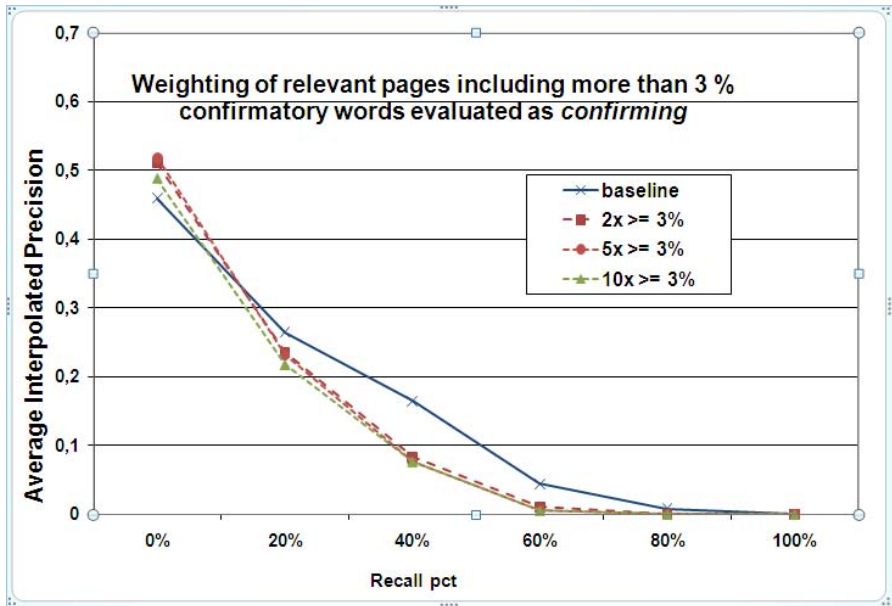


(b) Weighting relevant pages with 3 percent or more confirmatory words

Fig. 1. Precision-recall curves for detecting relevant pages. Baseline marked by solid lines in both subfigures.



(a) Weighting relevant pages with 1 percent or more confirmatory words



(b) Weighting relevant pages with 3 percent or more confirmatory words

Fig. 2. Precision-recall curves for detecting confirming (proving) pages. Baseline marked by solid lines in both subfigures.

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 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. The notion of relevance is a well known challenge in IR [2]. We suspect that the "prove it" notion is by no means simpler. 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* (topic 2010003) 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 refutation of a statement is arguably a totally different activity than detecting its confirmation. This poses challenges not only to mere retrieval, but also to its evaluation, at all levels.

Even though such activities may be developed and refined using techniques from e.g. Question Answering [3], we suspect that employing semantics-aware retrieval [4,5], which is closely connected to the development of the Semantic Web [6] 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) [7] 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., Doucet, A., Koolen, M., Landoni, M.: Overview of the INEX 2009 book track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 145–159. Springer, Heidelberg (2010)
2. Mizzaro, S.: Relevance: The whole history. *Journal of the American Society of Information Science* 48(9), 810–832 (1997)
3. Voorhees, E.M.: The trec question answering track. *Natural Language Engineering* 7, 361–378 (2001)
4. Finin, T., Mayfield, J., 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),
5. Mayfield, J., Finin, T.: Information retrieval on the semantic web: Integrating inference and retrieval. In: SIGIR Workshop on the Semantic Web, Toronto,
6. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. In: *Scientific American* (2001)
7. Zöllner-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 Trotman¹ and Qiuyue Wang²

¹ Department of Computer Science
University of Otago
Dunedin
New Zealand

² 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 that the Wikipedia will include a whole document result to almost any ad hoc query.

In the Data Centric Track we ask: Can whole document retrieval approaches outperform focused retrieval on highly structured document collection?

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 information¹. 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 system². 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 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 actors³, 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 Federal do Amazonas, and Universitat Pompeu Fabra). From these a total of 28 topics were selected based on uniqueness, preciseness, and being correctly formed. An example topic (2010001) is given in Fig. 1:

¹ However, as it is unclear how to evaluate this the standard INEX metrics were eventually used.

² <http://www.jmdb.de/>

³ 21 of the actor files were empty and removed.


```

<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
AttributeName ::=      '@' AttributeName
Index         ::=      '[' integer ']'

```

For Example the path `/article[1]/body[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 109UniXRun1 /article[1]/body[1]/sec[1]
1 Q0 9996 1 0.9999 109UniXRun1 /article[1]/body[1]/sec[2]/p[1]
1 Q0 9888 1 0.9999 109UniXRun1 /article[1]/body[1]/sec[3]
1 Q0 9997 2 0.9998 109UniXRun1 /article[1]/body[1]/sec[2]
1 Q0 9989 3 0.9997 109UniXRun1 /article[1]/body[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 Retrieval⁴. 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).

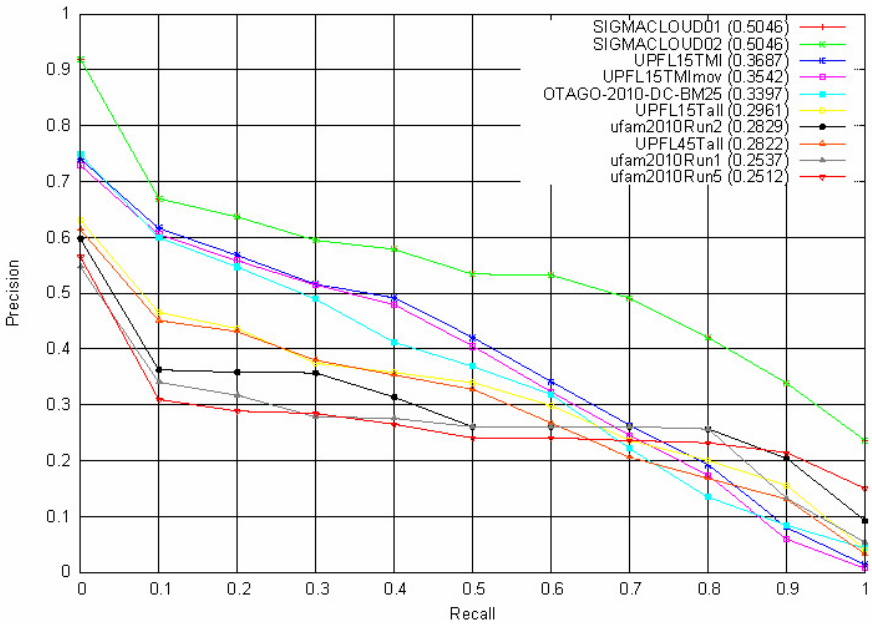


Fig. 2. Best runs measured with MAP

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

⁴ See the ad hoc track overview paper (in this volume) for details on the metrics.

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 data centric, most runs did not use structure in ranking.

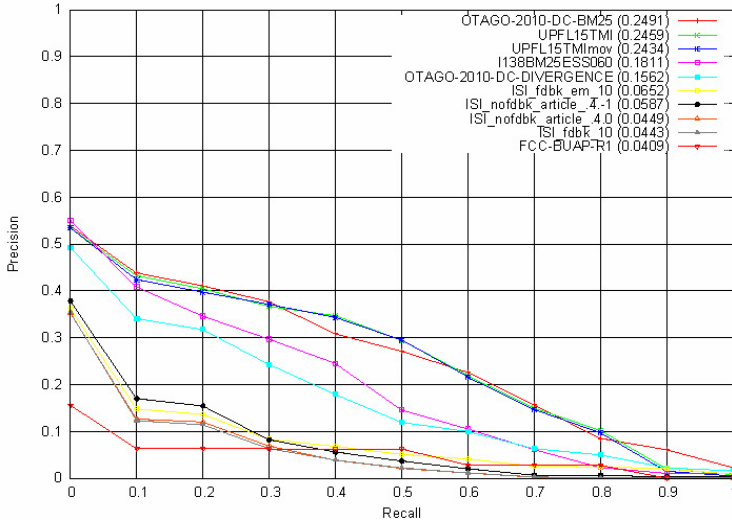


Fig. 3. Best runs measured with MAGP T2I(300)

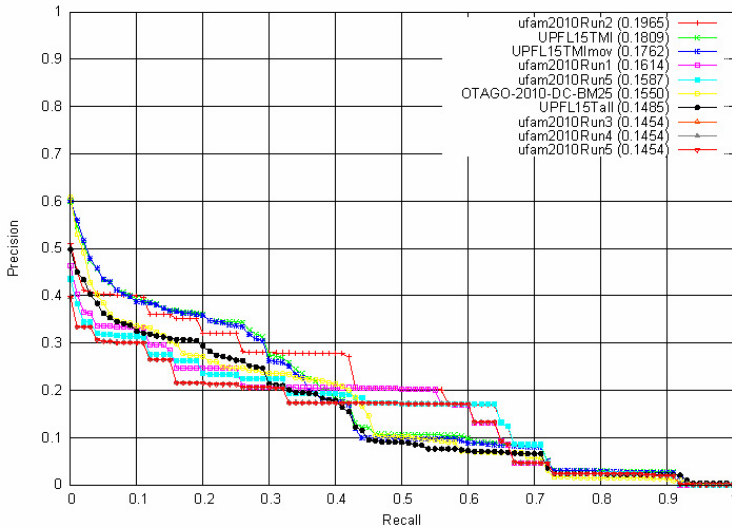


Fig. 4. Best runs measured with MAiP

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 ambitious 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.

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. Qiuyue Wang is supported by the 863 High Tech. Project of China under Grant No. 2009AA01Z149.

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 genre (#PCDATA)>
<!ELEMENT tagline (#PCDATA)>
<!ELEMENT plot (#PCDATA)>
<!ELEMENT keywords (keyword+)>
<!ELEMENT keyword (#PCDATA)>
```

```
<!ELEMENT cast (actors?, composers?, editors?, 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 editors (editor+)>
<!ELEMENT editor (#PCDATA)>
<!ELEMENT cinematographers (cinematographer+)>
<!ELEMENT cinematographer (#PCDATA)>
<!ELEMENT producers (producer+)>
<!ELEMENT producer (#PCDATA)>
<!ELEMENT production_designers (production_designer+)>
<!ELEMENT production_designer (#PCDATA)>
<!ELEMENT costume_designers (costume_designer+)>
<!ELEMENT costume_designer (#PCDATA)>
<!ELEMENT miscellaneous (person+)>
<!ELEMENT person (#PCDATA)>
```

```
<!ELEMENT additional_details
(aliaes?,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, #PCDATA?)>
<!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_name (#PCDATA)>
<!ELEMENT birth_date (#PCDATA)>
<!ELEMENT death_date (#PCDATA)>
<!ELEMENT height (#PCDATA)>
<!ELEMENT spouse (#PCDATA)>
<!ELEMENT trademark (#PCDATA)>
<!ELEMENT biographies (biography+)>
<!ELEMENT biography (#PCDATA, by)>
<!ELEMENT by (#PCDATA)>
<!ELEMENT nicknames (name+)>
<!ELEMENT trivias (trivia+)>
<!ELEMENT trivia (#PCDATA)>
<!ELEMENT personal_quotes (quote+)>
<!ELEMENT quote (#PCDATA)>
<!ELEMENT where_are_they_now (where+)>
<!ELEMENT where (#PCDATA)>
<!ELEMENT alternate_names (name+)>
<!ELEMENT salaries (salary+)>
<!ELEMENT salary (#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 interview (#PCDATA)>
<!ELEMENT articles (article+)>
<!ELEMENT article (#PCDATA)>
<!ELEMENT biography_prints (print+)>
<!ELEMENT print (#PCDATA)>
<!ELEMENT biographical_movies (biographical_movie+)>
<!ELEMENT biographical_movie (#PCDATA)>
<!ELEMENT portrayed_ins (portrayed_in+)>
<!ELEMENT portrayed_in (#PCDATA)>
<!ELEMENT magazine_cover_photos (magazine+)>
<!ELEMENT magazine (#PCDATA)>
<!ELEMENT pictorials (pictorial+)>
<!ELEMENT pictorial (#PCDATA)>
```

DCU and ISI@INEX 2010: Adhoc and Data-Centric Tracks

Debasis Ganguly¹, Johannes Leveling¹, Gareth J.F. Jones¹
Sauparna Palchowdhury², Sukomal Pal², and Mandar Mitra²

¹ CNGL, School of Computing, Dublin City University, Dublin, Ireland

² CVPR Unit, Indian Statistical Institute, Kolkata, India

{dganguly,jleveling,gjones}@computing.dcu.ie,

sauparna.palchowdhury@gmail.com, {sukomal_r,mandar}@isical.ac.in

Abstract. We describe the participation of Dublin City University (DCU) and Indian Statistical Institute (ISI) in INEX 2010 for the Adhoc and Data Centric tracks. The main contributions of this paper are: i) a simplified version of Hierarchical Language Model (HLM), which involves scoring XML elements with a combined probability of generating the given query from itself and the top level article node, is shown to outperform the baselines of LM and VSM scoring of XML elements; ii) the Expectation Maximization (EM) feedback in LM is shown to be the most effective on the domain specific collection of IMDB; iii) automated removal of sentences indicating aspects of irrelevance from the narratives of INEX ad hoc topics is shown to improve retrieval effectiveness.

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” task which has been designed particularly for displaying results on a mobile device with limited screen size. The Adhoc track tasks comprises of the following tasks: a) the “Restricted Focused” task which asks systems to return a ranked list of elements or passages to the user restricted to at most 1000 characters per topic; b) the (un)restricted “Relevant in Context” task which asks systems to return relevant elements or passages grouped by article, a limit of at most 500 characters being imposed on the restricted version; and c) the “Efficiency” task which expects systems to return thorough article level runs. In addition, we participated in the new “Data Centric track” which is similar to ad hoc retrieval of elements or passages on a domain specific collection of IMDB movie pages. In INEX 2010 we submitted 9 ad hoc focused runs, (3 for each Focused task) and 3 Thorough runs for the Ad Hoc track. In addition we submitted 10 runs for the Data Centric task.

The remainder of this 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 Section 4 we propose a simplified version of HLM for XML retrieval, which involves scoring an XML element with a linear combination of the probability of generating the given query from itself and its parent article. Using INEX 2009 topic set for training and 2010 topic set for testing, we show that it outperforms the standard LM and VSM method of scoring of XML elements. Section 5 explores the effectiveness of Blind Relevance Feedback (BRF) on the domain specific collection of movie database. Section 6 describes the work of automatically identifying and removing negation from the verbose ‘narrative’ sections of the INEX queries, and observing how retrieval effectiveness improves. The entire topic (*TDN*) is treated as the query in this experiment. Section 7 concludes the paper with directions for future work.

2 Document Retrieval

2.1 Preprocessing

Similar to INEX 2009, for extracting useful parts of documents, we shortlisted about thirty tags that contain useful information: $\langle p \rangle$, $\langle pl \rangle$, $\langle st \rangle$, $\langle section \rangle$ etc. [2]. 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 were indexed using single terms and a controlled vocabulary (or pre-defined set) of statistical phrases employing the SMART [1] system. Words in the standard stop-word list included within SMART were removed from both documents and queries. The default stemmer implementation of SMART which is a variation of the Lovin’s stemmer [3] was used. Frequently occurring word bi-grams (loosely referred to as phrases) were also used as indexing units. We used the N-gram Statistics Package [2] (NSP) 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 is 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 a linear combination of estimated probabilities $P(q|d)$ of generating a query q from the document d and $P(t_i)$ of generating the term from the collection. The

¹ <ftp://ftp.cs.cornell.edu/pub/smart/>

² <http://www.d.umn.edu/~tpederse/nsp.html>

document is modelled to choose $q = \{t_1, t_2 \dots t_n\}$ as a sequence of independent words as proposed by Hiemstra [4].

$$P(q|d) = P(d) \log P(d) \prod_{i=1}^n \lambda_i P(t_i|d) + (1 - \lambda_i) P(t_i) \quad (1)$$

$$\log P(q|d) = \log P(d) + \sum_{i=1}^n \log\left(1 + \frac{\lambda_i}{1 - \lambda_i} \frac{P(t_i|d)}{P(t_i)}\right) \quad (2)$$

$P(d)$ is the prior probability of relevance of a document d and it is typically assumed that longer documents have higher probability of relevance. The term weighting equation can be derived from Equation 1 by dividing it with $(1 - \lambda_i)P(t_i)$ and taking logarithm on both sides to convert the product to summation. This transformation also ensures that the computed similarities between documents and a given query are always positive. We index each query vector \mathbf{q} as $q_k = tf(t_k)$ and each document vector \mathbf{d} as $d_k = \log\left(1 + \frac{P(t_k|d)}{P(t_k)} \frac{\lambda_k}{1 - \lambda_k}\right)$, so that the dot product $\mathbf{d} \cdot \mathbf{q}$ gives the likelihood of generating \mathbf{q} from \mathbf{d} and hence can be used as the similarity score to rank the documents.

3 XML 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 at most 1500 documents per topic) for each query using the LM retrieval method as described in the previous section 2.2. In the second pass, these documents were parsed using the libxml2 parser, and leaf nodes having textual content were identified. The total set of such leaf-level textual elements obtained from the 1500 top-ranked documents were then indexed and compared to the query as before to obtain the final list of 1500 retrieved elements. The preprocessing steps are similar to those as described in Section 2.1. The following section provides the details of our proposed method of scoring XML elements.

3.1 Simplified Hierarchical Language Model

Motivation. The objective of the focused task is to retrieve short relevant chunks of information. Scoring an XML element by its similarity with the query may retrieve short XML elements such as a small paragraph or sub-sections with a dense distribution of query terms in the top ranks. But there is an implicit risk associated with the retrieval of short high scoring elements namely that the text described in the short element might be a digression from the main topic of the parent article. Thus it is unlikely that the retrieved text from the XML element would serve any useful purpose to the searcher because it would not have the necessary context information to do so. As an example consider the artificial paragraph of text as shown in Fig. 1. Now let us imagine that a searcher’s query is “top-10 IMDB movies actors”. Let us also imagine that this paper has been converted into an XML formatted document and put into

We crawled the IMDB movie collection and categorized the crawled IMDB data into categories such as movies, actors etc. Movie reviews and ratings were also stored.

Fig. 1. An artificial paragraph of text to illustrate the implicit risk of out-of-context retrieval with response to a query “top-10 IMDB movies actors”

the INEX document collection. The contents of the above paragraph thus could be retrieved at a top rank because of its high similarity due to the presence of three matching query terms as highlighted with boxes. But the searcher was certainly not looking for a technical paragraph on indexing the IMDB collection. This shows that the retrieval model for XML elements needs to be extended to take into consideration the distribution of query terms in the parent articles in addition to the element itself. If the query terms are too sparse in the parent article, then it is more likely that the XML element itself is an out-of-context paragraph e.g. the highlighted words in Fig. 1 are very sparsely distributed throughout the rest of this document, which suggests that the shown artificial paragraph is not a good candidate for retrieval.

It is also desirable to retrieve an element with a few query terms if the other 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 specific facet of the user information need) whereas the rest of the article throws light on the other general facets hence providing the necessary context for the specific subtopic.

The Scoring Function. An extension of LM for Field Search named Field Language Model (FLM) 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. We propose to assign a score to the constituent element itself from the parent article evidence thus differing from FLM in the directionality of assignment of the scores. We use Equation 3 to score an XML element e .

$$P(q|e) = \log P(d) \prod_{i=1}^n \mu_i P(t_i|e) + \lambda_i P(t_i|d) + (1 - \lambda_i - \mu_i) P(t_i) \quad (3)$$

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 a non zero probability of generation is contributed to the product. Two levels of smoothing are employed in this case.

Ogilvie and Callan developed the general HLM which involves two way propagation of the LM element scores from the root to the individual leaf nodes and vice-versa [6]. Our model is much simpler in the sense that we use only

the current node and the top level article node for the score computation. We call this method Simplified Hierarchical Language Model (SHLM) because we restrict our smoothing choice to the root article element only in addition to the collection. The SHLM equation can be further simplified by using $\lambda_i = \lambda \wedge \mu_i = \mu \forall i = 1 \dots n$. It can be seen that Equation 3 addresses the motivational requirements as described in the previous section in the following ways:

- a) An element e_1 which has a query term t only in itself but not anywhere else in the top level article, scores lower than an element e_2 which has the term present both in itself and somewhere else in the article. Thus the model favours 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 with a few of the given query terms might be retrieved at a high rank if the missing terms are abundant in the article.

The experimental evaluations of SHLM for adhoc XML retrieval are provided in Section 4.2

4 Adhoc Track Experiments and Results

4.1 Thorough Task

We trained our system using the INEX 2009 topic set. All the initial article level runs were performed using LM retrieved as described in Equation 1. We assign $\lambda_i = \lambda \forall i = 1 \dots n$ and also assigned uniform prior probabilities to the documents. After a series of experiments by varying λ we found that best retrieval results are obtained with $\lambda = 0.4$ and henceforth we use this setting for all the article level LM runs reported in the paper. We officially submitted three article level runs containing 15, 150 and 1500 documents as *thorough* runs. On getting unexpectedly low retrieval precision, we conducted a posthoc analysis after the INEX results were officially out. We found that there was a bug in our retrieval control-flow where we used inverted lists constituted from the raw document vectors instead of the LM weighted ones. The results for the thorough runs, along with post-submission corrected versions are shown in Table 1.

Table 1. Official evaluation of the thorough runs

Run Id	# docs retrieved	Submitted		Corrected	
		MAP	MAiP	MAP	MAiP
ISI2010_thorough.1500	1500	0.0431	0.0846	0.1539	0.1750
ISI2010_thorough.150	150	0.0309	0.0826	0.1185	0.1421
ISI2010_thorough.15	15	0.0110	0.0714	0.0647	0.0930

4.2 Focused Task

We submitted 3 element level runs for the restricted focused task. The first two subsections report the training of SHLM on the INEX 2009 topics and the last section reports the official submissions under the restricted focused task.

Tuning SHLM. To test the effectiveness of SHLM (as outlined in Section 3.1) for element retrieval, we run SHLM with different combinations of λ and μ (simplifying Equation 3 by employing $\lambda_i = \lambda \wedge \mu_i = \mu \forall i = 1 \dots n$) on the INEX 2009 training topics. A revisit of Equation 3 suggests that a higher value of μ as compared to λ attaches too much importance on the presence of the query terms. 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. Our hypothesis is that a value of μ lower than λ ensures that we lay emphasis on not retrieving out-of-context small elements and we do expect better retrieval performance for the setting $\mu < \lambda$.

Another critical issue to explore in the model of Equation 3 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 Focused 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.

SHLM Results. As baseline we use standard LM scoring of the elements which is a special case of SHLM obtained by setting $\lambda = 0$. To verify our hypothesis that λ should be higher than μ , we ran two versions of SHLM one with $\lambda < \mu$ and the other $\mu > \lambda$. 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 - i) uniform, ii) proportional and iii) inversely proportional for the case $\mu < \lambda$. Table 2 provides empirical evidence to the hypothesis that elements when scored with contextual information from their parent article yield better retrieval results. 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.

Official Results. The “Restricted Focused” task required systems to return a ranked list of elements or passages restricted to at most 1000 characters per topic.

Table 2. SHLM for element retrieval for INEX 2009 topics

λ	μ	Element Prior probability	Retrieval Effectiveness			
			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

The evaluation metric used was P@500 characters. Since this metric favours retrieval runs with a high precision at low recall levels (recall is expected to be low when only 1000 characters are retrieved), we use the settings as reported in the third row of Table 2 i.e. with the settings $(\lambda, \mu) = (0.25, 0.15)$ with uniform element prior probability. We perform SHLM element retrieval on i) our best performing LM retrieved article level run ($\lambda = 0.4$) and ii) reference BM25 run provided by the INEX organizers. To show that SHLM performs better than the pivoted length normalized scoring which was our element level retrieval strategy for INEX 2009 [2], we also submitted a run scoring the elements by Equation 4.

$$normalization = 1 + \frac{slope}{(1 - slope)} \cdot \frac{\#unique\ terms}{pivot} \quad (4)$$

Table 3 shows that the corrected SHLM based element retrieval on the reference run yields the highest character-precision among our runs. We also see that the best character precision we achieved ranks third considering the list of official submissions (after LIP6-OWPCparentFo and DURF10SIXF).

SHLM clearly outperforms pivoted normalized element retrieval on the same document level run showing that given the same document level run, it is more effective than VSM based element retrieval. Table 3 also shows that SHLM outputs a better element level run for a better input article run as evident from

Table 3. Official evaluation of the Focused runs

Run Id	Methodology	P@500 chars submitted corrected	
		ISI2010_rfcs_ref	SHLM element retrieval on article level reference run
ISI2010_rfcs_fm	SHLM element retrieval ($\mu < \lambda$ and uniform prior of the elements) on article level LM run	0.2151	0.2841
ISI2010_rfcs_vsm	Pivoted normalized element retrieval on article level LM run	0.1289	0.2343
LIP6-OWPCparentFo (Best run)		0.4125	

the first and second rows (MAP of the reference run is higher than our LM based article run).

5 Data Centric Track Experiments and Results

We indexed the IMDB collection using SMART in a similar way as outlined in Section 2, the only difference being we did not use pre-extracted list of commonly occurring bigrams for constructing the phrase index. Our Data Centric official submissions also suffered from the same bug as had been the case with our Adhoc submissions. In this section we report a set of refined results after carrying out experiments with a corrected version.

Our approach in this track comprises of exploring the standard IR techniques on a domain specific collection like the movie database. Our initial experiments show that we get the optimal baseline MAP by using $\lambda = 0.6$. While performing feedback experiments we found that a selection of query expansion terms by the LM score [8] results in worse retrieval performance. Fig. 2a shows the effect of query expansion with different settings for R (the number of pseudo-relevant documents used) and t (the number of terms used for query expansion) on MAP. We implemented the EM feedback in SMART as proposed by Hiemstra [4] where each λ_i , associated with the query term t_i , is modified aiming to maximize the expectation of the LM term generation probability from the top ranked pseudo-relevant documents as shown in Equation 5.

$$\lambda_i^{p+1} = \frac{1}{R} \sum_{j=1}^R \frac{\lambda_i^p P(t_i|D_j)}{\lambda_i^{(p)} P(t_i|D_j) + (1 - \lambda_i^{(p)}) P(t_i)} \quad (5)$$

We use only one iteration of the feedback step i.e. we calculate λ_i^1 s from the initial $\lambda_i^0 = \lambda$ for each i . We also tried out applying EM to an expanded query with additional terms but we found out that it did not improve the MAP. The results as shown in Figure 2b reveal that EM performs better than the LM term based expansion as shown in Figure 2a. While doing a per-topic analysis of the document retrieval for the IMDB collection, we made the following interesting observation. Query 2010015 reads “May the force be with you” of which all are stopwords except the word *force*. As a result the obtained MAP for this query is 0. Interestingly, when the IMDB search is fed with the same query, it also returns documents with the term *force* in it but not with all the other words present which suggests that IMDB retrieval is also non-phrase based.

A characteristic of the IMDB collection is that the documents are grouped into categories such as *movies*, *actors* etc. To find if relevance is biased towards one of the categories, we computed the percentage of relevant documents in each of the categories from the article level manual assessments. We also computed the percentage of retrieved documents in each of category. Fig. 3 shows that relevance is heavily biased to the movie documents suggesting that the searchers mostly seek movie pages in response to a submitted query. For the retrieved set we find that movie pages are retrieved highest in number followed by actor pages and so on. The relative ranks of the number of hits for the relevant and

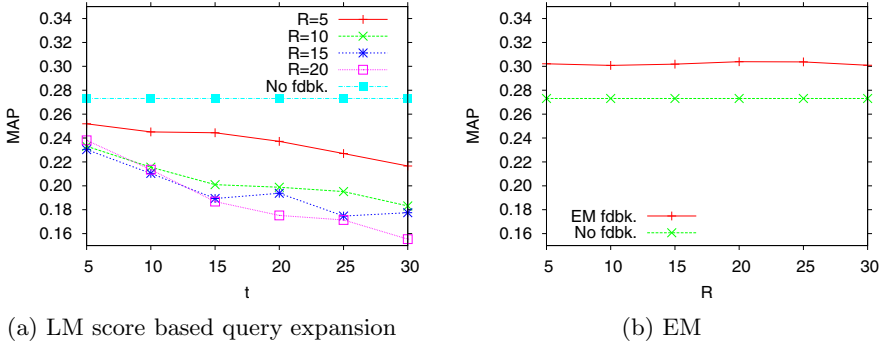


Fig. 2. Feedback effects on the IMDB collection

retrieved categories are almost the same with an exception for the categories of producers and directors where producer pages have the least number of hits in the relevant set whereas this category is not with least number of hits for the retrieved set. Although relative ranks are similar, there is a noticeable difference in the percentages between the two sets, especially in the categories movies and actors which suggests that adjusting the prior relevance $P(d)$ (Equation II) not according to the length of a document but according to its category could be a way to improve on the retrieval effectiveness. This would help to reduce the percentage gaps in the relevant and retrieved sets.

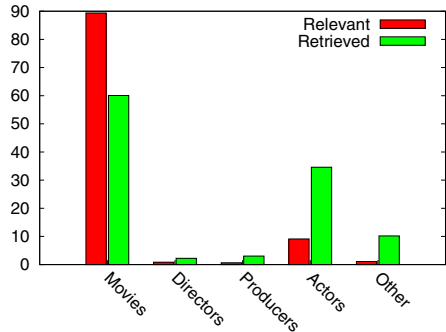


Fig. 3. Percentage of relevant and retrieved documents in the different categories

6 Using Negative Information Present in Verbose Queries

Motivation. It is easier for a user to formulate a complex information need in natural language rather than using terse keyword queries. The INEX topics have a detailed narrative (N) section that reinforces the information expressed in the title (T) and description (D) fields. Notably, in the narrative section, the topic creator specifies what he is not looking for.

One such INEX query is shown in Table 4. The emphasized sentence of N is the negative information. To support our claim that such negative information can introduce a query drift towards non-relevant documents, we report a comparison of retrieval results, using queries processed using manual and automatic methods. Results show that the modified queries, with negation removed, yield higher retrieval effectiveness.

Table 4. INEX 2009 Topic No. 80 with negative information

```

<topic id="2009080" ct_no="268">
<title>international game show formats</title>
<description>I want to know about all the game show formats that have adaptations
in different countries.</description>
<narrative> Any content describing game show formats with international adapta-
tions are relevant. National game shows and articles about the players and producers
are not interesting.
</narrative>
<topic>

```

Table 5. Classifier performance

Test set	Training set	# of training sentences	Accuracy
2008	2007	589	90.4%
2009	2008	679	89.1%
2010	2009	516	93.8%

Approach. Our unmodified set of queries is Q . From Q we create two new sets. The first one, P , consists of only those queries in Q which have negation, and, with these negative sentences or phrases removed. (P , stands for ‘positive’ in the context of this experiment). The second set, P_M (‘positive’, ‘machine’-processed set), is similar, but negation was now automatically identified using a Maximum Entropy Classifier (Stanford Classifier) [3](#), and removed. In [Table 6](#) the cardinalities of P and P_M differ because the classifier does not identify all the negative phrases and sentences with full accuracy. Some queries in Q , which have negation, and can be found in P , may not make it to P_M . We did retrieval runs using Q , P and P_M and noted the change in MAP (Refer to [9](#) for more details). The classifier performed well with accuracies around 90% ([Table 5](#)).

Retrieval Results. We used the SMART retrieval system to index the document collection using pivoted document length normalized VSM (Equation [4](#)), and the initial retrieval run was followed by a BRF run employing Rocchio feedback. For feedback we used the most frequent 20 terms and 5 phrases occurring in the top 20 pseudo-relevant documents setting $(\alpha, \beta, \gamma) = (4, 4, 2)$. [Table 6](#) that the positive sets show an improvement in all cases.

Of particular interest is the P_M results; the P results are included only to refer to the maximum relative gain in performance that is achievable. Although, as expected, the relative gains for the P_M set is lower as compared to the P set, the differences between the two relative gains are not too big, which shows that the automated process does well.

The Wilcoxon test verifies that the differences in the relative gains of Q and P are statistically significant, corroborating the fact that removal of negation

³ <http://nlp.stanford.edu/software/classifier.shtml>

Table 6. Comparison of performance of the manually processed (P) and automatically processed (P_M) positive query sets

Topic	Manually processed				Automatically processed			
	Set	$ P $	MAP_q	MAP_p	change	$ P_M $	MAP_q	MAP_{P_M}
INEX 2008	44	0.2706	0.2818	4.1%	31	0.2638	0.2748	4.2%
INEX 2009	36	0.2424	0.2561	5.7%	30	0.2573	0.2581	0.3%
INEX 2010	26	0.3025	0.3204	6.0%	20	0.2922	0.2983	2.1%

Table 7. MAP values for retrieval using increasing query size

INEX year	T	TD	Δ (%)	TDN	Δ' (%)	trend ($\Delta, \Delta' \geq 5\%$)
2008	0.2756	0.2815	2.14	0.2998	8.78	- \uparrow
2009	0.2613	0.2612	-0.03	0.2547	-2.52	- -
2010	0.2408	0.2406	-0.08	0.2641	9.67	- \uparrow

improves performance. Also, a test of significance between P and P_M shows that their difference is statistically insignificant showing that the automated process is as good as the manual one.

One must note that our baseline comprises retrieval runs over the set Q where the queries are of maximum length. The queries in P and P_M , are shorter. It is expected that the retrieval effectiveness will improve with an increase in query size for the bag-of-words approach. We needed this to be empirically verified to rule out the possibility of an improvement in the retrieval effectiveness due to query length shortening.

Three retrieval runs were done using T , TD and TDN . The results shown in Table 7 shows that there is a positive correlation between the MAP and query length. This eliminates the possibility of an improvement in MAP due to a negative correlation between query length and MAP for INEX topics. We also observe that the results for the 2009 set degrades across T , TD and TDN .

7 Conclusions and Future Work

Through our participation in the adhoc track of INEX 2010, we wanted to explore an extension of LM for IR which we call SHLM as a new element retrieval strategy. Trial experiments on INEX 2009 topics show that it outperforms the baseline LM element retrieval of the elements. Official restricted focused runs show that SHLM element retrieval outperforms the pivoted normalized VSM element retrieval. Also our corrected official focused runs ranks third among the submitted runs. The concept of SHLM can be extended to arbitrary passages by defining a series of fixed length window-subwindow structures.

For the data-centric track, we have shown that LM retrieval works well on a domain specific collection such as the movie database. We also show that query expansion using terms selected by LM scores do not improve retrieval

effectiveness whereas the EM feedback does well on this collection. We report a bias of relevance towards the movie and actor categories which suggests a possible future work of assigning higher prior probabilities of relevance for documents in these categories to help improve MAP.

Using the ad hoc track topics, we show that it is possible to automate the process of removing negative sentences and phrases and this improves retrieval effectiveness. Future work may involve detection of sub-sentence level negation patterns and handling complex negation phrases so as to prevent loss of keywords.

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. Kamps, J., Geva, S., Trotman, A., Woodley, A., Koolen, M.: Overview of the INEX 2008 ad hoc track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 1–28. Springer, Heidelberg (2009)
2. Pal, S., Mitra, M., Ganguly, D.: Parameter tuning in pivoted normalization for XML retrieval: ISI@INEX09 adhoc focused task. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 112–121. Springer, Heidelberg (2010)
3. Lovins, J.B.: Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics* 11, 22–31 (1968)
4. Hiemstra, D.: Using language models for information retrieval. PhD thesis, University of Twente (2001)
5. Sigurbjrnsson, B., Kamps, J., de Rijke, M.: An element-based approach to xml retrieval. In: INEX 2003 Workshop Proceedings (2004)
6. Ogilvie, P., Callan, J.: Hierarchical language models for XML component retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 224–237. Springer, Heidelberg (2005)
7. Singhal, A.: Term Weighting Revisited. PhD thesis, Cornell University (1996)
8. Ponte, J.M.: A language modeling approach to information retrieval. PhD thesis, University of Massachusetts (1998)
9. Palchowdhury, S., Pal, S., Mitra, M.: Using negative information in search. In: Proc. 2011 Second International Conference on Emerging Applications of Information Technology (EAIT 2011), pp. 53–56 (2011)

Automatically Generating Structured Queries in XML Keyword Search

Felipe da C. Hummel¹, Altigran S. da Silva¹,
Mirella M. Moro², and Alberto H.F. Laender²

¹ Departamento de Ciência da Computação
Universidade Federal do Amazonas
Manaus, Brazil

{fch, alti}@dcc.ufam.edu.br

² Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil

{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 query engine. 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 for the query results. Also, StruX uses IR techniques to determine in which elements a term is more likely to occur.

Keywords: XML, Keyword Search, XPath, Entities.

1 Introduction

Specifying queries through keywords is currently very common. Specially in the context of search engines on the *World Wide Web*, users with different levels of computer skills are familiar with keyword-based search. As a consequence, such a concept has been exploited outside the scope of the Web. For example, in relational databases, several methods [2, 10, 1, 8, 3, 21] have been proposed. 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.

Likewise, recently, 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 techniques 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 methods aim at finding, inside the XML document, subtrees in which each query term occurs at least once in one of its leaves, and then return 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, such methods tend to return a single-element subtree, a query result not desired in general.

Furthermore, after an initial indexing phase, they disregard the source XML document, as any query will be answered considering its own indexes only. This behavior may not be suitable in a dynamic environment in which data is frequently updated or when XML data is stored in a DBMS. Considering such an environment, it becomes relevant to develop XML keyword search methods 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 XML data.

This paper presents a novel method for keyword search over XML data based on a fresh perspective. Specifically, our method, called *StruX*¹, combines the intuitiveness of keyword-based queries with the expressiveness of XML query languages (such as XPath). Given a user keyword 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. Furthermore, it assigns each XPath expression a score that measures its likelihood of representing the most appropriate interpretation of the user query. Then, a system can submit one or more of such queries to a DBMS and retrieve the results. Like in [21], 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 in INEX 2010 data-centric track. Section 5 discusses the experimental results, and Section 6 concludes the paper.

2 Related Work

The basic principle behind *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. Thus, this section focuses on XML keyword search.

Using keywords to query XML databases has been extensively studied. Current work tackles keyword-based query processing by modeling XML documents as labeled trees, considers that the desired answer is a meaningful part of the XML document and retrieves 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. Following such an idea,

¹ The name *StruX* is derived from the latin verb form *struxi*, which means to structure or build things.

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

```

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 (element tags) and keyword-labels in the query. In a similar direction, XRANK [7] employs an adaptation of Google’s PageRank [22] to XML documents for computing a ranking score for the answer trees. A distinct approach is XSeek [18], in which query processing relies on the notion of entities inferred from DTDs. *StruX* shares a similar view with XSeek in this regard, but the latter does not generate queries.

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. Such method does not generate structured queries, as *StruX* does. Instead, it uses the structural knowledge to refine a retrieval model.

There are also other methods that go beyond simple keyword-based search. 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 search over XML streams. Although 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*

This section presents *StruX*, our method for generating structured XML queries from an unstructured keyword-based query. First, it gives an overview of *StruX* and then it details each of its steps.

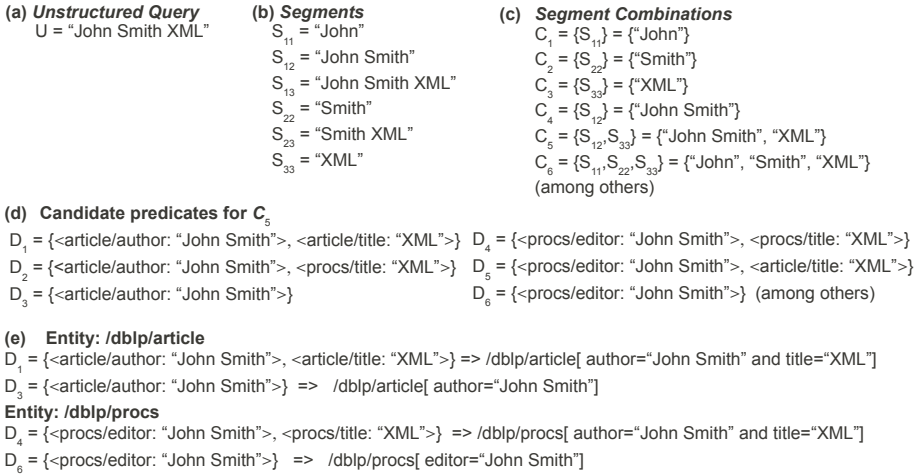


Fig. 1. Example of *StruX* steps

3.1 Overview

Algorithm 1 describes *StruX* general steps. Next, we describe each step using an example shown in Fig. 1. Given an unstructured keyword-based query as input (Fig. 1a), *StruX* first splits the input sequence of keywords into segments (Fig. 1b) and then generates combinations of these segments (Fig. 1c) to obtain possible semantic interpretations of the input unstructured query. This process assumes that each keyword-based query is an *unstructured query* U composed of a sequence of n terms, i.e., $U = \{t_1, t_2, \dots, t_n\}$. 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 the interest in two persons named “John Clark” and “Greg Austin”, respectively. 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 a different intention. Also, this intuition helps *StruX* dealing with possible ambiguous keywords.

The next step labels segment combinations with element names from the target XML database, forming sets of element-segment pairs (Fig. 1d), or *candidate predicates*. Once these candidate predicates have been formed, *StruX* finds adequate *entities* for each candidate (Fig. 1e). In fact, *StruX* relies on the concept of *entity* [18, 20] in order to intuitively represent real-world objects. For this task, it uses a few simple heuristics.

For instance, consider an element y that has multiple sub-elements x , then y is considered an *entity*. This can be observed by looking at the document schema (or by traversing the actual document) and verifying that x can occur multiple times within an instance of element y . For example, considering the DTD specification of *author* as “ $\langle \text{!ELEMENT author (book*, curriculum)} \rangle$ ”, *book* is a possible entity, while *curriculum*, according to this heuristic, is not.

In addition, we extend the concept of entity by adding another constraint that avoids very 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 the elements in which the keywords are more likely to occur. It then computes scores for candidate structured queries. Such scores are needed to determine which XML query represents more accurately the user’s intention. Finally, one or more top ranked structured queries are evaluated, returning the results to the users.

The final result of *StruX* is a query expressed in XPath², which specifies patterns of selection predicates on multiple elements that have some specified tree structure relationship. 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 methods 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 their tag label.

3.2 Input Transformation

The first step executed by *StruX* (Algorithm 1, line 2) splits 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 in Fig. 1a, the generated segments are shown in Fig. 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. Therefore, sets of tokens that are not in sequence such as “John” and “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:

$$SEA(n, S_{ij}) = \sum_{k=i}^j \text{TF-IEF}(n, t_k), \quad (1)$$

where, $\text{TF-IEF}(n, t_k)$ measures the relevance of a keyword t_k for an element type n in 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 element type”. This new measure is defined by Equation 2:

$$\text{TF-IEF}(n, t_k) = \text{TF}(n, t_k) \times \text{IEF}(t_k), \quad (2)$$

where each frequency is calculated by Equations 3 and 4, respectively.

² <http://www.w3.org/TR/xpath.html>

$$TF(n, t_k) = \frac{\log(1 + f_{nk})}{\log(1 + m)} \quad (3)$$

$$IEF(t_k) = \log\left(1 + \frac{e}{e_k}\right) \quad (4)$$

where 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 element 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 two (or more) keywords is intuitively more selective than a segment with a single keyword. For example, S_{13} (from Fig. 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.

$$WSEA(n, S_{ij}) = (1 + j - i) \times \sum_{k=i}^j TF-IEF(n, t_k) \quad (5)$$

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. 1c illustrates some of the combinations for the segments in Fig. 1b.

For each segment combination C_i , *StruX* generates all possible sets of element-segment pairs (Algorithm 1, line 5). For example, using combination $C_5 = \{\text{“John Smith”, “XML”}\}$, *StruX* obtains the sets of element-segment pairs illustrated in Fig. 1d, in which each set of pairs D_i is called a *candidate predicate*, or simply candidate. Note that $\langle \text{procs}/\text{title} \rangle$ in D_4 is different from $\langle \text{article}/\text{title} \rangle$ in D_5 as *StruX* identifies elements by their complete path to the root. At 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 Predicate 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 to Algorithm 1). During the traversal, the scores are propagated in a bottom-up fashion.

The propagation constant α (Algorithm 2, line 8) determines the percentage of a child’s score that is assimilated by its parent score (bottom-up propagation). As a result, Algorithm 2 produces a rank that 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 user’s original intention while

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:     $Rank.add(root)$ 

```

writing her query. By doing so, it is possible 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. 11e.

One important final note is that we chose to transform the keyword-based queries to XPath query statements for the language simplicity. 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 and the content of 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 Experimental Setup

Following the INEX 2010 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*, each one consisting in processing all topics under a certain setup.

Table 1. Description of the runs used in the experiments

Run	Structured Queries used	Target Datasets	Name
1	top-5	"movies"	<i>ufam2010Run1</i>
2	top-10	"movies"	<i>ufam2010Run2</i>
3	top-5	"movies", "actors"	<i>ufam2010Run3</i>
4	top-5	all except "other"	<i>ufam2010Run4</i>
5	top-5	all	<i>ufam2010Run5</i>
6	top-10	all	<i>ufam2010Run6</i>

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 $S_1^T, S_2^T, \dots, S_n^T$ as a result, being each S_i^T associated with a likelihood score, calculated as described in Section 3.3. Then, we executed the top- K structured queries over the IMDB datasets. We used runs with different types of target documents to assess how the redundancy and ambiguity between entities in the datasets affect *StruX* (e.g., “Steven Spielberg” may appear in many parts of a *movie* document and also on *person* documents as director, producer or actor). The complete description of the runs is presented in Table 1. In the following, we discuss details regarding the generation and the processing of the runs.

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 extended *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>` element 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 “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

```

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*

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. This could be performed by using some native XML DBMS such as *eXists-db*³. 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 using both formats.

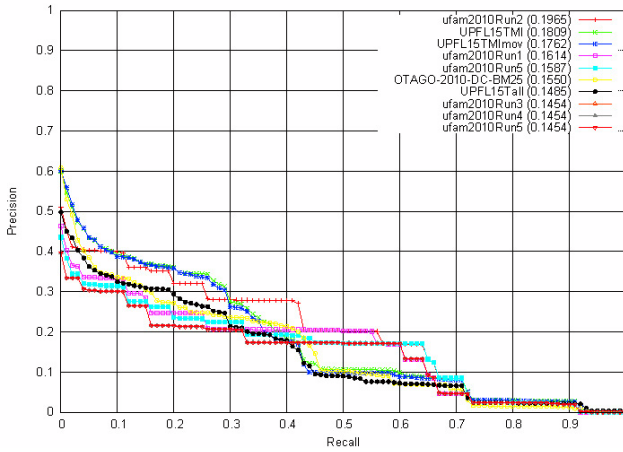
Regarding the scores of the results, as the final answers for the structured queries are produced by an external system (in our case a simple XPath matcher), there are no relevance scores directly associated to them. Thus, we simply assigned to the result the same score *StruX* has generated for the 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 Experimental Results

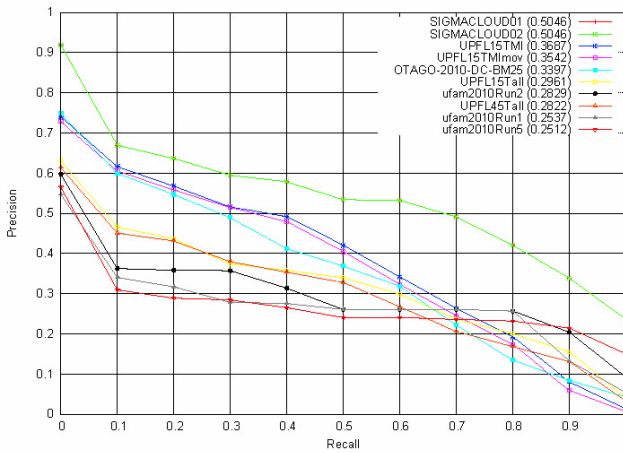
In this section, we present the results obtained in the INEX 2010 experiments.

Regarding the focused retrieval evaluation, *ufam2010Run2* was the best run among all, with MAiP value of 0.1965. As detailed in Section 4, this run returned structured queries targeting only *movies* documents.

³ <http://exist.sourceforge.net/>



(a)



(b)

Fig. 3. Focused Retrieval – MAiP Metric (a) and Document Retrieval – MAP Metric (b) Results

Notice that *ufam2010Run2* uses *top-10* structured queries, while *ufam2010Run1* (fourth best run) uses only *top-5* structured queries. This illustrates that considering more structured queries does not affect the quality of the results. On the contrary, it can even improve it. This happens because many of the generated *top-10* structured queries are often not feasible, i.e., they do not retrieve any results but, on the other hand, they do not harm the final results.

Regarding the document retrieval metric, our best run was, again, *ufam2010Run2*, who achieved the seventh best MAP value among all runs. It was followed by runs *ufam2010Run1* and *ufam2010Run6*, with no significant difference between them. Runs *ufam2010Run3*, *ufam2010Run4* and *ufam2010Run5* achieved the same MAP values, meaning that adding other target document types beyond *movie* and *actor* did not affect

the overall precision. Also, resembling the focused retrieval metric, the two runs with *top-10* structured queries performed better than their counterparts with *top-5* queries.

6 Conclusions

We presented a novel method called *StruX* 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 for IMDB datasets. The results demonstrated that query structuring is feasible and that our method is quite effective.

As future work, we plan to optimize even further our method. Specifically, we need to improve *StruX* for handling very large datasets. We also 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 projects InWeb, Amanajé and MINGroup (CNPq grants no. 573871/2008-6, 47.9541/2008-6 and 575553/2008-1), and by the authors' scholarships and individual grants from CNPq, CAPES, FAPEAM and FAPEMIG.

References

1. Aditya, B., Bhalotia, G., Chakrabarti, S., Hulgeri, A., Nakhe, C., Parag, P., Sudarshan, S.: BANKS: Browsing and Keyword Searching in Relational Databases. In: Proceedings of the 28th International Conference on Very Large Data Bases, pp. 1083–1086 (2002)
2. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: A System for Keyword-Based Search over Relational Databases. In: Proceedings of the 18th International Conference on Data Engineering, pp. 5–16 (2002)
3. Balmin, A., Hristidis, V., Papakonstantinou, Y.: ObjectRank: Authority-Based Keyword Search in Databases. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, pp. 564–575 (2004)
4. Barros, E.G., Moro, M.M., Laender, A.H.F.: An Evaluation Study of Search Algorithms for XML Streams. *Journal of Information and Data Management* 1(3), 487–502 (2010)
5. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSearch: A Semantic Search Engine for XML. In: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 45–56 (2003)
6. Demidova, E., Zhou, X., Zenz, G., Nejdl, W.: SUITS: Faceted User Interface for Constructing Structured Queries from Keywords. In: Proceedings of the International Conference on Database Systems for Advanced Applications, pp. 772–775 (2009)
7. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: Ranked Keyword Search over XML Documents. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 16–27 (2003)

8. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient IR-style Keyword Search over Relational Databases. In: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 850–861 (2003)
9. Hristidis, V., Koudas, N., Papakonstantinou, Y., Srivastava, D.: Keyword Proximity Search in XML Trees. *IEEE Transactions on Knowledge and Data Engineering* 18(4), 525–539 (2006)
10. Hristidis, V., Papakonstantinou, Y.: DISCOVER: Keyword Search in Relational Databases. In: Proceedings of 28th International Conference on Very Large Data Bases, pp. 670–681 (2002)
11. Hristidis, V., Papakonstantinou, Y., Balmin, A.: Keyword Proximity Search on XML Graphs. In: Proceedings of the 19th International Conference on Data Engineering, pp. 367–378 (2003)
12. Kim, J., Xue, X., Croft, W.: A probabilistic retrieval model for semistructured data. *Advances in Information Retrieval*, pp. 228–239 (2009)
13. Laender, A.H.F., Moro, M.M., Nascimento, C., Martins, P.: An X-ray on Web-Available XML Schemas. *SIGMOD Record* 38(1), 37–42 (2009)
14. Li, G., Feng, J., Wang, J., Zhou, L.: Effective Keyword Search for Valuable LCAs over XML Documents. In: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, pp. 31–40 (2007)
15. Li, G., Feng, J., Wang, J., Zhou, L.: An Effective and Versatile Keyword Search Engine on Heterogenous Data Sources. *Proceedings of the VLDB Endowment* 1(2), 1452–1455 (2008)
16. Li, Y., Yang, H., Jagadish, H.V.: NaLIX: an Interactive Natural Language Interface for Querying XML. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 900–902 (2005)
17. Li, Y., Yu, C., Jagadish, H.V.: Schema-Free XQuery. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, pp. 72–83 (2004)
18. Liu, Z., Chen, Y.: Identifying Meaningful Return Information for XML Keyword Search. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 329–340 (2007)
19. Luo, Y., Wang, W., Lin, X.: SPARK: A Keyword Search Engine on Relational Databases. In: Proceedings of the 24th International Conference on Data Engineering, pp. 1552–1555 (2008)
20. Mesquita, F., Barbosa, D., Cortez, E., da Silva, A.S.: FleDEx: Flexible Data Exchange. In: Proceedings of the 9th ACM International Workshop on Web Information and Data Management, pp. 25–32 (2007)
21. Mesquita, F., da Silva, A.S., de Moura, E.S., Calado, P., Laender, A.H.F.: LABRADOR: Efficiently publishing relational databases on the web by using keyword-based query interfaces. *Information Process Management* 43(4), 983–1004 (2007)
22. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project (1998)
23. Sun, C., Chan, C.Y., Goenka, A.K.: Multiway SLCA-based keyword search in XML data. In: Proceedings of the 16th International Conference on World Wide Web, pp. 1043–1052 (2007)
24. Xu, Y., Papakonstantinou, Y.: Efficient Keyword Search for Smallest LCAs in XML Databases. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 527–538 (2005)

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. We first propose a classification of INEX topics and analyze several characteristics of the relevance assessments from INEX 2009 for each of the topic classes. The goal of our study is to investigate whether there are differences in relevance judgements between topic classes in order to use this information at retrieval time. We also present the experiments we performed on the INEX 2010 data. In the ad-hoc track we study the performance effects of changing the article order (fetching phase) while in the data-centric track we experiment with the use of different indices and retrievable element types. Our main finding is that indexing uniquely movie documents leads to much better performance than indexing the complete collection.

Keywords: XML, focused retrieval, INEX, query-based retrieval, query classification.

1 Introduction

Retrieval tasks such as XML retrieval, where focused access to relevant information is provided, allow users to perform very focused searches and pose restrictions on the type of information being requested (e.g., I want references or experimental results). This is one of the reasons why several query languages and interfaces have been designed—to allow users to explicitly express more complex needs. However, these tools are not always available and users often specify in their *keyword* queries not only what they are looking for but also the type and the specificity of the information they are searching for. Thus, the number and variety of topic types that XML retrieval introduces differ from those of traditional document retrieval, where the task is to return whole documents.

In this paper we propose a classification of XML retrieval topics based on three different dimensions: 1) the type of information sought (general or restricted), 2) the specificity of the topic (generic or specific), and 3) the complexity of the topic (simple or compound). Once a classification is defined, it can be used in multiple ways. Our goal is to use it to perform specific retrieval strategies for each of the topic classes. A classification can also help to provide a more balanced benchmark topic-set, avoiding to reward retrieval systems that perform well solely on the most popular topic type.

We analyze several characteristics of the relevant judgements from INEX 2009 for each of the topic classes defined in our classification. The aim of the study is to investigate whether the differences between topic classes could be used to decide on specific retrieval strategies for each of the topic types, a first step towards query-type based focused retrieval.

We also describe the experiments performed on the INEX 2010 data for two different tracks: ad-hoc and data-centric. Our experiments for the ad-hoc track study the performance effects of changing the article order (fetching phase) while in the data-centric track we experiment with the use of different indices and retrievable element types.

The rest of the paper is organized as follows: In Section 2 we present our INEX topic classification and explain the dimensions used. The analysis performed on INEX 2009 data is described in Section 3. Section 4 presents the results of our experiments for both tracks: ad-hoc and data-centric. Finally, we discuss the main contributions and future work in Section 5.

2 INEX Topic Classification

In this section we propose a classification of INEX topics. We extend our previous work on classification of INEX information needs [2] with a new dimension: the type of information sought.

2.1 Dimensions

Our classification uses the following three dimensions: 1) the type of information sought (general or restricted), 2) the specificity of the topic (generic or specific), and 3) the 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).

Type of Information Sought. In an XML retrieval scenario, where focused access to relevant information is provided, users can pose restrictions on the type of information being searched for (e.g., I want images or results).

We classify topics into *General* and *Restricted* requests. **General requests** are those that ask for *any type of* information about a topic, without restrictions. Note that the topic might be very specific but the type of information the user wants to see is generic (any type of information about it). **Restricted requests** are those in which some type of constraint on the type of information being sought is specified. This constraint can be topical (e.g., I want exercises or experiment results) or structural (e.g., I want references or images). In other words, the restrictions can specify which part of the content has to be returned, e.g., “I like to know the *speed capacity* of vehicles” (not any other information on vehicles) or the type of object that it is returned, e.g., “I like too see *images* of sunflowers” (not any other information/object about sunflowers). Note that *General* requests are those that are typically used in web search and document retrieval, where the task is to return whole documents or web pages.

Complexity. In the *complexity* dimension, two categories are used: *Simple* and *Compound*. **Simple requests** are those that ask for information about just one topic or aspect of a topic (i.e., mono-faceted requests). While **Compound requests** are those that ask for information about several topics or 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).

Specificity. In the *specificity* dimension, we classify requests into *Specific* and *Generic*, depending on the topical broadness of the information being searched for. In other words, **General requests** are those that ask for information about a broad topic, while **Specific requests** are those that ask for information about a narrow topic. Note that here we talk about the information being searched for and not about the *type* of information being searched for.

While the *specificity* and the *complexity* of the request are dimensions that have already been used to classify standard IR requests [5], the *type of information sought* is specific to focused retrieval.

Notice also the difference between restricting a topic (e.g., classical movies) and restricting the type of information sought (e.g., pictures of movies). The first one would be *specific* while the second one would be *structurally restricted*.

We hypothesize that the characteristics of the relevant information between the classes of each dimension differ. If these differences exist, XML retrieval systems should use this information in order to optimize their retrieval performance by using specific retrieval strategies for each of the topic classes.

2.2 Data Classification

The INEX ad-hoc topics are created by the participants following precise instructions. Candidate topics contain a short CO (keyword) query, an optional structured CAS query, a phrase title, a one line description of the search request, and a narrative with details of the topic of request and the task context in which the information need arose. An example of an INEX topic with all its fields can be seen in Table 1. We used the description field of the topics to classify the INEX 2009 topics into different classes. If needed, we used the narrative field from the topic to clarify. All 68 topics were classified by two different volunteers. Table 2 shows the resulting topic classes, the number of topics belonging to each class and gives an example for each of them. We also investigate how *intuitive* the dimensions and categories used in our classification are. We do so by analyzing the level of agreement between volunteers. Table 3 shows the agreement on each of the dimensions between the two volunteers. We can see that the agreements on the first two dimensions are rather high, suggesting that these dimensions are quite intuitive and objective. However, the specificity dimension has a very low agreement percentage and it is probably too subjective to be used in a real setting.

Table 1. Example of INEX topic

Topic ID	2009011
Title	olive oil health benefit
CAS query	//food[about(., olive oil) and about(., health benefit)]
Phrase title	“olive oil” “health benefit”
Description	Find information about what sort of health benefit olive oil has
Narrative	I’m a health/beauty buff. I recently learned that olive oil is “good for you”. What are the specific health/beauty benefits for consuming olive oil? Any article that mentions health benefits of olive oil is fine, EXCEPT those in which the claim is based on either unpopular/obscure or unscientific methods. So for example, if XXX diet recommends consuming olive oil then it’s irrelevant. Note that since Mediterranean diet is not a weight-loss fat diet, but the traditional Mediterranean ways of eating, an article describing the health benefits of olive oil in this setting is relevant. Anything outside of health benefit is irrelevant, how olive oil is produced, the different grades of olive oil etc.

3 Relevance Assessments Analysis

In this section we investigate whether the characteristics of the relevance judgements differ between the different topic classes described above. Having in mind that for some classes the number of topics is generally too low to draw statistically significant conclusions, we analyze INEX 2009 relevance judgements and look at the relevance characteristics of each of the topic classes. We analyze the following characteristics: 1) the number of relevant documents, 2) the density of the relevant documents, and 3) the number and size of the relevant fragments.

Table 2. Number of INEX 2009 topics belonging to each of the topic classes and example of topic description for each of them

Dimension	Class	Num.	Example
Type of information sought	General	64	Information about Nobel prize.
	Restricted (structurally)	2	Explain “mean average precision” and “reciprocal rank” with images or plots . Provide references in proceedings and journals.
	Restricted (topically)	2	I want to know vehicles and its speed capacity
Complexity	Simple	46	Information about classical movies
	Compound	22	Find information about applications of bayesian networks in the field of bioinformatics
Specificity	Generic	13	I want to find some information about IBM computer
	Specific	55	Find information on causes of vitiligo and treatment for it

Table 3. Agreement between the two volunteers that classified the topics

Type of information sought	Complexity	Specificity
94%	85 %	46%

Number of Relevant Documents. By number of relevant documents we refer to the number of unique documents in the collection that contain relevant information given a topic, even if the fraction of relevant information is small. Figure 1 (upper part) shows the average number of documents containing relevant information for each of the topic classes. We can clearly see that *restricted* requests tend to find much less relevant documents than the *general* ones. On average, there are 18 relevant documents for the *restricted* topics and 75 for the *general* ones (13 and 56.5 when looking at the median). Although the difference is not that big for the other dimensions, we see that *compound* requests tend to find less relevant documents than *simple* ones (51 vs. 81 on average and 26 vs. 58 when looking at the median). *Specific* topics are also satisfied with a smaller number of documents than *generic* ones (65 vs. 100 on average and 48 vs. 65 when looking at the median).

These tendencies are not surprising, it seems reasonable that the more complex, restricted, and specific a topic is, the more difficult is to find information that satisfies it.

Density. We also analyze how densely relevant are the documents that contain relevant information. According to recent work [6], focused search works better

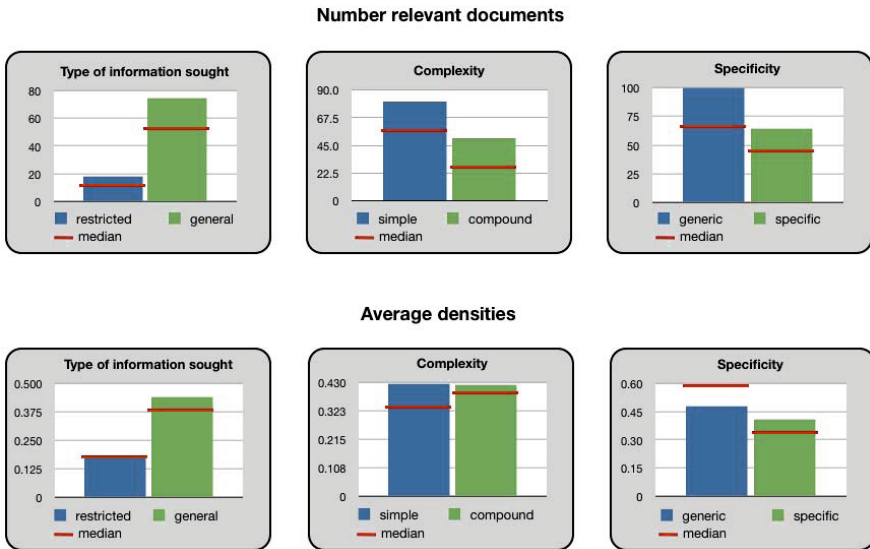


Fig. 1. Average number of documents containing relevant information (top) and average density of those documents (bottom)

on sparsely dense documents. We define density of a document as the percentage of relevant text contained in that document (i.e., ratio of relevant text to all text). Text size is given by the number of characters.

Figure 1 (bottom part) shows the average density of the documents containing relevant information for each of the topic classes. We can see that documents that contain relevant information for the *restricted* topics tend to be sparsely dense. On average, 18% of the text in a document is relevant for the *restricted* topics while that is 44% for the general ones (19% and 38% when looking at the median). Focused retrieval seems to be more desirable for *restricted* topics.

Regarding the other dimensions, we see that there is not much difference in terms of density between the *compound* and *simple* topics. In both cases, documents are quite dense on average. The difference is bigger in the specificity dimension. While *generic* topics tend to be answered with highly dense documents (on average 48% and median 59%), specific topics tend to be answered with less dense documents (41% on average and median 34%).

Number and Size of Relevant Fragments. To see how the relevant information is distributed within an article, we look at the number and size of relevant fragments, the fragments that contain the relevant information. Figure 2 shows this information for each of the topic classes. While there are not big differences in the number of relevant fragments between the *restricted* topics (average 2, median 2) and the *general* topics (average 1.7, median 1.5), the fragments for the *restricted* topics tend to be much smaller (see Figure 2 bottom part). On average, relevant fragments for the *restricted* topics are 540 characters long

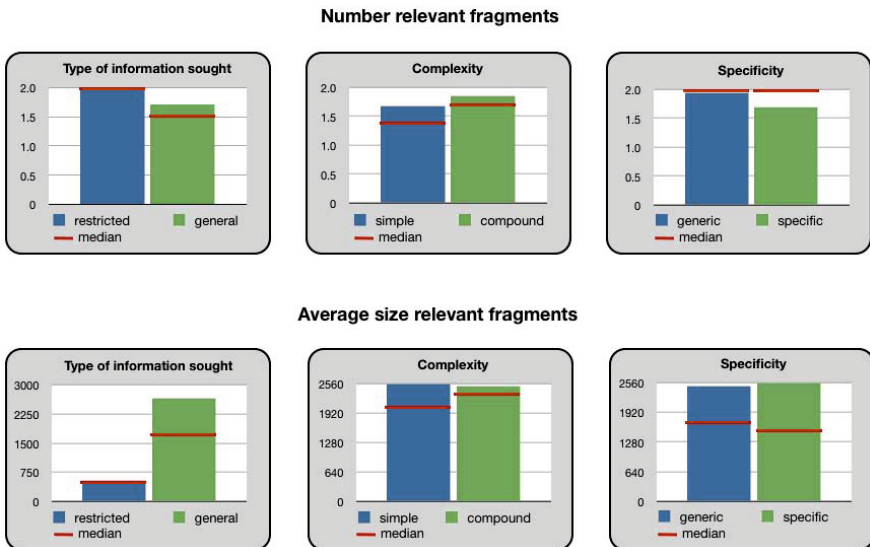


Fig. 2. Number and average size of relevant fragments

(median 512) while the average length for the *general* topics is 2668 (median 1644).

This is not the case for the other two dimensions where the number and average size of the relevant fragments are very similar between classes. We can see that, in general, a very small number of very long fragments are assessed as relevant, not the best scenario for focused retrieval.

We also look at two characteristics of the topic itself: 1) the number of query terms and 2) the type of CAS query. If there are differences between topic classes, these characteristics can help to automatically classify topics.

Number of Query Terms. By number of query terms we refer to the number of terms in the title field of the topic after removing stop-words. Figure 3 (upper part) shows the average number of query terms for each of the topic classes. *Restricted* topics tend to be long, they have an average of 6.5 terms per topic (median 6) while the average number of query terms for the *general* ones is 3.7 (median 4). The difference can be explained from the fact that *restricted* requests specify not only what the users are searching for but also the type of information they would like to see. We can see a similar pattern for the specificity dimension. While *generic* topics are expressed, on average, with a very small number of query terms (2.5, median 2), *specific* topics tend to be longer (average 4, median 4). We can also see that there are not big differences regarding the number of query terms between *simple* and *compound* topics (complexity dimension).

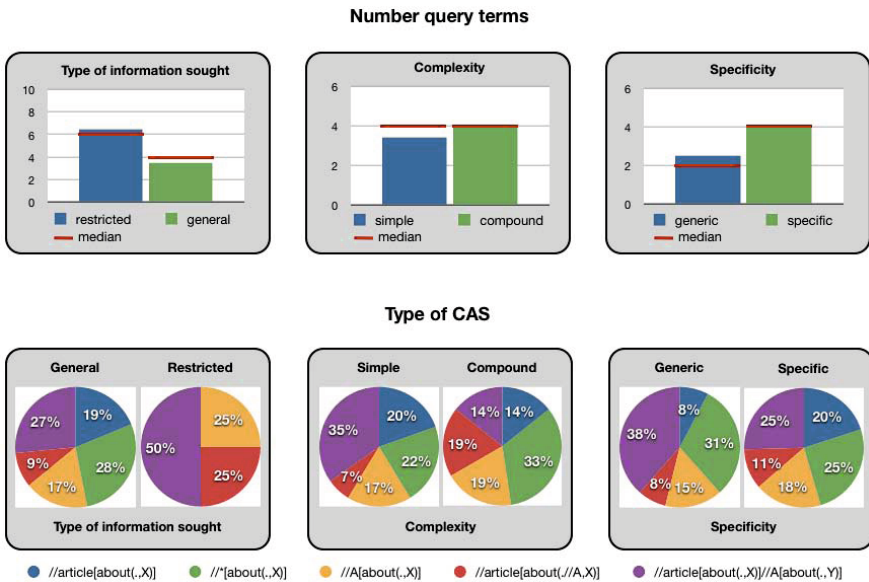


Fig. 3. Average number of query terms and percentages of CAS query types

Table 4. CAS query patterns

Pattern	Meaning
//article[about(.,X)]	Return articles about topic X
//*[about(.,X)]	Return any type of element about topic X
//A[about(.,X)]	Return element types A about topic X (where A can be any element type except article)
//article[about(./A,X)]	Return articles that contain an element type A about topic X
//article[about(.,X)]//A[about(.,Y)]	Return element types A about topic Y that are contained in articles about topic X.

Type of CAS Query. We also analyze which types of CAS query are associated with each of the topic classes. We used five different CAS patterns to classify all topics (see Table 4). Figure 3 (bottom part) shows the percentage of CAS queries of each pattern for each of the topic classes. We can see some differences. While *restricted* topics tend to be specified with longer and more specific CAS queries, the majority of the *general* ones use the most generic form of CAS query 1. More surprising is that a big portion of the *simple* and *generic* topics use the more specific CAS queries. In general terms however, it is difficult to associate any of the CAS query pattern to a specific topic class.

4 Experiments

This section describes the setup and discusses the results of the experiments carried out for the ad-hoc and the data-centric tracks of INEX 2010. 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. Topics and documents have been pre-processed using the porter stemmer 3 and the smart stop-word list 4.

4.1 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. We study the importance of the fetching phase, i.e., the performance effects of changing the article order.

¹ Note that topics that were submitted without a CAS query were assigned the most generic one: //article[about(.,X)].

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 mentioned above, we 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 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 where the second most relevant paragraph appears, etc.).

Our original submissions were not valid due to a bug in our code and could not be evaluated. The results of the *un-official* runs (after fixing) are shown in Table 5. We can see that changing the article order affects considerably the performance of the run. The reference run article order outperforms the other two. Paragraphs are not good indicators of article relevance. Ranking articles by their most relevant paragraph performs the worst.

Table 5. Un-official results for the relevant in context runs. The number in parentheses indicates the estimated run position in the official ranking.

run name	MAGP (est. position)
UPFpLM45coRC1	0.1109 (34)
UPFpLM45coRC2	0.1571 (9)
UPFpLM45coRC3	0.0763 (39)

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 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 where the second most relevant section appears, etc.).

As in the previous task, our original submissions were not valid due to a bug in our code and could not be evaluated. The results of the *un-official* runs (after fixing) are shown in Table 6.

We see similar performances as in the previous task; the reference run article order outperforms the other two. In absolute numbers paragraph runs seem to outperform section runs. However, the lower performance of the section runs could be due to the task restrictions. When looking at the relative position of the runs regarding the other groups, the three runs performed relatively well (top 15-20).

Table 6. Unofficial results for the restricted relevant in context runs. The number in parentheses indicates the estimated run position in the official ranking.

run name	MAGP (est. position)
UPFP _{LM45coRRC1}	0.0894 (15)
UPFP _{LM45coRRC2}	0.1210 (13)
UPFP _{LM45coRRC3}	0.0633 (17)

Table 7. Official results for the restricted focused runs. The number in parentheses indicates the run position in the official ranking.

run name	char prec (position)
UPFP _{LM45coRF1}	0.2984 (19)
UPFP _{LM45coRF2}	0.3066 (15)
UPFP _{LM45coRF3}	0.1156 (30)

Table 8. 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
UPFL15TMI _{mov}	movies	movie	0.15

Table 9. Official results for the data-centric track. The number in parentheses 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)
UPFL15TMI _{mov}	0.2434 (3)	0.1762 (3)	0.3542 (4)

Restricted Focused. The Restricted Focused task aims at giving a quick overview of the relevant information in the whole of 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 1,000 characters per topic. The assumption is that users prefer to see an overview based on the largest number of articles rather than seeing several relevant paragraphs of the same article. Our three official runs are again based on different article order as in the previous tasks; 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 paragraph appears, followed by the article where the second most relevant paragraph appears, etc.).

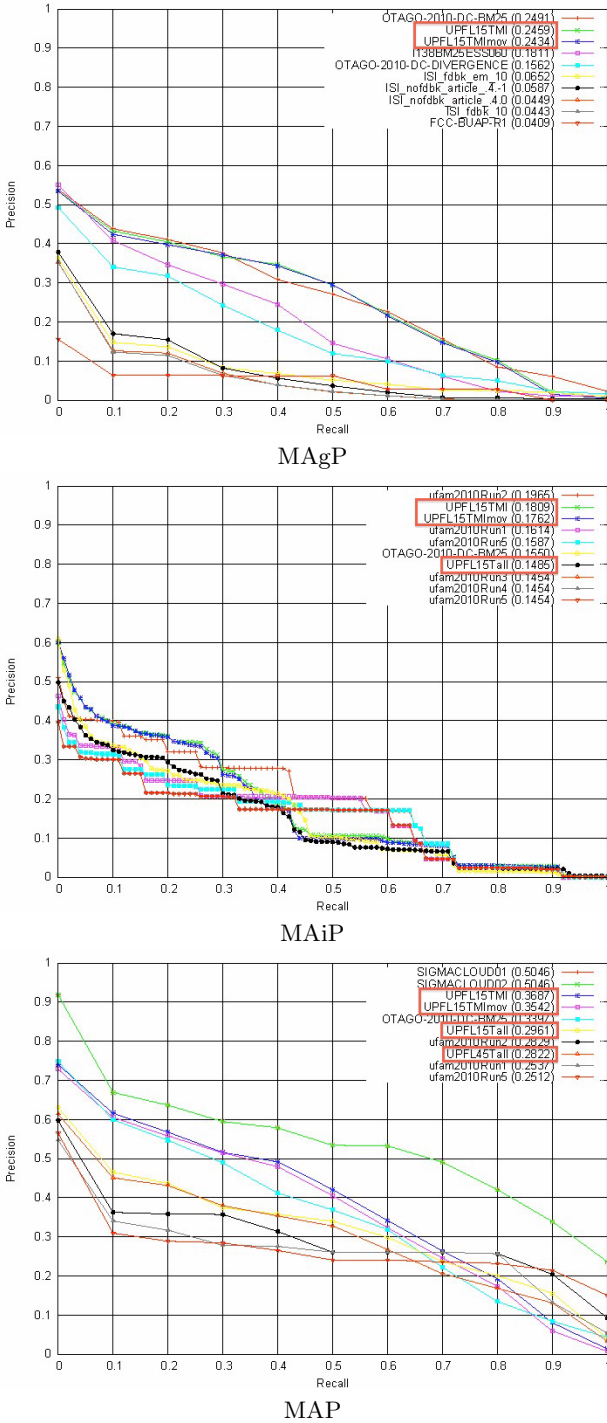


Fig. 4. Official evaluation graphs for the data-centric track

The results of these runs are shown in Table 7. As in the other tasks, 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 order and the reference run order and our paragraph run order. Paragraphs are not good estimators of the total relevance of an article. In other words, a relevant paragraph does not imply that the article is relevant to the same degree.

4.2 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 experimented with two different values that have been successfully used in other collections. We also experimented 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 8 shows the parameters used for each of our official runs and Table 9 the official results. 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 noise. We also see that lambda 0.15 always performs better than lambda 0.45, indicating that it is better to give less emphasis to the collection statistics. Figure 4 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, it does not manage to do so at middle and/or high recall levels (MAiP and MAP graphs). This is because a large part of the collection is not indexed, which makes it difficult, if not impossible, to have a high overall recall.

5 Discussion and Conclusions

This paper described our participation at INEX 2010. We presented a classification of INEX topics and an analysis of the characteristics of the relevance assessments for each of the topic classes. The goal of our study was to investigate whether there are differences in relevance judgements between topic classes in order to use them for retrieval. We have seen, for instance, that *restricted* topics have a small set of relevant documents which are sparsely dense and relevant information is contained in small fragments of documents. Although some of the analyzed relevance characteristics differ between classes, it is not clear whether this information could be used for retrieval. More data needs to be analyzed in order to see whether these differences are statistically significant.

The classification presented is based on three different dimensions (type of information sought, complexity, and specificity), generic enough to be used in

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.

other focused retrieval scenarios. Our goal is to use it to perform different retrieval strategies for each of the topic classes. A classification can also help to provide a more balanced benchmark topic-set, avoiding to reward retrieval systems that perform well solely on the most popular topic type. We note that not all the dimensions are objective enough to be easily used. The specificity of a topic is a subjective matter and it might not be easy to apply in real settings.

As future work we plan to investigate whether it is beneficial to use different retrieval strategies for the different topic types.

We reported on our experiments for INEX 2010, in the ad-hoc and data-centric tracks. In the ad-hoc track we studied the performance effects of changing the article order (fetching phase) while in the data-centric track, we experimented with the use of different indices and retrievable element types. Our results on the ad-hoc track confirm that article order is a very important factor on the overall performance of the systems. In all of our experiments, the article order of the reference run outperforms the other runs. In the data-centric track, our main finding is that indexing only the movie documents leads to much better performance 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. Strohman, T., Metzler, D., Turtle, H., Croft, W.B.: Indri: a language model based search engine for complex queries. In: Proceedings of the International Conference on Intelligent Analysis (2005)
2. Ramírez Camps, G.: Structural Features in XML Retrieval. PhD thesis, University of Amsterdam (2007)
3. Porter, M.F.: An algorithm for suffix stripping. *Program* 14(3), 130–137 (1980)
4. Salton, G.: The SMART Retrieval System Experiments in Automatic Document Processing. Prentice-Hall, Inc., Upper Saddle River (1971)
5. Ingwersen, P., Järvelin, K.: The Turn: Integration of Information Seeking and Retrieval in Context. The Information Retrieval Series, vol. 18. Springer, Heidelberg (2005)
6. Arvola, P., Kekalainen, J., Junkkari, M.: Focused Access to Sparsely and Densely Relevant Documents. In: Proceedings of SIGIR 2010 (2010)

BUAP: A First Approach to the Data-Centric Track of INEX 2010*

Darnes Vilariño, David Pinto, Carlos Balderas, Mireya Tovar, and Saul León

Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla, México
{darnes,dpinto,mtovar}@cs.buap.mx, charlie_kanon@hotmail.com,
saul.ls@live.com

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 Data-Centric track of the Initiative for the Evaluation of XML retrieval (INEX 2010). This track is 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 focused 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. The new Data-Centric 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 promoting 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 differently by the database (DB) and the information retrieval (IR) research communities. The DB community has focussed on developing query languages and efficient evaluation algorithms used primarily for Data-Centric XML documents. On the other hand, the IR community has focussed on document-centric XML documents

* This work has been partially supported by the CONACYT project #106625, VIEP # VIAD-ING11-I, as well as by the PROMEP/103.5/09/4213 grant.

by developing and evaluating techniques for ranked element retrieval. Recent research trends show that each community is willing to adopt the well-established techniques developed by the other to effectively retrieve XML content [3].

The Data-Centric track uses the IMDB data collection newly built from the following website: <http://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 document 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 how 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.

The original XML file has been previously processed in order to eliminate stopwords and punctuation symbols. Moreover, we have transformed the original hierarchical structure given by XML tags to a similar representation which may be easily analyzed by our parser.

For the presented approach we have used an inverted index tree in order to store the XML templates of the corpus. 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).

With respect to the dictionary of the inverted index, we have considered to include both, the term and the XML tag (the last one in the hierarchy). In Figure 1, we show an example of the inverted index (pay special attention to the dictionary). The aim was to be able to find 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 full path in which the term occurs, however, it would be better to use all the hierarchy in the dictionary. Further experiments would verify this issue.

In the following subsection we present the algorithms developed for indexing and searching information based on content and structure.

2.1 Data Processing

Before describing the indexing techniques used, we first describe the way we have processed the data provided for the competition. We have cleaned the XML files


```

title hannibal 40      : 981731:1 994171:1 78811:1 [...] 1161611:1 [...]
character lecturer 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. 1. Example of the type of inverted index used in the experiments

in order to obtain an easy way of identifying the XML tag for each data. For this purpose, as we previously mentioned, 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 2, which is then traduced to the sentence shown in Figure 3. 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.

In the competition we submitted two runs. The first one uses the complete data of each record (word n -gram), whereas the 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 topic showing the second approach is given in Figure 4, whereas its traduced version is given in Figure 5.

In order to obtain the list of candidate documents for each topic, we have calculated the similarity score between the topic and each corpus document as shown in Eq. (1) 4, which was implemented as presented in Algorithm 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. 2. An example of a given query (topic)

```

2010001 3 1 //movie//director yimou zhang and //movie//releasedate 2010
or //movie//releasedate 2009

```

Fig. 3. Representation of the topic

```

<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. 4. An example of another topic

```

2010025 19 1 //movie tom and //movie hanks and //movie steven and
//movie spielberg

```

Fig. 5. The representation of a topic by splitting the data

$$\text{SIM}(q, d) = \sum_{c_k \in B} \sum_{c_l \in B} CR(c_k, c_l) \sum_{t \in V} \text{weight}(q, t, c_k) \frac{\text{weight}(d, t, c_l)}{\sqrt{\sum_{c \in B, t \in V} \text{weight}(d, t, c)^2}} \quad (1)$$

where the CR function is calculated as shown in Eq. (2), V is the vocabulary of non-structural terms; B is the set of all XML contexts; and $\text{weight}(q, t, c)$ and $\text{weight}(d, t, c)$ are the weights of term t in XML context c in query q and document d , respectively (as shown in Eq. (3)).

$$CR(c_q, c_d) = \begin{cases} \frac{1+|c_q|}{1+|c_d|} & \text{if } c_q \text{ matches } c_d \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where c_q and c_d are the number of nodes in the query path and document path.

$$weight(d, t, c) = idf_t * wf_{t,d} \quad (3)$$

where idf_t is the inverse document frequency of term t , and $wf_{t,d}$ is the frequency of term t in document d .

Algorithm 1. Scoring of documents given a topic q

Input: q, B, V, N : Number of documents, *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      $w_q = weight(q, t, c_q)$ 
5     foreach  $c \in B$  do
6       if  $CR(c_q, c) > 0$  then
7          $postings = GetPostings(c, t)$ 
8         foreach  $posting \in postings$  do
9            $x = CR(c_q, c) * w_q * PostingWeight(posting)$ 
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) are presented in Table 1 and in Figure 6.

As may be seen, we have obtained a low performance, which we consider is derived of the fact of using only one tag for identifying each indexed term. We

Table 1. Evaluation measured as focused retrieval (MAGP)

#	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	UPFL15TMI _{mov}
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

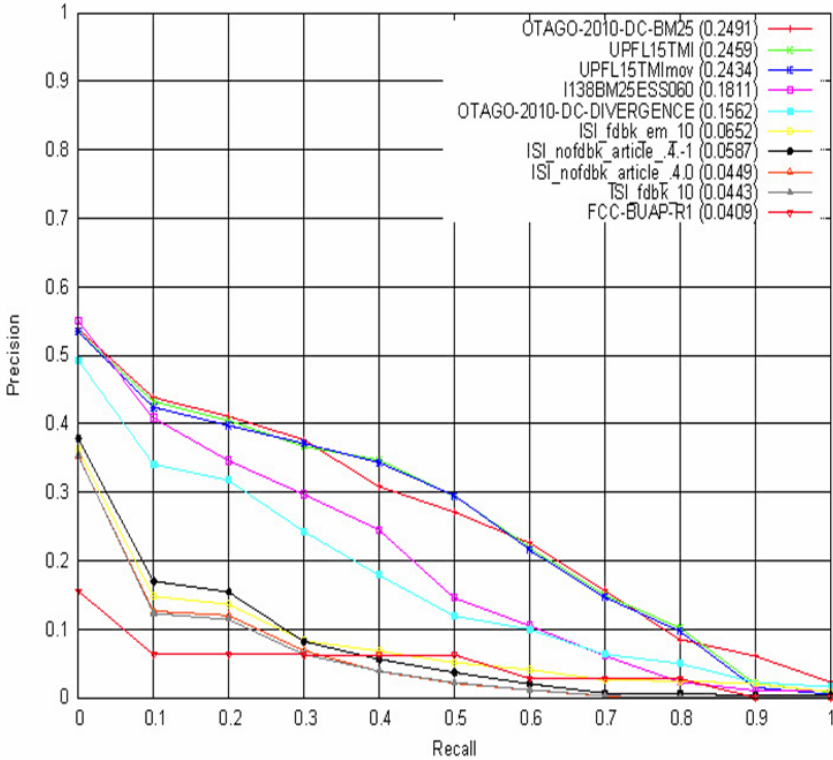


Fig. 6. Evaluation measured as focused retrieval (MAGP)

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.

Table 2. Evaluation measured as document retrieval (whole document retrieval)

#	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	UPFL15TMIImov
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

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, in particular, in the Data-Centric track of the Initiative for the Evaluation of XML retrieval (INEX 2010).

We presented an indexing method based on an inverted index with XML tags embedded. For each category (movies, actors, producers, directors and others), we constructed an 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: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 133–144. Springer, Heidelberg (2010)
2. Wang, Q., Trotman, A.: Task description of INEX 2010 Data-Centric track. In: Proc. of INEX 2010 (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, pp. 575–586. ACM, New York (2006)
4. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2009)

Overview of the INEX 2010 Interactive Track

Nils Pharo¹, Thomas Beckers², Ragnar Nordlie¹, and Norbert Fuhr²

¹Faculty of Journalism, Library and Information Science, Oslo University College, Norway
nils.pharo@jbi.hio.no, ragnar.nordlie@jbi.hio.no

²Department of Computer Science and Applied Cognitive Science,
University of Duisburg-Essen, Germany
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 task. 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 Wikipedia corpus was used. In 2009 [7] the iTrack switched to a collection consisting of book metadata compiled 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 of time and effort. The data collected has been subjected to 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 had 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 available (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 was chosen with the intention 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 was designed to let searchers use a broad selection of metadata, in particular combining topical searches with the use of review data. The tasks were thus designed to inspire users to create “polyrepresentative” [17] search strategies, i.e. to use explorative search strategies, which should give us data on query development, metadata type preference and navigation patterns.

At the second stage we have attempted to simulate searchers that are in a rather mechanistic data gathering mode. The tasks have also been designed to focus on non-topical characteristics of the books. Information should typically be found in publisher's texts and possibly in user-provided tags.

The self-selected task was intended to function as a “control” task for comparison with the performance of two others.

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

3 research groups participated in this year’s track: Oslo University College, University of Duisburg-Essen, and University of Glasgow. Data from a total of 147 sessions performed by 49 test subjects were collected from October 2010 to January 2011. The participation was compensated for some participant with a EUR 12 Amazon voucher.

4 Research Design

4.1 Search System

The experiments were conducted on a Java-based retrieval system built within the ezDL framework¹, 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 for the experiments.

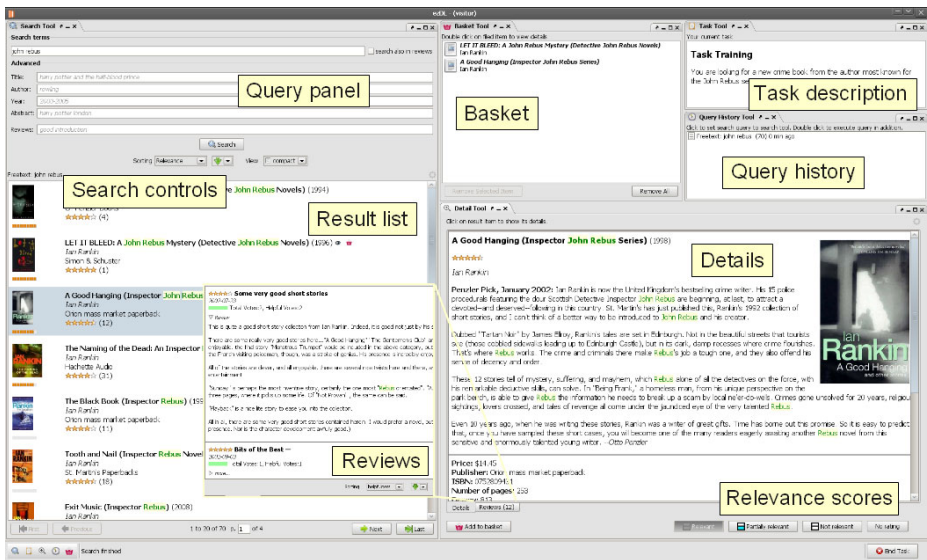


Fig. 1. The search system interface

¹ <http://ezdl.de>, <http://www.is.inf.uni-due.de/projects/ezdl/>

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, i.e. 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.. 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 has been 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 presented 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 was 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 provides 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 Results and Future Plans

Table 1 shows the distribution of systems and tasks. As can be seen very few searchers chose task 4 (Nobel Prize winning novels), the other tasks were fairly evenly distributed. For some unknown reason one searcher performed two tasks in system A and one task in system B, although our distribution system was programmed to allocate three system B task for this user. This explains the system distribution being slightly skewed.

We are currently analyzing our data, which will be presented in forthcoming conference and journal papers. A primary focus of the analysis will be searchers’ choice of sources of information for the completion of the tasks. Amongst the issues that we plan to look at is the effect of searchers’ topic knowledge and the influence of task types.

Table 1. Distribution of systems and tasks

	Task							Total
	1	2	3	4	5	6	7	
System A	14	10	2	6	8	11	26	77
B	11	10	2	13	5	6	23	70
Total	25	20	4	19	13	17	49	147

References

- [1] Malik, S., Trotman, A., Lalmas, M., Fuhr, N.: Overview of INEX 2006. In: Fuhr, N., Lalmas, M., Trotman, A. (eds.) INEX 2006. LNCS, vol. 4518, pp. 1–11. Springer, Heidelberg (2007)
- [2] Ruthven, I.: Interactive Information Retrieval. *Annual Review of Information Science and Technology* 42, 43–91 (2008)
- [3] Tombros, A., Larsen, B., Malik, S.: The Interactive Track at INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 410–423. Springer, Heidelberg (2005)
- [4] Larsen, B., Malik, S., Tombros, A.: The Interactive Track at INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 398–410. Springer, Heidelberg (2006)
- [5] Malik, S., Tombros, A., Larsen, B.: The Interactive Track at INEX 2006. In: Fuhr, N., Lalmas, M., Trotman, A. (eds.) INEX 2006. LNCS, vol. 4518, pp. 387–399. Springer, Heidelberg (2007)
- [6] Pharo, N., Nordlie, R., Fachry, K.N.: Overview of the INEX 2008 Interactive Track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 300–313. Springer, Heidelberg (2009)
- [7] Pharo, N., Nordlie, R., Fuhr, N., Beckers, T., Fachry, K.N.: Overview of the INEX 2009 Interactive Track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 303–311. Springer, Heidelberg (2010)
- [8] Pharo, N., Nordlie, R.: Context Matters: An Analysis of Assessments of XML Documents. In: Crestani, F., Ruthven, I. (eds.) CoLIS 2005. LNCS, vol. 3507, pp. 238–248. Springer, Heidelberg (2005)
- [9] Hammer-Aebi, B., Christensen, K. W., Lund, H., Larsen, B.: Users, structured documents and overlap: interactive searching of elements and the influence of context on search behaviour. In: Ruthven, I. et al. (eds.) *Proceedings of Information Interaction in Context : International Symposium on Information Interaction in Context : IIIiX 2006*, Copenhagen, Denmark, October 18–20, pp. 80–94, Royal School of Library and Information Science (2006)
- [10] Pehcevski, J.: Relevance in XML retrieval: the user perspective. In: Trotman, A., Geva, S. (eds.) *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology: Held in Seattle, Washington, USA, August 10*, pp. 35–42. Department of Computer Science, University of Otago, Dunedin, New Zealand (2006)

- [11] Malik, S., Klas, C.-P., Fuhr, N., Larsen, B., Tombros, A.: Designing a user interface for interactive retrieval of structured documents: lessons learned from the INEX interactive track? In: Gonzalo, J., Thanos, C., Verdejo, M.F., Carrasco, R.C. (eds.) ECDL 2006. LNCS, vol. 4172, pp. 291–302. Springer, Heidelberg (2006)
- [12] Kim, H., Son, H.: Users Interaction with the Hierarchically Structured Presentation in XML Document Retrieval. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 422–431. Springer, Heidelberg (2006)
- [13] Kazai, G., Trotman, A.: 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, pp. 247–260. Foundation for Information Society, Budapest (2007)
- [14] Larsen, B., Malik, S., Tombros, A.: A Comparison of Interactive and Ad-Hoc Relevance Assessments. In: Fuhr, N., Kamps, J., Lalmas, M., Trotman, A. (eds.) INEX 2007. LNCS, vol. 4862, pp. 348–358. Springer, Heidelberg (2008)
- [15] Pharos, N.: The effect of granularity and order in XML element retrieval. *Information Processing and Management* 44(5), 1732–1740 (2008)
- [16] Kuhlthau, C.C.: Seeking meaning: a process approach to library and information services, 2nd edn. Libraries Unlimited, Westport (2004)
- [17] Larsen, B., Ingwersen, P., Kekäläinen, J.: The Polyrepresentation continuum in IR. In: Ruthven, I., et al. (eds.) *Information Interaction in Context: International Symposium on Information Interaction in Context, IiX 2006*, pp. 148–162. ACM Press, New York (2006)
- [18] Fuhr, N., Klas, C.-P., Schaefer, A., Mutschke, P.: Daffodil: An integrated desktop for supporting high-level search activities in federated digital libraries. In: Agosti, M., Thanos, C. (eds.) ECDL 2002. LNCS, vol. 2458, pp. 597–612. Springer, Heidelberg (2002)

Using Eye-Tracking for the Evaluation of Interactive Information Retrieval

Thomas Beckers and Dennis Korbar

Universität Duisburg-Essen
Department of Computer Science and Applied Cognitive Science
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 Areas of Interest (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 interact with an integrated 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.

Our main research goals are to check the assumptions of interactive IR models and to find out how users interact with an integrated 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 *ezDI*[2]. Pharo et al. provide a more detailed explanation (see [3]).

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 [4]). 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

¹ Live system: <http://www.ezdl.de/>

Developer site: <http://www.is.inf.uni-due.de/projects/ezdl/>

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 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 the **query history tool**. Finally, the **task tool** shows the current working task.

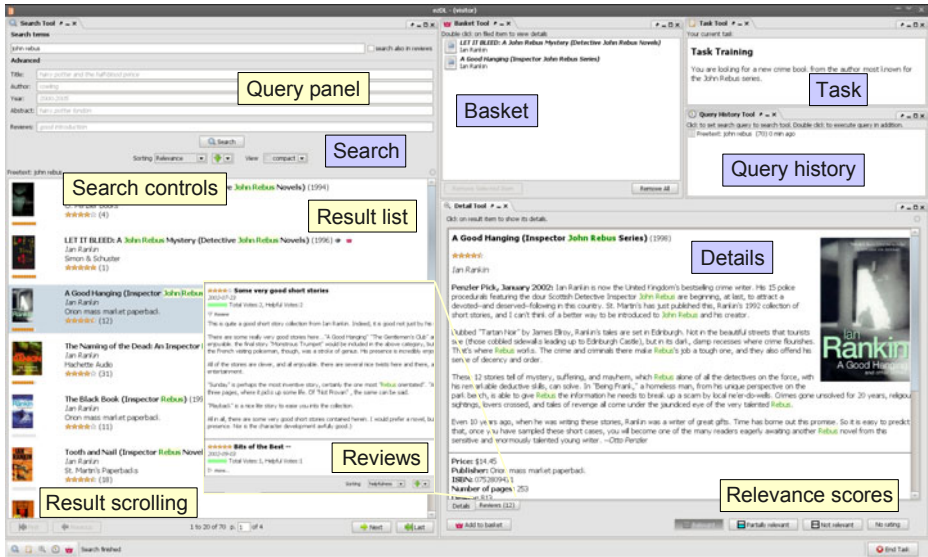


Fig. 1. The search system interface (blue boxes: tools mentioned in the description in section 2)

3 Eye-Tracking for Interactive Information Retrieval Systems

In addition to the questionnaires and system log data we also used an eye-tracking system² to record the user's eye gaze data.

The search system provides a multi-panel layout that presents a lot of information to the user. The user does not need to explicitly interact with the system

² SMI RED: <http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/red-red250-red-500.html>

to get certain information. Thus, system logs and questionnaires are not sufficient since they can only capture explicit interaction of the user with the search system (e.g. clicking on a button or selecting a tab).

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 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 it is generally not applicable in practice, especially when trying to analyze very dynamic components.

Biedert et al. capture gaze data for elements in HTML pages to provide gaze-aware text in web browsers [2]. For the *ezDL* desktop client a browser-based solution would not be applicable. 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.

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 (Interactive Information Retrieval Probability Ranking Principle)³. The IIR-PRP tries to offer decision lists and information based on

³ <http://www.is.inf.uni-due.de/projects/hiir/index.html.en>

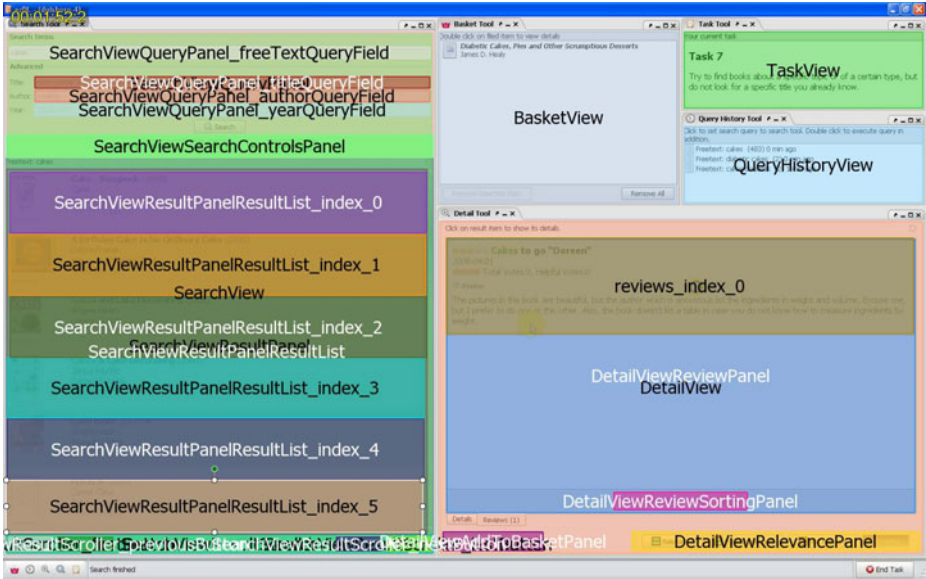


Fig. 2. Areas of Interest (screenshot from BeGaze analysis software)

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. As we stated above, the IIR-PRP tries to sort available choices by use of a cost/benefit ratio. In order to apply the model to interactive IR systems we first have to measure the user's effort for evaluating certain types of choices. By use of traditional evaluation methods, we would not be able to measure the time a user spends on evaluating a certain choice as there are always multiple choices available. By logging the time between two choices we would not be able to determine how much of this time has been spend to evaluate each of the available choices. Using an eye-tracker in conjunction with the AOI logging framework described above will enable us to measure the time a user spends on evaluating the offered choices. We will be able to measure exactly how much time a user spends gazing upon a certain available choice.

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 to 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 User Study @ iTrack 2010

For the 2010 run of the iTrack we recruited 24 participants for experiments with the additional eye-tracking support. The participation was compensated with an Amazon 12 EUR voucher. The experiments have been performed in our eye-tracking laboratory from November 2010 until December 2010. We plan to perform the analysis of the experiments later this year.

6 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: Geva, S., et al. (eds.) INEX 2010. LNCS, vol. 6932, pp. 227–235. Springer, Heidelberg (2010)
2. Biedert, R., Buscher, G., Schwarz, S., Möller, M., Dengel, A., Lottermann, T.: The Text 2.0 framework - writing web-based gaze-controlled realtime applications quickly and easily. In: Proceedings of the International Workshop on Eye Gaze in Intelligent Human Machine Interaction, EGIHMI (2010)
3. Fuhr, N.: A probability ranking principle for interactive information retrieval. *Information Retrieval* 11(3), 251–265 (2008)

Overview of the INEX 2010 Link the Wiki Track

Andrew Trotman¹, David Alexander¹, and Shlomo Geva²

¹Department of Computer Science
University of Otago
Dunedin
New Zealand

²Faculty of Science and Technology
Queensland University of Technology
Brisbane, Australia

Abstract. The INEX 2010 Link-the-Wiki track examined link-discovery in the Te Ara collection, a previously unlinked document collection. Te Ara is structured more a digital cultural history than as a set of entities. With no links and no automatic entity identification, previous Link-the-Wiki algorithms could not be used. Assessment was also necessarily manual. In total 29 runs were submitted by 2 institutes. 70 topics were assessed, but only 52 had relevant target documents and only 45 had relevant links in the pool. This suggests that the pool was not diverse enough. The best performing run had a MAP of less than 0.1 suggesting that the algorithms tested in 2010 were not very effective.

1 Introduction and Motivation

Keeping a rich hypermedia document collection such as the Wikipedia up-to-date is problematic. Each time a new document is added new links from that document into the collection are needed (and vice versa). Each time a document is deleted links to that document must be deleted. If a document changes then the links must be updated. This served as the motivation for the Link-the-Wiki track in 2007.

Deletion of a document is a simple maintenance problem – simply remove link from documents that refer to the document being deleted. Update is analogous to deletion then reinsertion. Consequently the entire link-the-wiki problem can be examined from the perspective of document insertion. In fact, this is the approach taken by the track from 2007-2009. Choose a set of documents from within the collection; remove them from the collection; then measure the efficacy of automatic link discovery algorithms on those documents as if each were a new addition to the collection. These topic-documents were referred to as orphans.

As early as 2007 two effective algorithms were seen, Geva's algorithm [1] and Itakura's algorithm [2].

Geva's algorithm builds an index of titles of documents in the collection and searches for those titles in the topic-document. The algorithm is effective in the Wikipedia because Wikipedia documents are about entities and their names are essentially canonical and if they appear in the text they are probably accurate.

Itakura’s algorithm builds an index of links in the collection and orders these on the proportion of times the anchor-text seen as a link to the number of times it is seen as a phrase anywhere in the collection. This is essentially the strength of the anchor-text as an anchor and the most likely target. The algorithm has proven problematic to implement –several implementations were seen at INEX 2009 but Trotman’s 2008 implementation remains the highest performing [3].

In 2007 the performance of algorithms was measured against the links that were in the collection before the document was orphaned. This was the same approach taken by others (for example Milne & Witten [5]). Results from evaluation in this way show extremely high performance. Otago, for example, achieved a MAP score of 0.734. Milne & Witten [5] see similar such scores using their machine learning approach.

In 2008 and 2009 TREC-style manual assessment was performed. Runs were pooled and manually assessed (to completion) for accuracy. As a twist on the experiment the links present in the Wikipedia itself were added to the pools. The evaluation showed that Geva’s algorithm, Itakura’s algorithm and the Wikipedia were performing comparably and that MAP scores in previous years had been inflated by non-relevant links present in Wikipedia articles. It is not known which links those are or why there are there, but it has been speculated that the links might themselves be put in by bots using similar algorithms to those of Milne & Witten, Geva, and Itakura.

During the running of the track the organizers became aware of an alternative link discovery scenario. The New Zealand Ministry for Culture and Heritage has an encyclopedia-like collection (Te Ara) that when complete “will be a comprehensive guide to the country’s peoples, natural environment, history, culture, economy, institutions and society”. It does not contain links between articles.

Linking Te Ara is more complex than linking the Wikipedia for many reasons. The articles are often digital narratives and in being so do not represent entities – the controlled vocabulary of the Wikipedia is not present. Geva’s title-matching algorithm is unlikely to be effective. There are no links in the collection and so the machine learning algorithms of Milne & Witten[5] and of Itakura & Clarke [2] can’t be used.

2 Te Ara Test Set

The document collection used in 2010 was the 2010 dump of Te Ara. It is a single 48MB XML file consisting of 36,715 documents. The task was to link each and every document. As such the topic set was the document collection itself.

After runs were submitted a set of documents were chosen for manual assessment. First, the collection was ordered on the number of links in each document. This was then divided into 10 deciles. Finally, one work-set was built by randomly selecting one document from each decile. Seven such (non-overlapping) work sets were assessed to completion resulting in a total of 70 assessed documents.

3 Runs

In total 29 runs were submitted by 2 institutes. QUT submitted 5 runs and Otago submitted 24 runs.

3.1 Submission Format

Results were submitted in the 2009 Link-the-Wiki Te Ara format, except that certain elements and attributes were made optional:

- The root element, `<inexltw-submission>`, had attributes for the participant's numeric ID, the run ID and the task (LTeAra).
- The `<details>` element give information about the machine on which the results were produced, and how long it took to produce them. This element was optional.
- The `<description>` gave an explanation of the linking algorithm.
- The `<collections>` element contained a list of document collections used in the run.
- Each topic was in a `<topic>` element which contained an `<anchor>` element for each anchor-text.
- One or more `<toBep>` elements, within each `<anchor>` element, gave the offset and the target document ID for the link. If the offset was specified, the target document ID was optional because the offsets were relative to the single XML file containing the collection. If no offset was specified it was assumed to be the start of the document.
- Each topic could contain up-to 50 anchors, and each anchor could contain up-to 5 BEPs (each in different target documents).

Example

An example of a submission is:

```
<inexltw-submission participant-id="12"
  run-id="Otago_LTeAraA2B_01"
  task="LTeAra">
  <details>
  <machine>
  <cpu>Intel Celeron</cpu>
  <speed>1.06GHz</speed>
  <cores>1</cores>
  <hyperthreads>1</hyperthreads>
  <memory>128MB</memory>
  </machine>
  <time>3.04 seconds</time>
  </details>
  <description>
  Describe the approach here, NOT in the run-id.
  </description>
  <collections>
  <collection>TeAra_2010_Collection</collection>
  </collections>
  <topic file="9638" name="Matariki - Maori New Year">
  <outgoing>
```

```

<anchor offset="7445748" length="8" name="balloons">
<tobep offset="7952293">10151</tobep>
<tobep offset="10553520">12991</tobep>
<tobep offset="11686141">14270</tobep>
<tobep offset="8016276">10208</tobep>
<tobep offset="7226359">9363</tobep>
</anchor>
</outgoing>
</topic>
</inexltw-submission>

```

DTD

The DTD for the submission format was:

```

<!ELEMENT inexltw-submission (details, description,
collections, topic+)>
<!ATTLIST inexltw-submission
  participant-id CDATA #REQUIRED
  run-id CDATA #REQUIRED
  task (LTara_A2B) #REQUIRED>

<!ELEMENT details (machine|time)>

<!ELEMENT machine
(cpu|speed|cores|hyperthreads|memory)>
<!ELEMENT cpu (#PCDATA)>
<!ELEMENT speed (#PCDATA)>
<!ELEMENT cores (#PCDATA)>
<!ELEMENT hyperthreads (#PCDATA)>
<!ELEMENT memory (#PCDATA)>

<!ELEMENT time (#PCDATA)>

<!ELEMENT description (#PCDATA)>

<!ELEMENT collections (collection+)>
<!ELEMENT collection (#PCDATA)>

<!ELEMENT topic (outgoing|anchor+)>
<!ATTLIST topic
  file CDATA #REQUIRED
  name CDATA #IMPLIED>

```



```

<!ELEMENT outgoing (anchor+)>
<!ELEMENT anchor (tobep+)>
<!ATTLIST anchor
  name CDATA #IMPLIED
  offset CDATA #REQUIRED
  length CDATA #REQUIRED>

<!ELEMENT tobep (#PCDATA)>
<!ATTLIST tobep
  offset CDATA #REQUIRED>

```

4 Assessment Tool

A new cross-platform assessment tool was written in C/C++ using SQLite and GTK+. This tool was written in an effort to reduce the assessment time and this increase in the number of topics that could be assessed in a “reasonable” period of time. Of course, the assessment of “incoming” links was not necessary in 2010 as the entire collection was linked.

A screenshot of the 2010 assessment tools is shown in Figure 1. Brief assessing instructions are given in the top window. On the left is source document with the current anchor highlighted in yellow. On the right at top is the list of links from the pool with assessed-relevant links marked in green and assessed-non-relevant links

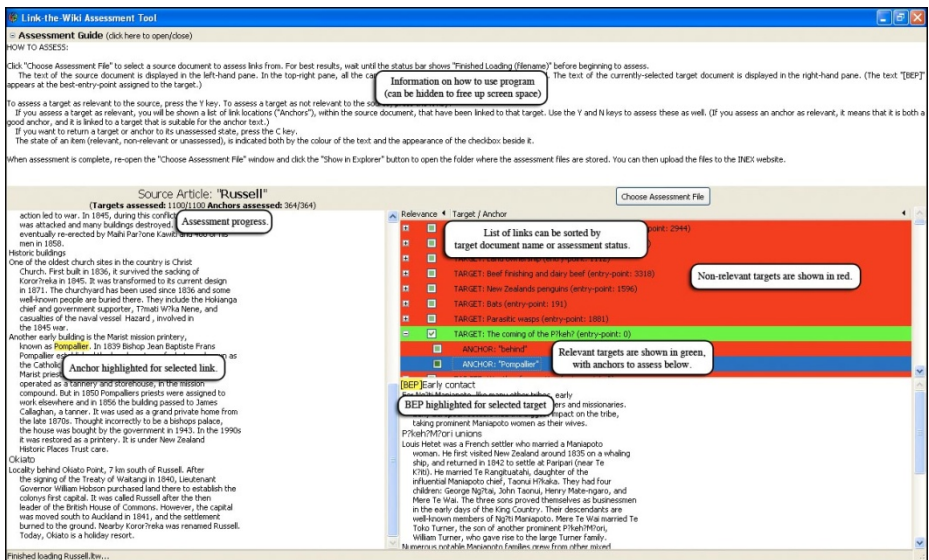


Fig. 1. 2010 Link-the-Wiki assessment tool

marked in red (un-assessed had a white background). On the right at the bottom is the target document with the best-entry-point (BEP) marked in yellow.

Several changes were made to the assessment method. First, the assessor is shown the target document (and BEP). If this was not relevant then all anchors to that document must be non-relevant. This made it possible to assess several links in one click. If the target document was relevant then the assessor was shown all anchors for that document and asked to assess which were relevant. In this way the assessor was simply choosing whether the given anchor was an accurate way to link the two documents.

The 2010 assessment tool does not ask the assessor to choose a BEP in the case where the document is relevant but the BEP was badly placed. There were several reasons for this decision, the strongest of which was that the results from the INEX ad hoc track BEP experiments show that BEPs are usually at (or very close to) the start of the given document (311.5 characters in 2009 [4]); but also because the pool was (supposed to be) assessed to completion and so the usefulness of the BEP could be determined by the assessor.

5 Metrics

As is the informal convention at INEX, the metrics for the Link-the-Wiki track in 2010 were not published before the runs were submitted. As is also the informal convention, the metric changed in 2010.

In a Link-the-Wiki run it is possible (and correct) to identify more than one anchor targeting the same document. It is also possible and correct to identify more than one target per anchor. Consequently metrics based on recall (such as un-interpolated Mean Average Precision (MAP)) are meaningless. If there is only one relevant target document, but the link-discovery algorithm identifies two different anchors for that target then what is the recall? Exactly this happens in this very document, the reference to Itakura's algorithm and to the paper by Itakura & Clarke are different anchors for the same document. This also happens in the submitted runs. The runs were, consequently, de-duplicated by taking on the highest ranking instance of a target for a given topic and ignoring other instances of the target.

The relevance of each target was then determined using the manual assessments. Then the score for each position in the results list was 1 if any target for that anchor was relevant and 0 otherwise. Mean un-interpolated Average Precision is then computed from this.

This approach gives a highly optimistic evaluation because the run has 5 trials at each point in the results list and if any one trial is correct the run scores a relevant hit. It is also optimistic because the anchor and BEP are not considered (if multiple BEPs are seen in the assessments then if any-one is relevant the document is relevant). It also guarantees that recall cannot exceed 1.

6 Results

In total 70 topics were assessed. 18 of these had no relevant target documents in the pool. The mean number of relevant targets per topic was 8.8 and the mean number of non-relevant targets per topic was 274.6. Topic 2919 had the most relevant targets (97). Figure 2 shows the distribution of the number of relevant targets documents per topic (solid line) and the number of relevant anchor-target pairs in the pool (dotted) ordered from most relevant targets to least. In some cases the number of relevant targets far exceeds the number of relevant links (for example, topic 25591 has 27 relevant targets but only one relevant anchor-target pair was in the pool). In cases where the number of relevant anchor-target pairs exceeds the number of relevant target documents (such as topic 15765) the assessor has identified more than one anchor as relevant to the same target document (in this case 66 relevant links to 62 relevant documents). On average there are 5.5 relevant links per document and only 11 topics have more 10 relevant links in them.

Figure 3 shows the performance of all the runs submitted to the track. The best Otago run was Otago_LTeAraA2B_11 with a MAP of 0.0906. The best QUT run was QUT_LTeAraA2B_DH3 with a MAP of 0.0863. In the figure the MAP of each run is shown in brackets after the run name. As can be seen, the performance of the runs is not comparable to the high scores seen in the link-the-wiki track in previous years. Early precision is low and drops quickly – recall also that the evaluation is overly optimistic because the anchors are not evaluated (it is equivalent to previous year’s file-to-file evaluation) and that the score at each point in the results list is the best of the 5 possible targets seen there.

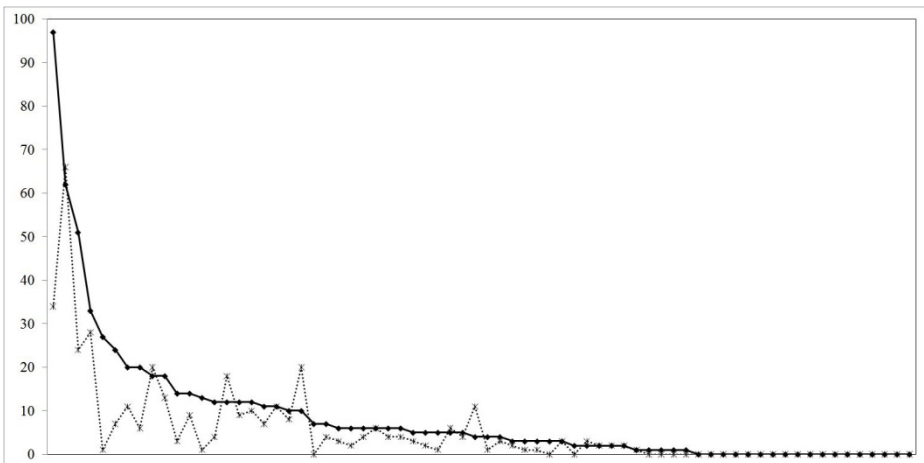


Fig. 2. Relevant targets per topic

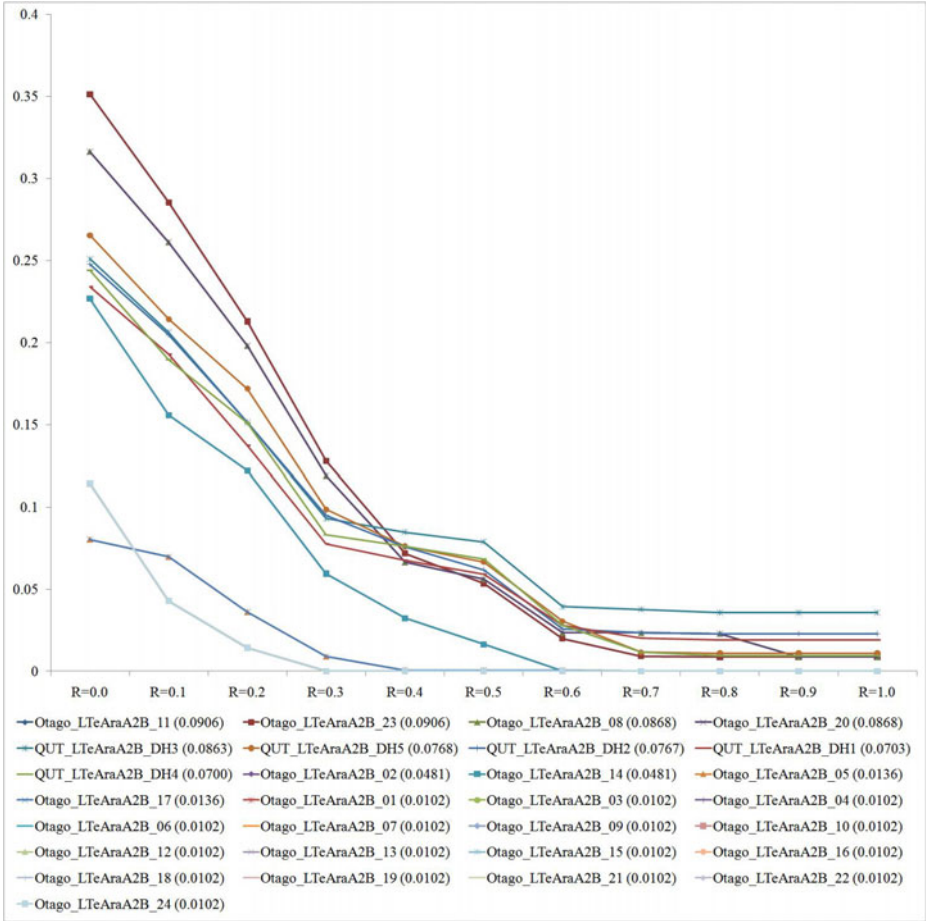


Fig. 3. Precision / Recall of all submitted runs

7 Conclusions and Further Work

The INEX 2010 Link-the-Wiki track changed document collection from the Wikipedia to Te Ara. Te Ara is a far more difficult document collection because there are no prior links (so Itakura’s algorithms cannot be used), and the document titles are not canonical entity names (so Geva’s algorithm cannot be used). As a consequence of the increased difficulty the number of groups participating dropped to just two (Otago and QUT).

In total 70 topics were manually assessed, but of those the assessors found only 52 with relevant target documents and only 45 with relevant links in the pool. This suggests that both the pool was not diverse enough and that the algorithms tested were not effective enough. This is echoed in the evaluation where the best scoring optimistic MAP is less than 0.1 and the runs perform poorly at all recall points.

Further work on Link-the-Wiki has already started at NTCIR with the crosslink track. That track is using a multi-lingual dump of the Wikipedia to examine algorithms that discover links from orphans in one language to targets in another language. This interested in further link-discovery research are referred to NTCIR.

References

- [1] Geva, S.: GPX: Ad-Hoc Queries and Automated Link Discovery in the Wikipedia. In: Focused Access to XML Documents, pp. 404–416. Springer, Heidelberg (2007)
- [2] Itakura, K.Y., Clarke, C.L.: University of Waterloo at INEX2007: Adhoc and Link-the-Wiki Tracks. In: Focused Access to XML Document, pp. 417–425. Springer, Heidelberg (2007)
- [3] Jenkinson, D., Leung, K.-C., Trotman, A.: Wikisearching and Wikilinking. In: Advances in Focused Retrieval, pp. 374–388. Springer, Heidelberg (2009)
- [4] Kamps, J., Geva, S., Trotman, A.: Analysis of the INEX 2009 ad hoc track results. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 26–48. Springer, Heidelberg (2010)
- [5] Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceeding of the 17th ACM Conference on Information and Knowledge Management, pp. 509–518. ACM, Napa Valley (2008)

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 discuss the effect of various stemming algorithms. In the Efficiency Track, we compare three query pruning algorithms and discuss 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 Ad 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 discuss 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).

Table 1. S Stripper rules. The first suffix matched on the left is replaced by the suffix on the right.

ies	→	y
es	→	
s	→	

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 a 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:

$$RSV_d = \sum_{t \in q} \log \left(\frac{N}{df_t} \right) \cdot \frac{(k_1 + 1) t f_{td}}{k_1 \left((1 - b) + b \times \left(\frac{L_d}{L_{avg}} \right) \right) + t f_{td}}$$

here, N is the total number of documents, and df_t and $t f_{td}$ 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.

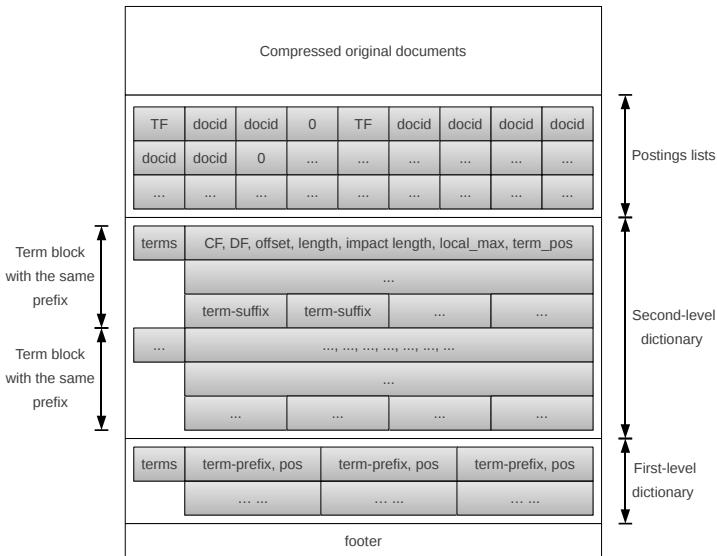


Fig. 1. The index structures

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

This section details the results of assessing the 24 runs described in Section 4.3.

Figure 2 shows the mean average precisions for the BEPs produced by each run. Precision/Recall graphs are included in the track overview paper.

All the full document contexts runs outperformed the paragraph context runs. This result suggests that context is important when predicting links for Te Ara, and a generalisation of the result that context matters in Focused Retrieval in general.

The difference squared method for summation always worked better than the sum method, and the single relevance context worked best (in that order). This suggests that although context is important in identifying links, the best link to use is determined by using just one context.

The best target context to use is the heading, followed by heading and section, then just section. This results suggests that headings are important for identifying targets – something that was show to be the case with the Wikipedia link-the-wiki.

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.

Run	Context	Summation	Contribution	Element	MAP
1	Article	Diff	Single	Heading	0.0906
2	Article	Diff	Single	Both	0.0906
3	Article	Diff	Single	Section	0.0868
4	Article	Diff	Average	Heading	0.0868
5	Article	Diff	Average	Both	0.0863
6	Article	Diff	Average	Section	0.0768
7	Article	Sum	Single	Heading	0.0767
8	Article	Sum	Single	Both	0.0703
9	Article	Sum	Single	Section	0.0700
10	Article	Sum	Average	Heading	0.0481
11	Article	Sum	Average	Both	0.0481
12	Article	Sum	Average	Section	0.0136
13	Paragraph	Diff	Single	Heading	0.0136
14	Paragraph	Diff	Single	Both	0.0102
15	Paragraph	Diff	Single	Section	0.0102
16	Paragraph	Diff	Average	Heading	0.0102
17	Paragraph	Diff	Average	Both	0.0102
18	Paragraph	Sum	Average	Section	0.0102
19	Paragraph	Sum	Single	Heading	0.0102
20	Paragraph	Sum	Single	Both	0.0102
21	Paragraph	Sum	Single	Section	0.0102
22	Paragraph	Sum	Average	Heading	0.0102
23	Paragraph	Sum	Average	Both	0.0102
24	Paragraph	Sum	Average	Section	0.0102

Fig. 2. Results of the Otago runs in INEX 2010 Link-the-Wiki

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 3 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

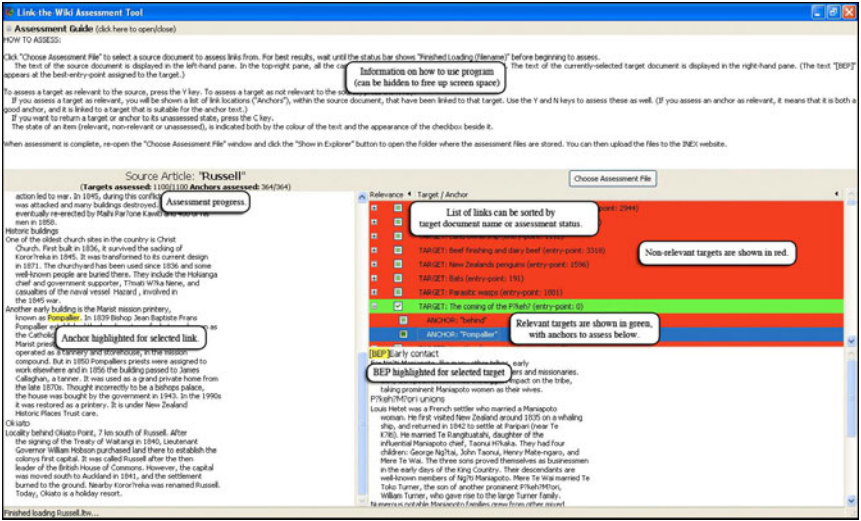


Fig. 3. An annotated screenshot of the 2010 assessment tool

“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.

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

Table 2. The Otago Stemmer. Rule sections are separated by lines.

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	
<hr/>		
4	rs	tie
1	s	
1	f	uow
0	f	uow
0	q	
1	s	
<hr/>		
2	que	sy
0	sl	anu
2	e	
1	f	
3	ague	dz
0	ean	

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.

Table 3. Stemming runs

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

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.

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 conducted 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].

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-Stripping for stemming.

(a)		(b)	
Collection Size	50.7 GB	Collection Size	50.7 GB
Documents	2666190	Documents	2666190
Avg Document Length	880 words	Avg Document Length	880 words
Unique Words	11437080	Unique Words	11186163
Total Worlds	2347132312	Total Worlds	2347132312
Postings Size	1.2 GB	Postings Size	1.5 GB
Dictionary Size	399 MB	Dictionary Size	390 MB

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.

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 4 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.

Table 5. The lists of run IDs and the associated lower-k and upper-K values

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

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*.

The MAiP measures of the original *topk* are quite different from the other two algorithms. Using term frequencies as impact values have better MAiP measures when the values of lower-k and upper-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 5. 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, 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

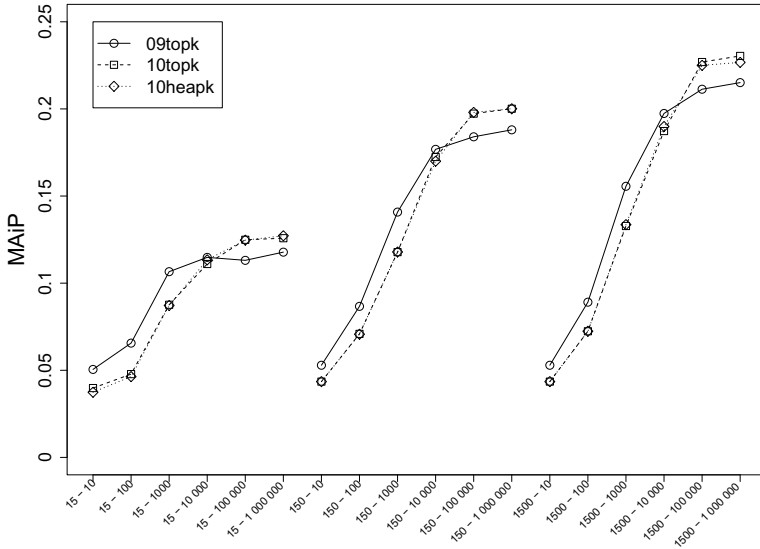


Fig. 4. MAiP measures for the original *topk*, improved *topk* and *heapk*

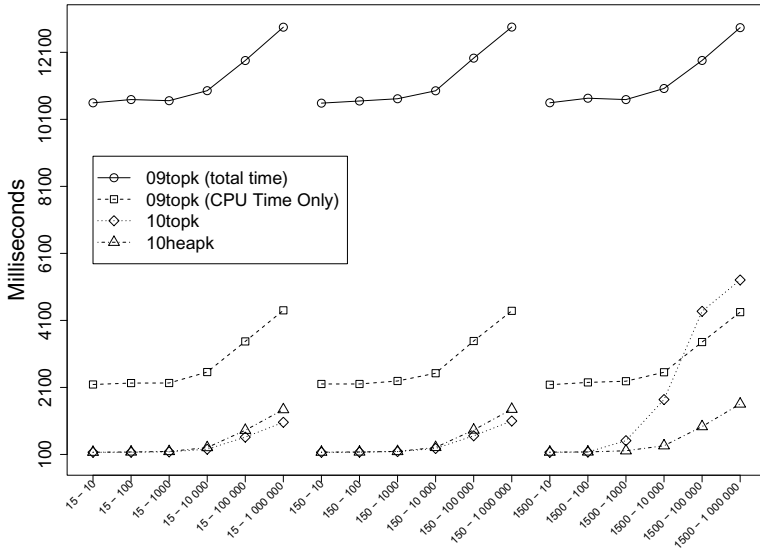


Fig. 5. Efficiency comparison

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. With the ranking shown as (position / total runs), the results suggest that BM25 is a better ranking function than Divergence from Randomness for this collection, it consistently performed better regardless of the measure. They also suggest that BM25 whole document ranking is effective with our best run consistently in the top 6 regardless of how it is measured. We believe that, as is already the case in the ad hoc track, BM25 document ranking should be used as a baseline in future years in the document centric track.

Table 6. Effectiveness measure for the Data Centric Track

Run ID	MAGP	MAiP	MAP
DC-BM25	0.2491 (#1/14)	0.1550 (#6/29)	0.3397 (#5/29)
DC-DIVERGENCE	0.1561 (#5/14)	0.1011 (#3/29)	0.2103 (#14/29)

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.) INEX 2007. LNCS, vol. 4862, pp. 373–387. Springer, Heidelberg (2008)
2. Geva, S.: Gpx: Ad-hoc queries and automated link discovery in the wikipedia. In: Fuhr, N., Kamps, J., Lalmas, M., Trotman, A. (eds.) INEX 2007. LNCS, vol. 4862, pp. 404–416. Springer, Heidelberg (2008)
3. Porter, M.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
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), 61–81 (1998)
6. Jia, X.F., Trotman, A., O’Keefe, R.: Efficient accumulator initialisation. In: Proceedings of the 15th Australasian Document Computing Symposium (ADCS 2010), Melbourne, Australia (2010)
7. Trotman, A.: Compressing inverted files. Inf. Retr. 6(1), 5–19 (2003)
8. Anh, V.N., Moffat, A.: Inverted index compression using word-aligned binary codes. Inf. Retr. 8(1), 151–166 (2005)
9. Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Plachouras, V., Silvestri, F.: The impact of caching on search engines. In: SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 183–190. ACM, New York (2007)

10. Jia, X.F., Trotman, A., O'Keefe, R., Huang, Z.: Application-specific disk I/O optimisation for a search engine. In: PDCAT 2008: Proceedings of the 2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 399–404. IEEE Computer Society, Washington, DC (2008)
11. Buckley, C., Lewit, A.F.: Optimization of inverted vector searches, pp. 97–110 (1985)
12. Moffat, A., Zobel, J.: Self-indexing inverted files for fast text retrieval. *ACM Trans. Inf. Syst.* 14(4), 349–379 (1996)
13. Tsegay, Y., Turpin, A., Zobel, J.: Dynamic index pruning for effective caching, pp. 987–990 (2007)
14. Persin, M., Zobel, J., Sacks-Davis, R.: Filtered document retrieval with frequency-sorted indexes. *J. Am. Soc. Inf. Sci.* 47(10), 749–764 (1996)
15. Anh, V.N., de Kretser, O., Moffat, A.: Vector-space ranking with effective early termination, pp. 35–42 (2001)
16. Trotman, A., Jia, X.F., Geva, S.: Fast and effective focused retrieval. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 229–241. Springer, Heidelberg (2010)
17. Bentley, J.L., Mcilroy, M.D.: Engineering a sort function (1993)
18. Persin, M.: Document filtering for fast ranking, pp. 339–348 (1994)
19. Moffat, A., Zobel, J., Sacks-Davis, R.: Memory efficient ranking. *Inf. Process. Manage.* 30(6), 733–744 (1994)
20. Moffat, A., Zobel, J., Klein, S.T.: Improved inverted file processing for large text databases, pp. 162–171 (1995)
21. Anh, V.N., Moffat, A.: Random access compressed inverted files. In: Australian Computer Science Comm.: Proc. 9th Australasian Database Conf. ADC, vol. 20(2), pp. 1–12 (February 1998)
22. Anh, V.N., Moffat, A.: Compressed inverted files with reduced decoding overheads, pp. 290–297 (1998)
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), 357–389 (2002)

Overview of the INEX 2010 Question Answering Track (QA@INEX)

Eric SanJuan¹, Patrice Bellot¹, Véronique Moriceau², and Xavier Tannier²

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 Question Answering track (QA@INEX) 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.

Long answers have been evaluated based on Kullback Leibler (KL) divergence between n-gram distributions. This allowed summarization systems to participate. Most of them generated a readable extract of sentences from top ranked documents by a state-of-the-art document retrieval engine. Participants also tested several methods of question disambiguation.

Evaluation has been carried out on a pool of real questions from OverBlog and Yahoo! Answers. Results tend to show that the baseline-restricted focused IR system minimizes KL divergence but misses readability meanwhile summarization systems tend to use longer and stand-alone sentences thus improving readability but increasing KL divergence.

1 Introduction

The INEX QA 2009-2010 track [1] aimed to compare the performance of QA, XML/passage retrieval and automatic summarization systems on special XML enriched dumps of the Wikipedia : the 2008 annotated Wikipedia [2] used in the INEX ad-hoc track in 2009 and 2010.

Two types of questions were considered. The first type was factual questions which require a single precise answer to be found in the corpus if it exists. The second type consisted of more complex questions whose answers required a multi-document aggregation of passages with a maximum of 500 words exclusively.

Like for the 2010 *ad-hoc restricted focus task*, systems had to make a selection of the most relevant information, the maximal length of the abstract being fixed. Therefore focused IR systems could just return their top ranked passages meanwhile automatic summarization systems need to be combined with a document IR engine. The main difference between the QA long type answer task and the *ad-hoc restricted focus* one is that in QA, readability of answers [3] is as important as the informative content. Both need to be evaluated. Therefore answers

cannot be any passage of the corpus, but at least well formed sentences. As a consequence, informative content of answers cannot be evaluated using standard IR measures since QA and automatic summarization systems do not try to find all relevant passages, but to select those that could provide a comprehensive answer. Several metrics have been defined and experimented with at DUC [4] and TAC workshops [5]. Among them, Kullback-Leibler (KL) and Jentsen-Shanon (JS) divergences have been used [6] to evaluate the informativeness of short summaries based on a bunch of highly relevant documents. In this edition we used the KL one to evaluate the informative content of the long answers by comparing their n-gram distributions with those from 4 highly relevant Wikipedia pages.

In 2009 a set of encyclopedic questions about ad-hoc topics was released [1]. The idea was that informativeness of answers of encyclopedic questions could be evaluated based on the ad-hoc qrels [7]. This year, a set of “real” questions from *Over-Blog* [8] website logs not necessarily meant for the Wikipedia was proposed. A state of the art IR engine powered by Indri was also made available to participants. It allowed the participation of seven summarization systems for the first time at INEX. These systems only considered long type answers and have been evaluated on the 2010 subset. Only two standard QA systems participated to the factual question sub-track. Therefore most of QA@INEX 2010 results are about summarization systems versus a state of the art restricted focused IR system.

The rest of the paper is organized as follows. First, the focused IR system used as baseline is introduced in Section 2. It is described in (§2.1) and evaluated in (§2.2). Section 3 details the collection of questions (§3.1-3.3) and available reference texts for assessments (§3.5). Section 4 explains the final choice of metrics used for the evaluation of the informativeness of long answers after several experiments. Results are reported in Section 5. Finally, Section 6 discusses our findings and draws perspectives for next year edition.

2 Baseline System: Restricted Focused IR System

Several on-line resources have been made available to facilitate participation and experiment the metrics. These resources available *via* a unique web interface at <http://termwatch.es/Term2IR> included:

1. a document index powered by Indri,
2. a sentence and Part of Speech tagger powered by the TreeTagger,
3. a summarization and Multi-Word Term extractor powered by TermWatch,
4. a tool for automatic evaluation of summary informativeness powered by FRESA,
5. links to document source on the TopX web interface.

2.1 Features

The system allows to test on the INEX 2009 ad-hoc corpus the combination of a simple IR passage retrieval system (Indri Language Model) with a baseline summarization system (a fast approximation of Lexrank).

¹ <http://www.over-blog.com/>

Different outputs are available. The default is a selection of relevant sentences with a link towards the source document in TopX. Sentences have been selected following approximated LexRank scores among the 20 top ranked passages returned by Indri using a Language Model over INEX 2008 corpus. Multiword terms extracted by shallow parsing are also highlighted.

A second possible output gives a baseline summary with less than 500 words, made of the top ranked sentences. The Kullback-Leibler divergence between distributions of n-grams in the summary and in the passages retrieved by Indri are also shown. They are computed using the FRESA package. It is also possible to test any summary against this baseline.

Finally, the passages retrieved by Indri are available, in several formats: raw results in native INEX XML format, raw text, POS tagged text with TreeTagger.

Questions and queries can be submitted in plain text or in Indri language. The following XML tags have been indexed and can be used in the query: *b*, *bdy*, *category*, *causal_agent*, *country*, *entry*, *group*, *image*, *it*, *list*, *location*, *p*, *person*, *physical_entity*, *sec*, *software*, *table*, *title*. These are examples of correct queries:

- Who is Charlie in the chocolate factory?
- #1(Miles davis) #1(Charles Mingus) collaboration
- #1(Charles Mingus).p, #combine[p](Charles Mingus)

2.2 Evaluation on the 2010 *Restricted Focus Ad-Hoc Task*

Let us first give some details on this restricted focus system.

As stated before it starts by retrieving n documents using an Indri language model. These sentences are then segmented into sentences using shallow parsing. Finally sentences are ranked using a fast approximation of LexRank. Basically, we only consider sentences that are at distance two from the query in the intersection graph of sentences. These are sentences that share at least one term with the query, or with another sentence that shares it. The selected sentences are then ranked by entropy.

We evaluated this baseline system on the Ad-hoc restricted focused task, by setting $n = 100$. We then retrieve for each sentence all passages in which the same word sequence appears, with possible insertions. We return the first 1000 characters.

The precision/recall function of this system starts high compared to other participant runs. It gets among automatic runs, the third char precision (0.3434) and the best iP[0.01] with a value of 0.15 (0.1822 for the best manual run).

3 Sets of Questions and References

A total set of 345 questions has been made available. There are four categories of questions:

1. factual and related to 2009 ad-hoc topics (151),
2. complex and related to 2009 ad-hoc topics (85),

3. factual from Over-Blog logs (44),
4. complex from Over-Blog logs (70)

This year evaluation has been carried out on the fourth category. Answers for the first category are available. A run is also available for categories 1 and 3. Informativeness of answers to questions in category 3 can be partially evaluated based on qrel from ad-hoc 2009 INEX track.

3.1 Encyclopedic vs. General Questions

236 questions are related to 2009 INEX ad-hoc topics. Most of the remaining questions come from a sample of the log files from the search engine on Over-Blog. These are real questions submitted to their website by visitors looking for answers among the blogs hosted on 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 website².

We considered three different types of questions: `short_single`, `short_multiple` and `long`.

3.2 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 had to 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 had to be self-contained to decide if the answer is correct or not.

Besides, we collected manually answers for the first category (factual and related to 2009 ad-hoc topics) in the Wikipedia INEX collection. These answers will be made available as a development set for 2011 campaign. Moreover, a sample run is available for all factual questions. This run has been produced by FIDJI, an open-domain QA system for French and English [8]. This system combines syntactic information with traditional QA techniques such as named entity recognition and term weighting in order to validate answers through different documents. Question analysis in FIDJI turns the question into a declarative sentence. It also aims to identify the syntactic dependencies, the expected type(s) of the answer (named entity type) and the question type (factoid, definition, complex, list questions).

² <http://answers.yahoo.com/>

3.3 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. Are considered as words any sequence of letters and digits. An example of a long type question is (#196): *What sort of health benefit has olive oil?* There can be questions of both short and long types, for example a question like *Who was Alfred Nobel?* can be answered by “a chemist” or by a short biography. However, most of the selected long type questions are not associated with obvious name entities and require at least one sentence to be answered.

3.4 Format Submission

The submission format has been simplified to follow INEX TREC ad-hoc format:

```
<qid> Q0 <file> <rank> <rsv> <run_id> <column_7> <column_8> <column_9>
```

where:

- the first column is the topic number.
- the second column 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 `jidi` of the Wikipedia document.
- the fourth column is the rank the result is retrieved, and fifth column shows the score (integer or floating point) that generated the ranking.
- the sixth column is called the “run tag” and should be a unique identifier for the group AND for the method used.

The remaining three columns depend on the question type (short or long) and on the chosen format (text passage or offset).

For textual content, raw text is given without XML tags and without formatting characters. The resulting word sequence has to appear in the file indicated in the third field. This is an example of such output:

```
1 Q0 3005204 1 0.9999 I10Run1 The Alfred [...] societies.
1 Q0 3005204 2 0.9998 I10Run1 The prize [...] Engineers.
1 Q0 3005204 3 0.9997 I10Run1 It has [...] similar spellings.
```

An Offset Length format (FOL) can also be used. In this format, passages are given as offset and length calculated in characters with respect to the textual content (ignoring all tags) of the XML file. File offsets start counting from 0 (zero). Previous example would be the following in FOL format:

```
1 Q0 3005204 1 0.9999 I10Run1 256 230
1 Q0 3005204 2 0.9998 I10Run1 488 118
1 Q0 3005204 3 0.9997 I10Run1 609 109
```

This would mean that results are from article 3005204. The first passage starts at the 256th character (so 257 characters beyond the first character), and has a length of 230 characters.

In the case of short type questions, we use an extra field that indicates the position of the answer in the passage. This position is given by counting the number of words before the detected answer.

3.5 Reference Texts

For each question we have selected four highly relevant Wikipedia pages from which we have extracted the most relevant sections. Questions for which there were too few relevant passages were not submitted to participants. These passages that were not publicly available have been then used as reference text to evaluate long type answers using KL divergence.

4 Evaluation of Long Answers

Only long answer evaluation is presented. As short answer runs come from organizers' systems, we decided not to evaluate them but they will be made available for future participants.

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 [6] 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 [6], 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 reference texts or passages.

We used the FRESA package³ 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

³ <http://lia.univ-avignon.fr/fileadmin/axes/TALNE/Ressources.html>

Table 1. Cumulative KL divergence for best runs per participant

ID	Specificity	unigrams	bigrams	4 skip grams	average	readability
98	Focused IR	1599.29	2207.56	2212.49	2006.45	1/5
92	MWT expansion	1617.35	2226.61	2231.56	2025.17	2/5
860	System combination	1617.6	2227.37	2232.43	2025.8	3/5
857	Question reformulation	1621.14	2234.57	2239.85	2031.85	3/5
855	Semantic expansion	1625.76	2235.21	2240.35	2033.77	3/5
943	Long sentences	1642.93	2252.28	2257.22	2050.81	4/5
557	JS minimization	1631.29	2237.61	2242.83	2037.24	3/5

INEX ad-hoc task. It requires a human evaluation by organizers and participants where the assessor indicates where he misses the point of the answers because of highly incoherent grammatical structures, unsolved anaphora, or redundant passages.

5 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 webpage⁴.

Table 1 shows results based on KL divergence on long-type questions from OverBlog logs. The cumulative divergence is the sum of KL scores between participant extracts and selected pages.

As expected, baseline-restricted focused IR system (98) minimizes KL divergence but the resulting readability is poor. Meanwhile the system (943) having best readability favors long sentences and gets highest divergence figures. 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. Meanwhile this is a system slightly adapted from the focused IR system used in previous INEX 2008 and 2009 ad-hoc track [9,10].

Surprisingly sentence JS minimization (557) does not seem to minimize overall KL divergence. This system ranks sentences in decreasing order according to their JS divergence with the query and the retrieved documents.

Only score differences between the baseline and the other systems are significant, as shown in Table 2.

⁴ <http://qa.termwatch.es/>

Table 2. Probabilities of signed t-tests over KL divergence scores

ID	92	860	857	855	943	557
98	* 0.0400	* 0.0149	** 0.0028	* 0.0172	**** 0.0005	*** 0.0000
92		0.3777	0.1037	0.2304	0.0821	0.0529
860			0.1722	0.4442	0.1497	0.1104
857				0.2794	0.5047	0.1788
943					0.1798	0.1013
557						0.1857

The standard deviation among systems KL divergences varies. The ten question minimizing standard deviation and, therefore, getting most similar answers among systems are:

- 2010044** What happened to the president of Rwanda death?
- 2010107** What are the symptoms of a tick bite?
- 2010096** How to make rebellious teenager obey you?
- 2010066** How much sadness is normal breakup?
- 2010062** How much is a typical sushi meal in japan?
- 2010083** What are the Refugee Camps in DRC?
- 2010046** How to get Aljazera sports?
- 2010047** How to be a chef consultant?
- 2010005** Why did Ronaldinho signed for Barcelona?
- 2010049** Where can I find gold sequined Christain Louboutin shoes?

All these questions contain at least one named entity that refers to a Wikipedia page. Therefore, the systems mostly built their answer based on the textual content of this page and KL divergence is not accurate enough to discriminate among them.

On the contrary, the 10 following questions are the top ten that maximized standard deviation and have the greatest impact in the ranking of the systems:

- 2010093** Why is strategy so important today?
- 2010114** What is epigenetics and how does it affect the DNA/genes in all of our cells?
- 2010009** What does ruddy complexion mean?
- 2010066** What do nice breasts look like?
- 2010022** How to get over soul shock?
- 2010092** How to have better sex with your partner?
- 2010080** How to be physically attractive and classy?
- 2010014** Why is it so difficult to move an mpeg into imovie?
- 2010010** What do male plants look like?
- 2010075** WHAT IS A DUAL XD ENGINE?

Clearly, these questions are not encyclopedic ones and do not refer to particular Wikipedia pages. Meanwhile partial answers exist in the Wikipedia but they are spread among several articles.

Figure 1 shows the KL divergence values for 4-skip n-grams over the most discriminative questions. It can be observed that cumulative KL divergence varies along questions. These variations reveal the gaps between reference documents and textual content extracted by participant systems.

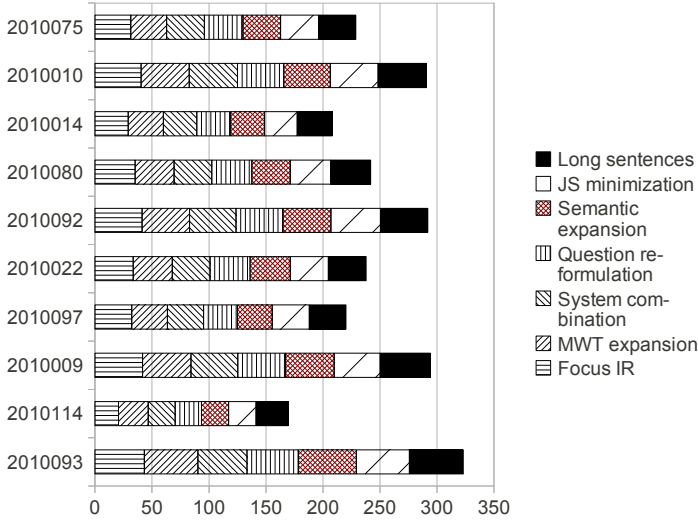


Fig. 1. KL divergence per system over the most discriminative questions

Figure 2 shows the same values but scaled in order to reveal main differences between systems.

6 Plans for Next QA Track 2011 Edition

In 2011, we will keep on addressing real-world focused information needs formulated as natural language questions using special XML annotated dumps of the Wikipedia, but we will mix questions requiring long answers and those not.

6.1 Merging Short and Long Answer Task

Contrary to long answer task, we had too few participant teams interested in short answer evaluation in 2010. We think that this is mainly due to the fact that this task required too many modifications to traditional INEX participant systems.

To avoid this problem, we plan to make the task more manageable for questions where short answers are expected.

Consequently, potential short answers will be tagged in the collection and considered as traditional XML retrieval units. The provided collection will thus be enriched with named entity annotations. These entities are generally the most

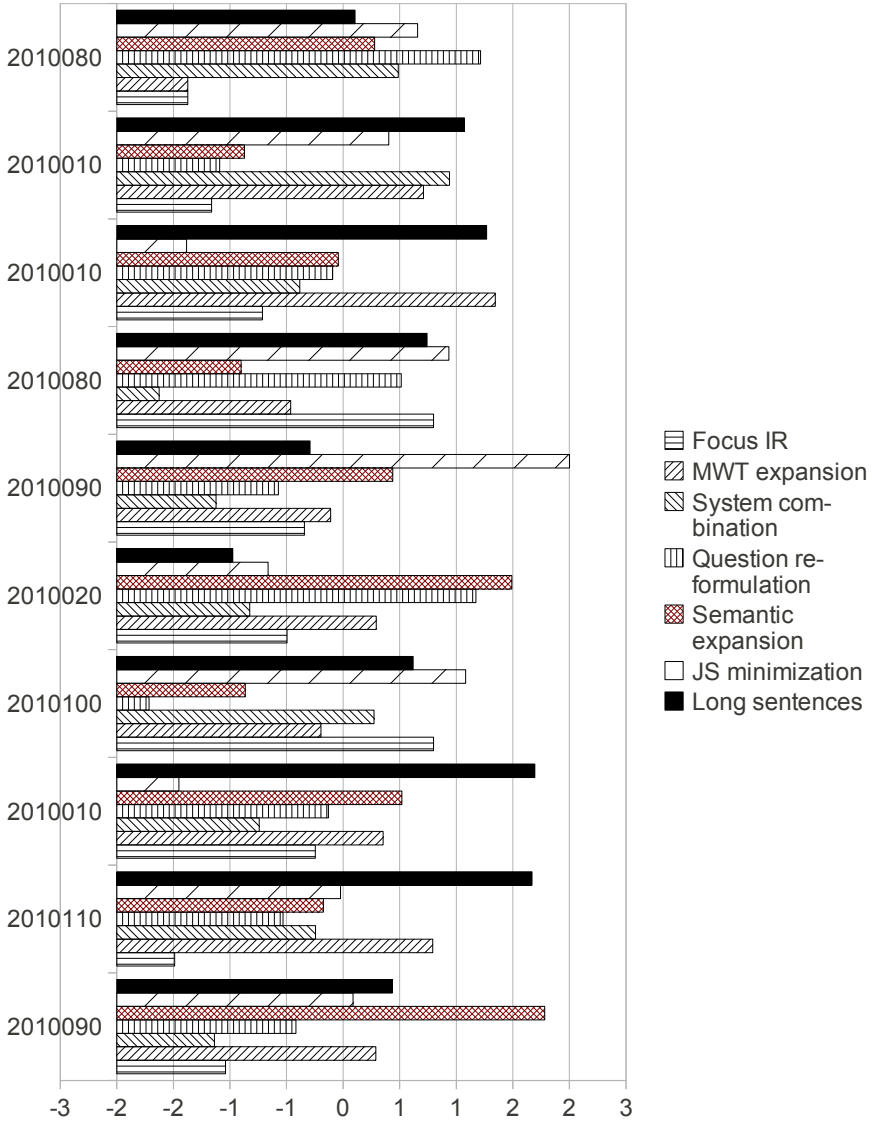


Fig. 2. scaled KL divergence per system over the most discriminative questions

relevant for short answers. However, the type of entities (persons, locations, organizations, etc.) will not be provided.

Moreover, the information concerning the short/long type of expected answers will be removed from the input question format. Given a question, the decision whether the answer should be rather short or long is then left to the system.

Concerning the evaluation, many different answers can be considered as relevant, and a manual assessment is necessary. As a short answer is now an XML tag, the global methodology should not differ from long answer task.

Finally, by restricting output to XML elements and not any passage, we shall improve readability. Indeed, we have observed that a significant number of readability issues were due to incorrect sentence segmentation or mixed document structure elements like lists, tables and titles.

6.2 Towards Contextualization of Questions

In 2011, we shall build and make available a new dump of the Wikipedia adapted for QA where only document structure, sentences and named entities will be annotated. For each question, participants will have to guess the context category (for example, "expert", "pupil", "journalist"...), provide a context as a query oriented readable summary and a list of possible answers. Summary and answers will be made of XML elements from Wikipedia dump. Real questions will be selected from OverBlog⁵ website logs and Twitter⁶. Participants system will have to first detect the research context. Then, they shall adapt their strategy to select document nodes that match the query, by computing, for example, complexity or readability measures, expertise level scores or genre.

The assumption is that context could improve the performances of Information Retrieval Systems. Modelling context in IR is considered as a long-term challenge by the community [11]. It is defined as the combination of retrieval technologies and knowledge on the context of the query and the user in a single model to provide the most suitable response compared to an information need. The contextual aspect refers to tacit or explicit knowledge concerning the intentions of users, the environment of users and the system itself. Modelling context is not an end in itself. The system must be able to decide the most adequate technologies compared to a given context, i.e.: to adapt the searching methods to the context. Of course, the user does not provide the system with the knowledge on the context of the required search. Classical IR approaches suppose a common objective which is topic relevance of top ranked documents. Opposite to this, approaches investigated by [12] allow to integrate scores orthogonal to the relevance score such as complexity. Document complexity must be matched to both users capability and level of specialisation in the target knowledge area. Different measures of difficulty can be incorporated to the relevance scores.

⁵ <http://en.over-blog.com>

⁶ This track relates to Contextual Information Retrieval and will be supported by the French National Research Agency ANR via the project CAAS (Contextual Analysis and Adaptive Search - ANR 2010 CORD 001 02).

The international community in information retrieval is indeed interested in research in contextual information retrieval since few years. The group IRiX (Information Retrieval in Context), proposed by Pr. Rijsbergen organizes an annual workshop on this topic since 2004. In the framework of the 6th PRCD (European commission), the network of excellence DELOS (Network of Excellence on Digital Libraries⁷) is interested in the access to information for users of various categories: individuals, specialists or population. Project PENG (Personalized News Content Programming⁸) is interested in defining an environment for programming and modelling the contents which makes it possible to offer interactive and personalized tools for multimedia information access to specialists or simple users.

7 Conclusion

In 2010 we provided a reusable resource for QA evaluation based on INEX ad-hoc 2009-2010 wikipedia document collection, including an original evaluation approach and software implementation.

In 2011 we will try to deal with two types of challenges. The first challenge is defining the query features that could help in predicting the query type and extracting it automatically from the query formulation or from the environment of the user. The second challenge is to be able to help the system to disambiguate the users expectation and use it for answering the queries.

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: [7], pp. 355–365 (2009)
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. LNI, vol. 103, pp. 277–291 GI (2007)
3. Pitler, E., Louis, A., Nenkova, A.: Automatic evaluation of linguistic quality in multi-document summarization. In: ACL, pp. 544–554 (2010)
4. Nenkova, A., Passonneau, R.: Evaluating content selection in summarization: The pyramid method. In: Proceedings of HLT-NAACL, vol. 2004 (2004)
5. Dang, H.: Overview of the TAC 2008 Opinion Question Answering and Summarization Tasks. In: Proc. of the First Text Analysis Conference (2008)
6. Louis, A., Nenkova, A.: Performance confidence estimation for automatic summarization. In: EACL, The Association for Computer Linguistics, pp. 541–548 (2009)
7. Geva, S., Kamps, J., Trotman, A. (eds.): Focused Retrieval and Evaluation, 8th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2009, Brisbane, Australia, December 7-9, LNCS, vol. 6203. Springer, Heidelberg (2010) (Revised and selected papers)

⁷ <http://www.delos.info/>

⁸ <http://www.peng-project.org/>

8. Moriceau, V., Tannier, X.: FIDJI: Using Syntax for Validating Answers in Multiple Documents. *Information Retrieval, Special Issue on Focused Information Retrieval* 13, 507–533 (2010)
9. SanJuan, E., Ibekwe-Sanjuan, F.: Combining language models with nlp and interactive query expansion. In: [7], pp. 122–132
10. Ibekwe-Sanjuan, F., SanJuan, E.: Use of multiword terms and query expansion for interactive information retrieval. In: Geva, S., Kamps, J., Trotman, A. (eds.) *INEX 2008*. LNCS, vol. 5631, pp. 54–64. Springer, Heidelberg (2009)
11. Allan, J., Aslam, J., Belkin, N.J., Buckley, C., Callan, J.P., Croft, W.B., Dumais, S.T., Fuhr, N., Harman, D., Harper, D.J., Hiemstra, D., Hofmann, T., Hovy, E.H., Kraaij, W., Lafferty, J.D., Lavrenko, V., Lewis, D.D., Liddy, L., Manmatha, R., McCallum, A., Ponte, J.M., Prager, J.M., Radev, D.R., Resnik, P., Robertson, S.E., Rosenfeld, R., Roukos, S., Sanderson, M., Schwartz, R.M., Singhal, A., Smeaton, A.F., Turtle, H.R., Voorhees, E.M., Weischedel, R.M., Xu, J., Zhai, C.: *Challenges in Information retrieval and language modeling: report of a workshop held at the center for intelligent information retrieval, university of massachusetts amherst*. *SIGIR Forum* 37(1), 31–47 (2002)
12. Gao, J., Qi, H., Xia, X., Nie, J.Y.: Linear discriminant model for information retrieval, pp. 290–297. *ACM*, New York (2005)

The GIL Summarizers: Experiments in the Track QA@INEX'10

Edmundo-Pavel Soriano-Morales, Alfonso Medina-Urrea,
Gerardo Sierra Martínez, 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 briefly describe two summarizers: ResúmeME and GIL-UNAM-3. Both are used to extract 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 QA@INEX 2010 task. The results of the experiments are evaluated automatically by means of the FRESA system.

Keywords: INEX, Automatic summarization system, Question-Answering system, WordNet synonyms, ResúmeME, GIL-UNAM-3 summarizer, FRESA system.

1 Introduction

A wide variety of text extraction techniques for summarizing documents exists (see, for instance, [3,11]). Our experience with summarization systems has included mainly word information content, sentence or utterance position, and standard deviation of utterance position measurements ([2,4,5]). In this paper, we explore the implementation of two of our summarization systems: ResúmeME and GIL-UNAM-3. In order to conduct the experiments, these summarizers rely on a) query enrichment by means of word synonyms and b) token occurrence in both the query and the document summarized. Both systems, ResúmeME and GIL-UNAM-3, are evaluated in the context of the question-answering track of the INEX 2010 task,¹ which provides questions either with short answers or with complex answers dealing with one or more utterances, possibly very long ones. Our summarizers are proposed as tools for dealing with experiments 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 separately ResúmeME

¹ <http://www.inex.otago.ac.nz/>

² <http://www.lemurproject.org/indri/>

and 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 ([6,8,9]).

The organization of this paper is as follows: in Section 2 the system ResúmeME is described; in Section 3 we give details on the GIL-UNAM-3 summarizer; in Section 4 we present the algorithm used in order to modify queries; the experimental settings and results are presented in Section 5; and, finally, in Section 6, we briefly present our conclusions and future work.

2 The ResúmeME Summarizer

ResúmeME is a single-document extractive summarization system. Several unsupervised statistical measurements are used to identify relevant utterances from source text. As expected, each of them is assigned with a score: hence utterances with highest score are going to be part of the final summary. This score was calculated by means of two measures: TF-IDF and S-I. The latter is a combination of information content and standard deviation. For each word of an utterance the two measurements are calculated. Consequently, the score of an utterance is based on an average of word weights. Formally, the score of an utterance is:

$$score(utterance) = \frac{score_{TF-IDF}(utterance) + score_{S-I}(utterance)}{2}$$

Afterwards, the system chooses as many utterances with highest score as the ratio compression value allows. They are presented in the final extract in the same order as they appear in the source text. The final utterance score is calculated by means of a smoothed average of utterance words. This average gives higher value to long utterances. The subjacent idea is that long utterances contain more information. Let n be the total number of words in an utterance. The total utterance score is defined as:

$$score_{TF-IDF} = \frac{\sum_{t=1}^n TF - IDF_{t,e} \times n}{\log n}$$

The S-I measurement combines word information content and standard deviation of word position as a measurement of text dispersion. This weight assigns a higher score to words with higher information content (higher standard deviation means words occur dispersed). The subjacent idea is that the most relevant words are the least frequent and most dispersed in a text.

Let E be the set of utterances in the source text and T the set of words in the same text. If $e \in E$ and $t \in T$, the S-I weight of a word is defined as:

$$weight_{S-I}(t) = \frac{s_t + I(t)}{2}$$

where s_t is the standard deviation of the positions of t and $I(t)$ is the information content of t .

3 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 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.

4 Query Modification

Query expansion is a general technique in the information retrieval field for generating alternative queries (query enrichment). For our experiments, we used it to replace terms in the original query with synonyms, in order to take advantage of additional information from the documents. Furthermore, in information extraction the most common query expansion approach is accomplished by means of thesaurus. Here we use WordNet³, a common resource for these purposes, for obtaining lexical relations based in sets of synonyms (synsets).

³ <http://wordnet.princeton.edu/>

Thus, 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. 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 2.7⁴ 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 ∈ OQ do
  | OriginalTokens ← Tokenize(originalquery) ;
  | NotFunctional ← RemoveFunctionalTokens(OriginalTokens);
  | foreach token ∈ NotFunctional do
  | | Synonym ← GetSynonymFromWordNet(token);
  | | NewTokens += Synonym;
  | end
  | ModifiedQuery ← ReplaceTokens(OriginalTokens, NewTokens);
  | NewQueries += ModifiedQuery;
end
OQ ← 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)

5 Experiments Settings and Results

As mentioned above, in order to evaluate the performance of both, the ResúmeME and GIL-UNAM-3 summarizers, applied on the QA@INEX corpus, we used the FRESA system, which does not rely on human produced summaries or human validation. The results of the experiments can be observed in Tables 1, 2, 3 and 4. Values represent divergence of the summaries with respect to the original documents. Tables 1 and 2 refer to the experiment with ResúmeME, whereas Tables 3 and 4 exhibit results of the GIL-UNAM-3 summarizer.

Table 1. FRESA results for modified query number 2009071 using the ResúmeMe summarizer

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	16.94064	25.61087	25.5306	22.69404

Both tables for the first experiment show similar tendencies. Also, both tables for the second one exhibit the best results.

It is interesting to notice, on the one hand, that random summaries (unigram and 5-gram) exhibit smaller divergence values than the summaries generated by both our summarizers. However, these summaries are unintelligible (given their randomness).

Table 2. FRESA results for modified query number 2010062 using the ResúmeMe summarizer

Distribution type	unigram	bigram	with 2-gap	Average
Baseline summary	16.49457	23.52351	23.6694	21.22916
Empty baseline	21.63229	29.20944	29.34705	26.72959
Random unigram	14.95259	22.61202	22.68446	20.08302
Random 5-gram	13.75724	20.61196	21.02727	18.46549
<i>Submitted summary</i>	20.21189	27.79291	27.92865	25.31115

Table 3. FRESA results for modified query number 2009071 using the GIL-UNAM-3 summarizer

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
<i>Submitted summary</i>	13.92082	21.7732	21.7724	19.15548

On the other hand, the divergence values for the summarizers differ with respect to the baseline. Namely, while the ResúmeME system exhibited a higher divergence than this baseline, the GIL-UNAM-3 summarizer showed a slightly lower one. Specifically, it can be observed that all divergence values of the summaries generated by ResúmeME are greater than the values of the baseline, but smaller than those of the empty baseline summaries (which consist of the words in the original documents not included in our summaries). Contrastingly, it can be observed that all divergence values of the summaries generated by the GIL-UNAM-3 summarizer are somehow smaller than the values of the baseline and noticeably smaller than those of the empty baseline summaries.

Finally, it is important to mention that the ResúmeME summarizer is tailor made for short texts of around 10 pages and the test documents for this experiment are significantly longer. This is relevant because the minimum percentage of reduction for ResúmeME is 10%, which generates extracts considerably longer

Table 4. FRESA results for modified query number 2010062 using the GIL-UNAM-3 summarizer

Distribution type	unigram	bigram	with 2-gap	Average
Baseline summary	16.4946	23.52351	23.6694	21.22916
Empty baseline	21.6323	29.20944	29.34705	26.72959
Random unigram	14.9526	22.61202	22.68446	20.08302
Random 5-gram	13.7572	20.61196	21.02727	18.46549
<i>Submitted summary</i>	16.1715	23.47908	23.62243	21.091

than the required ones. Thus, we applied ResúmeME iteratively to each document until it generated an extract of the required size. Perhaps, this accounts for the poor performance of this summarizer with respect to the performance of the GIL-UNAM-3 summarizer.

6 Conclusions

In this brief paper, we have presented experiments on the document sets made available during the Initiative for the Evaluation of XML Retrieval (INEX) 2010⁶, in particular on the QA@INEX 2010 Task (QA@INEX) <http://www.inex.otago.ac.nz/tracks/qa/qa.asp>. We have described both, the ResúmeME system and the GIL-UNAM-3 summarizer, which extract utterances from sets of documents retrieved by means of synonym modified queries. That is, we obtain from the WordNet database word synonyms for each word of the query. The systems apply several utterance selection metrics in order to extract those most likely to summarize a document; namely, TF-IDF, a combination of information content and standard deviation of utterance position, normalized cosine measurements and normalized word frequencies. As mentioned before, the queries are provided by the QA@INEX 2010 task. The results of the experiments are evaluated automatically by means of the FRESA system. Interestingly, our summarizers exhibit somehow different result tendencies. On the one hand, the one based on TF-IDF and a combination of information content and standard deviation of utterance position (ResúmeME) performs not as well as the baseline (although, it performs better than the empty one). On the other one, the one based on normalized cosine measurements and normalized word frequencies (GIL-UNAM-3) performs somehow better than the baselines.

Many adjustments can be made to these experiments. 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. Boudin, F., Torres Moreno, J.M.: NEO-CORTEX: A performant user-oriented multi-document summarization system. In: Gelbukh, A. (ed.) CICLing 2007. LNCS, vol. 4394, pp. 551–562. Springer, Heidelberg (2007)
2. Vasques, G., Ximena, M.: Sistema de resumen extractivo automático. In: Facultad de Ingeniería, UNAM, Mexico (2010)
3. Mani, I., Maybury, M.T.: Automatic Text Summarization. The MIT Press, Cambridge (1999)
4. Medina, U., Alfonso: De la palabra gráfica al texto: sobre la extracción de enunciados para el resumen automático. In: Vázquez Laslop, M.E., Zimmermann, K., Segovia, F. (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)

⁶ <http://www.inex.otago.ac.nz/>

5. Méndez, C., Carlos, F., Medina, U., Alfonso: Extractive Summarization Based on Word Information and Sentence Position. In: Gelbukh, A. (ed.) CICLing 2005. LNCS, vol. Alfonso, pp. 653–656. Springer, Heidelberg (2005)
6. Saggion, H., Torres-Moreno, J.M., da Cunha, I., SanJuan, E., Velásquez-Morales, P.: Multilingual Summarization Evaluation without Human Models. In: Proceedings of the 23rd International Conference on Computational Linguistics COLING 2010, Beijing (2010)
7. Torres Moreno, J.M., St-Onge, P.L., Gagnon, M., El-Béze, M., Bellot, P.: Automatic Summarization System coupled with a Question-Answering System (QAAS), CoRR (2009)
8. Torres-Moreno, J.M., Saggion, H., da Cunha, I., SanJuan, E.: Summary Evaluation with and without References. Polibits Research Journal on Computer Science and Computer Engineering and Applications 42 (2010)
9. Torres-Moreno, J.M., Saggion, H., da Cunha, I., Velázquez-Morales, P., SanJuan, E.: Évaluation automatique de résumés avec et sans référence. In: 17e Conférence sur le Traitement Automatique des Langues Naturelles. TALN, Montreal (2010)
10. Torres-Moreno, J.M., Velázquez-Morales, P., Jean-Guy, M.: Cortex: un algorithme pour la condensation automatique des textes. In: La Cognition Entre Individu et Société ARCo 2001, Lyon (2001)
11. Weiss, S.M., Indurkha, N., Zhang, T., Damerau, F.: Text Mining Predictive Methods for Analyzing Unstructured Information. Springer, Heidelberg (2005)

The Cortex Automatic Summarization System at the QA@INEX Track 2010

Juan-Manuel Torres-Moreno^{1,2} and Michel Gagnon²

¹ Université d'Avignon et des Pays de Vaucluse - LIA
339, chemin des Meinajariès, Agroparc BP 91228, 84911 Avignon Cedex 9 France

² 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
<http://www.lia.univ-avignon.fr>

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.

Keywords: 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 [6] 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]:

¹ *CONDensés et Résumés de TEXte* (Text Condensation and Summarization).

- Transform the questions into queries, then associate them to a set of documents;
- 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 that 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 [8,9,10]. 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 [11,12,13]. 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 approaches 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 eleven 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 and results are presented in Section 3. Section 4 exposes the conclusions of the paper and the future work.

2 The CORTEX System

Cortex (*CO*ndensation et *R*ésumés de *T*extes) [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 question and the document: words are filtered, lemmatized and stemmed.

The Cortex system uses 11 metrics (see [16] for a detailed description of these metrics) to evaluate the sentence's relevance.

- The frequency of words (F).
- The overlap between the words of the question (R).
- The entropy the words (E).

- The shape of text (Z).
- The angle between question and document vectors (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 system scores each sentence with a decision algorithm that relies on the normalized metrics. Before combining the votes of the metrics, these are partitioned into two sets: one set contains every metric $\lambda^i > 0.5$, while the other set contains every metric $\lambda^i < 0.5$ (values equal to 0.5 are ignored). We then calculate two values α and β , which give the sum of distances (positive for α and negative for β) to the threshold 0.5 (the number of metrics is Γ , which is 11 in our experiment):

$$\alpha = \sum_{i=1}^{\Gamma} (\lambda^i - 0.5); \quad \lambda^i > 0.5$$

$$\beta = \sum_{i=1}^{\Gamma} (0.5 - \lambda^i); \quad \lambda^i < 0.5$$

The value given to each sentence is calculated with:

$$\begin{aligned} &\text{if } (\alpha > \beta) \\ &\quad \text{then } \text{Score}^{\text{cortex}}(s, q) = 0.5 + \alpha/\Gamma \\ &\quad \text{else } \text{Score}^{\text{cortex}}(s, q) = 0.5 - \beta/\Gamma \end{aligned}$$

In addition to this score, two other measures are used: the question-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 specific similarity measure [17] between the question 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 of the document and the question q . Let $\mathbf{d} = (d_1 \dots d_n)$ and $\mathbf{q} = (q_1 \dots q_n)$ be the vectors representing the document and the question, respectively. The definition of the measure is:

$$\text{Similarity}(\mathbf{q}, \mathbf{d}) = \frac{\sum_{i=1}^n d_i q_i}{\sqrt{\sum_{i=1}^n d_i^2 + \sum_{i=1}^n q_i^2}}$$

The last measure, the topic-sentence overlap, assigns a higher ranking for the sentences containing question words and makes selected sentences more relevant. The overlap is defined as the normalized cardinality of the intersection between the question word set T and the sentence word set S .

$$\text{Overlap}(T, S) = \frac{\text{card}(S \cap T)}{\text{card}(T)}$$

The final score of a sentence s from a document d and a question q is the following:

$$Score = \alpha_1 Score^{cortex}(s, q) + \alpha_2 Overlap(s, q) + \alpha_3 Similarity(d, q)$$

where $\sum_i \alpha_i = 1$

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) 2010², in particular on the INEX 2010 QA Track (QA@INEX) <http://www.inex.otago.ac.nz/tracks/qa/qa.asp>.

To evaluate the efficacy of Cortex on INEX@QA corpus, we used the FRESA package³.

INEX queries. No 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 system 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	SU4-gram	FRESA average
Baseline summary	26.679	34.108	34.218	31.668
Empty baseline	31.715	39.452	39.534	36.901
Random unigram	25.063	32.822	32.852	30.246
Random 5-gram	23.168	30.644	30.838	28.217
Cortex summary	26.421	33.935	34.030	31.462

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.

² <http://www.inex.otago.ac.nz/>

³ FRESA package is disponible at web site: http://lia.univ-avignon.fr/fileadmin/axes/TALNE/downloads/index_fresa.html

References

1. ANSI. American National Standard for Writing Abstracts. Technical report, American National Standards Institute, Inc., New York, NY (ANSI Z39.14.1979) (1979)
2. Luhn, H.P.: The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development* 2(2), 159 (1958)
3. Edmundson, H.P.: New Methods in Automatic Extracting. *Journal of the ACM (JACM)* 16(2), 264–285 (1969)
4. Salton, G.: *The SMART Retrieval System - Experiments un Automatic Document Processing*, Englewood Cliffs (1971)
5. Salton, G., McGill, M.: *Introduction to Modern Information Retrieval*. McGraw-Hill, New York (1983)
6. Torres-Moreno, J.M., Velazquez-Morales, P., Meunier, J.-G.: Condensés automatiques de textes. *Lexicometrica. L'analyse de données textuelles: De l'enquête aux corpus littéraires, Special* (2004), www.cavi.univ-paris3.fr/lexicometrica
7. Jacquemin, C., Zweigenbaum, P.: Traitement automatique des langues pour l'accès au contenu des documents. *Le Document en Sciences du Traitement de l'information* 4, 71–109 (2000)
8. Abracos, J., Lopes, G.P.: Statistical Methods for Retrieving Most Significant Paragraphs in Newspaper Articles. In: Mani, I., Maybury, M.T. (eds.) *ACL/EACL 1997-WS*, Madrid, Spain, July 11 (1997)
9. Teufel, S., Moens, M.: Sentence Extraction as a Classification Task. In: Mani, I., Maybury, M.T. (eds.) *ACL/EACL 1997-WS*, Madrid, Spain (1997)
10. Hovy, E., Lin, C.Y.: Automated Text Summarization in SUMMARIST. In: Mani, I., Maybury, M.T. (eds.) *Advances in Automatic Text Summarization*, pp. 81–94. The MIT Press, Cambridge (1999)
11. Kupiec, J., Pedersen, J.O., Chen, F.: A Trainable Document Summarizer. In: *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 68–73 (1995)
12. Tombros, A., Sanderson, M., Gray, P.: Advantages of Query Biased Summaries in Information Retrieval. In: Hovy, E., Radev, D.R. (eds.) *AAAI1998-S*, Stanford, California, USA, March 23–25, pp. 34–43. The AAAI Press, Menlo Park (1998)
13. Schlesinger, J.D., Backer, D.J., Donway, R.L.: Using Document Features and Statistical Modeling to Improve Query-Based Summarization. In: *DUC 2001*, New Orleans, LA (2001)
14. Torres-Moreno, J.M., Velazquez-Morales, P., Meunier, J.: CORTEX, un algorithme pour la condensation automatique de textes. In: *ARCo*, vol. 2, p. 365 (2005)
15. Torres-Moreno, J.M., St-Onge, P.-L., Gagnon, M., El-Bèze, M., Bellot, P.: Automatic summarization system coupled with a question-answering system (qaas). *CoRR*, abs/0905.2990 (2009)
16. Torres-Moreno, J.M., Velazquez-Morales, P., Meunier, J.G.: Condensés de textes par des méthodes numériques. *JADT* 2, 723–734 (2002)
17. Salton, G.: *Automatic text processing*, 9th edn. Addison-Wesley Longman Publishing Co. Inc., Amsterdam (1989)

The REG Summarization System with Question Reformulation at QA@INEX Track 2010

Jorge Vivaldi¹, Iria da Cunha¹, and Javier Ramírez²

¹ Instituto Universitario de Lingüística Aplicada - UPF
Barcelona

² Universidad Autónoma Metropolitana-Azcapotzalco
Mexico

{[iria.dacunha](mailto:iria.dacunha@upf.edu), [jorge.vivaldi](mailto:jorge.vivaldi@upf.edu)}@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 and name entities from the queries, in order to obtain a list of terms and name entities related with the main topic of the question. Using this strategy, REG obtained good results regarding performance (measured with the automatic evaluation system FRESA) and readability (measured with human evaluation), being one of the seven best systems into the task.

Keywords: 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 Ramírez, 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 and name entities they contain, 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 and name entities 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 Ramírez, 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.

The complexity of REG algorithm is $O(n^2)$. Nevertheless, there is a limitation, because it includes a fast classification algorithm which can be used only for short instances; this is the reason it is not very efficient for long texts.

3 Terms and Name Entity Extraction

The starting point of this work is to consider that the terms or name entities included into the queries are representative of the main subject of these queries. If this assumption is true, the results obtained by a search engine would be optimized if we give it as input the list of terms or name entities of the queries, instead the complete queries.

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 and the name entities present in every query. On the one hand, terms are usually defined as lexical units to designate concepts in a domain. The notion of term that we have adopted in this work is based on the “Communicative Theory of Terminology” (Cabr e, 1999): a term is a lexical unit (single/multiple word) that designates a concept in a given specialized domain. This means that terms have a designative function and, obviously not all the words in a language have such function. To detect which are the words that in a given text have this designative function is the main task of term detector/extractors (see Cabr e et al. (2001) for a review of this subject). The detection of these units is a complex task mainly because terms adopt, with a few exceptions, all word formation rules in a given language (Pearson, 1998; Sager, 1999). 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”. Some terminological extractors are described in Barr on-Cede no et al. (2009), Wong et al. (2008), Sclano et al. (2007), Vivaldi and Rodr guez (2001a, 2001b), Vivaldi (2001) or Bourigault and Jacquemin (1999), among others.

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 Cabr e et al., 2001; Pazienza et al., 2005), 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. We consider termhood as it has been defined in Kageura and Umino (1996): “the degree that a linguistic unit is related to domain-specific concepts”.

On the other hand, the main objective of name entity recognition systems is to found automatically some units into texts. These units can be names of persons, names of organizations, locations, etc. Some works about this subjects are Jun’ichi and Kentaro (2007), Leong and Tou (2002) or Volk and Clematide (2001), among many others.

As we have seen, nowadays several term extraction systems and name entity recognition systems exist for English. Nevertheless, their performances are not still perfect, so if we employ these systems in our work, their mistakes and the mistakes of the system we present here would be mixed. Moreover, term extractors are usually designed for a specialized domain, as medicine, economics, law, etc, but the topics of the queries provided by INEX@QA 2010 are several, that is, they do not correspond to an unique domain. As the INEX@QA 2010 organizers state (SanJuan et al., 2010: 2):

“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”.

These facts, the performance of term and name entity extractors, and the diversity of domains of the queries, made us to carry out the term and name entity extraction not using a specific term extraction tool but just using a POS tagger. In practice, we mainly extract nominal sentences. Doing the task in this way, we are sure that the results we obtain are precise and we can evaluate our system properly.

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 that the terms “glsl”, “vertex processing”, “fragment processing” and “single instruction set” are useful to find the right answer. But for the query “Who is Eiffel?”, there are not any term, only the name entity “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) 2010¹, in particular on the INEX 2010 QA Track (QA@INEX). These sets of documents were provided by the search engine Indri². 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. This evaluation framework (FRESA –FRamework for Evaluating Summaries Automatically-) includes document-based summary evaluation measures based on probabilities distribution, specifically, the Kullback Leibler (KL) divergence and the Jensen Shannon (JS) divergence. As in the ROUGE package (Lin, 2004), FRESA supports different n-grams and skip n-grams probability distributions. The FRESA environment has been used in the evaluation of summaries produced in several European languages (English, French, Spanish and Catalan), and it integrates filtering and lemmatization in the treatment of summaries and documents. FRESA is available in the following link:

<http://lia.univ-avignon.fr/fileadmin/axes/TALNE/Ressources.html>

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 (“What are the similarities and differences between mean average precision and reciprocal rank used in Information Retrieval?”). The extracted

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

terms for this query were: “similarities”, “differences”, “mean average precision”, “reciprocal rank” and “information retrieval”. 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.

Readability and coherence evaluation required human evaluation. Humans evaluated the correctness of the answers detecting incoherent grammatical structures, unsolved anaphora or redundant passages. For more information about the human evaluation methodology that INEX@QA 2010 organizers used, see San-Juan et al. (2010).

Table 2 includes the final results of the task. There were nearly 100 participants into the INEX@QA 2010 task, using different approaches. Our system (number 857) is one of the best seven systems. It obtains 2031.85 average performance and average readability. Regarding performance, it is the fourth best system and, regarding readability, its results are similar to four other systems. The best performance was obtained by the system 98 (2006.45), based on focused Information Retrieval (IR), but it obtains a bad readability (the worst readability into the evaluation). The best system regarding readability was the 943, but it was the worst regarding average performance (2050.81).

We consider that the results obtained by our system are positive: neither performance nor readability are bad. Moreover, results on both categories (performance and readability) are good. We think that the fact our system is able to maintain a good level in both categories means that our system is robust.

Table 2. Evaluation results for the seven best participants into the INEX@QA 2010 task: Cumulative KL divergence

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

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 and the name entities from the queries, in order to obtain a list of terms and name entities related with the main topic of the question.

We consider that, over the INEX-2010 corpus, REG obtained good results in the automatic evaluations. The experiments have shown that the performance and readability of our system are very good, in comparison to the other participants in the INEX@QA 2010 task: it is the fourth best system. Neither performance nor readability are bad, and we consider that this fact confirms the robustness of the system.

We think that the main problem of our methodology is that some queries are long and they have several terms or name entities we could extract, but there are some queries that are very short and the extraction is not possible or very limited. As future work, we would like to include more semantic aspects to solve this problem. Eventually, we may use Wikipedia for selecting the most relevant nominal sentences found in the question. In this way, some nominal sentences may be not used in the query, improving the results obtained from the search engine. In the case of very short questions, the query may be expanded using information obtained from this resource.

We would like to use in the future another similar but more efficient algorithm (like merge sort or heapsort, with complexity $O(n \lg n)$; see Cormen et al., 2005), before the greedy algorithm, when the sentences ranking is performed.

References

1. Abracos, J., Lopes, G.: Statistical methods for retrieving most significant paragraphs in newspaper articles. In: Proceedings of the ACL/EACL 1997 Workshop on Intelligent Scalable Text Summarization, Madrid, pp. 51–57 (1997)
2. Afantenos, S., Karkaletsis, V., Stamatopoulos, P.: Summarization of medical documents: A survey. *Artificial Intelligence in Medicine* 33(2), 157–177 (2005)
3. Barrón-Cedeño, A., Sierra, G., Drouin, P., Ananiadou, S.: An Improved Automatic Term Recognition Method for Spanish. In: Gelbukh, A. (ed.) *CICLing 2009*. LNCS, vol. 5449, pp. 125–136. Springer, Heidelberg (2009)
4. Bourigault, D., Jacquemin, C.: Term Extraction + Term Clustering: an integrated platform for computer-aided terminology. In: Proceedings of EACL, pp. 15–22 (1999)
5. Cabré, M.T.: *La terminología. Representación y comunicación*. IULA-UPF, Barcelona (1999)
6. Cabré, M.T., Estopà, R., Vivaldi, J.: Automatic term detection: a review of current systems. In: Bourigault, D., Jacquemin, C., L’Homme, M.C. (eds.) *Recent Advances in Computational Terminology*, pp. 53–87. John Benjamins, Amsterdam (2001)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to algorithms*, 2nd edn. The MIT Press, Cambridge (2005)

8. da Cunha, I., Wanner, L., Cabré, M.T.: Summarization of specialized discourse: The case of medical articles in Spanish. *Terminology* 13(2), 249–286 (2007)
9. Edmunson, H.P.: New Methods in Automatic Extraction. *Journal of the Association for Computing Machinery* 16, 264–285 (1969)
10. Farzindar, A., Lapalme, G., Desclés, J.-P.: Résumé de textes juridiques par identification de leur structure thématique. *Traitement Automatique des Langues* 45(1), 39–64 (2004)
11. Fuentes, M., Gonzalez, E., Rodriguez, H.: Resumidor de noticies en catala del projecte Hermes. In: *Proceedings of II Congr s d'Enginyeria en Llengua Catalana (CELC 2004)*, Andorra, pp. 102–102 (2004)
12. Gaizauskas, R., Herring, P., Oakes, M., Beaulieu, M., Willett, P., Fowkes, H., Jonsson, A.: Intelligent access to text: Integrating information extraction technology into text browsers. In: *Proceedings of the Human Language Technology Conference, San Diego*, pp. 189–193 (2001)
13. Johnson, D.B., Zou, Q., Dionisio, J.D., Liu, V.Z., Chu, W.W.: Modeling medical content for automated summarization. *Annals of the New York Academy of Sciences* 980, 247–258 (2002)
14. Jun'ichi, K., Kentaro, T.: Exploiting Wikipedia as External Knowledge for Name Entity Recognition. In: *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 698–707 (2007)
15. Kageura, K., Umino, B.: Methods of automatic term recognition: A review. *Terminology* 3(2), 259–289 (1996)
16. Lal, P., Reger, S.: Extract-based Summarization with Simplification. In: *Proceedings of the 2nd Document Understanding Conference at the 40th Meeting of the Association for Computational Linguistics*, pp. 90–96 (2002)
17. Leong Chieu, H., Tou Ng, H.: Named entity recognition: a maximum entropy approach using global information. In: *Proceedings of the 19th International Conference on Computational Linguistics*, pp. 1-7 (2002)
18. Lin, C.-Y.: ROUGE: A Package for Automatic Evaluation of Summaries. In: *Proceedings of Text Summarization Branches Out: ACL 2004 Workshop*, pp. 74–81 (2004)
19. Nanba, H., Okumura, M.: Producing More Readable Extracts by Revising Them. In: *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrucken, pp. 1071–1075 (2000)
20. Ono, K., Sumita, K., Miike, S.: Abstract generation based on rhetorical structure extraction. In: *Proceedings of the International Conference on Computational Linguistics, Kyoto*, pp. 344–348 (1994)
21. Paice, C.D.: Constructing literature abstracts by computer: Techniques and prospects. *Information Processing and Management* 26, 171–186 (1990)
22. Paziienza, M.T., Pennacchiotti, M., Zanzotto, F.M.: Terminology Extraction: An Analysis of Linguistic and Statistical Approaches. In: *Studies in Fuzziness and Soft Computing*, vol. 185, pp. 255–279 (2005)
23. Pearson, J.: *Terms in context*. John Benjamin, Amsterdam (1998)
24. Radev, D.: *Language Reuse and Regeneration: Generating Natural Language Summaries from Multiple On-Line Sources*. New York, Columbia University [PhD Thesis] (1999)
25. Sager, J.C.: In search of a foundation: Towards a theory of terms. *Terminology* 5(1), 41–57 (1999)
26. Saggion, H., Lapalme, G.: Generating Indicative-Informative Summaries with SumUM. *Computational Linguistics* 28(4), 497–526 (2002)

27. Saggion, H., Torres-Moreno, J.-M., da Cunha, I., SanJuan, E., Velázquez-Morales, P., SanJuan, E.: Multilingual Summarization Evaluation without Human Models. In: Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010), Pekin (2010)
28. SanJuan, E., Bellot, P., Moriceau, V., Tannier, X.: Overview of the 2010 QA Track: Preliminary results. In: Geva, S., et al. (eds.) INEX 2010. LNCS, vol. 6932, pp. 269–281. Springer, Heidelberg (2010)
29. Sclano, F., Velardi, P.: Termextractor: a web application to learn the shared terminology of emergent web communities. In: Proceedings of the 3rd International Conference on Interoperability for Enterprise Software and Applications, pp. 287–298 (2007)
30. Torres-Moreno, J.-M., Saggion, H., da Cunha, I., SanJuan, E., Velázquez-Morales, P., SanJuan, E.: Summary Evaluation With and Without References. *Polibitis: Research Journal on Computer Science and Computer Engineering with Applications* 42 (2010a)
31. Torres-Moreno, J.-M., Saggion, H., da Cunha, I., Velázquez-Morales, P., SanJuan, E.: 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 (2010)
32. Torres-Moreno, J.-M., Ramírez, J.: REG: un algorithme glouton appliqué au résumé automatique de texte. In: JADT 2010, Roma, Italia (2010)
33. Torres-Moreno, J.-M., Ramírez, J.: Un resumeur a base de graphes, indépendant de la langue. In: Proceedings of the International Workshop African HLT 2010, Djibouti (2010)
34. Torres-Moreno, J.M., Velázquez-Morales, P., Meunier, J.G.: Condensés de textes par des méthodes numériques. In: Proceedings of the 6th International Conference on the Statistical Analysis of Textual Data (JADT), St. Malo, pp. 723–734 (2002)
35. Vivaldi, J., da Cunha, I., Torres-Moreno, J.M., Velázquez, P.: Automatic Summarization Using Terminological and Semantic Resources. In: En Actas del 7th International Conference on Language Resources and Evaluation (LREC 2010), Valletta, Malta (2010)
36. Vivaldi, J.: Extracción de candidatos a término mediante combinación de estrategias heterogéneas. Ph.D. thesis, Universitat Politcnica de Catalunya, Barcelona (2001)
37. Vivaldi, J., Rodríguez, H.: Improving term extraction by combining different techniques. *Terminology* 7(1), 31–47 (2001a)
38. Vivaldi, J., Màrquez, L., Rodríguez, H.: Improving term extraction by system combination using boosting. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 515–526. Springer, Heidelberg (2001b)
39. Volk, M., Clematide, S.: Learn-filter-apply-forget. Mixed approaches to name entity recognition. In: Proceedings of the 6th International Workshop on Applications of Natural Language for Informations Systems, Madrid, Spain (2001)
40. Won, W., Liu, W., Bennamoun, M.: Determination of Unithood and Termhood for Term Recognition. In: Song, M., Wu, Y. (eds.) *Handbook of Research on Text and Web Mining Technologies*. IGI Global (2008)

Overview of the INEX 2010 Focused Relevance Feedback Track

Timothy Chappell¹ and Shlomo Geva²

¹ Queensland University of Technology
t.chappell@student.qut.edu.au

² Queensland University of Technology
s.geva@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 nine submissions from three participating organisations.

1 Introduction

This paper presents an overview of the INEX 2010 Focused Relevance Feedback track. The purpose behind the track is to evaluate the performance of focused relevance feedback plugins in comparison to each other against unknown data. The data used for this track is the document collection and the assessments collected for the INEX 2010 Ad Hoc track. Organisations participated in the track by submitting their algorithms in the form of dynamic libraries implementing a ranking function capable of receiving relevance information from the evaluation platform and acting on it to improve the quality of future results. The interface also allows the algorithms to provide back more detailed information, such as the section or sections within a document that it believes are most relevant, enabling focused results to be returned.

The result of running the algorithms against a set of topics is a set of relevance assessments, which can then be scored against the same assessments used to provide feedback to the algorithms. The result is a reflection of how well the algorithms were able to learn from the relevance information they were given.

2 Focused Relevance Feedback

The relevance feedback approach that is the focus of this track is a modified form of traditional approaches to relevance feedback, which typically involved nominating whole documents as either relevant or not relevant. The end user would typically be presented with a list of documents which they would mark as relevant or not relevant before returning this input to the system which would search the remainder of the collection for similar documents and present them to the user.

Due to a fundamental paradigm change in how people use computers since these early approaches to relevance feedback, a more interactive feedback loop where the user continues to provide relevance information as they go through the results is now possible. 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 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.

3 Evaluation

The Focused Relevance Feedback track is concerned with the simulation of a user interacting with an information retrieval system, searching for a number of different topics. The quality of the results this user receives is then used to evaluate the relevance feedback approach.

The INEX Ad-Hoc track, which evaluates ranking algorithms, makes use of user-collected *assessments* on which portions of documents are relevant to users searching for particular topics. These assessments are perfect, not just for the evaluation of the rankings produced by the algorithms, but also for providing Focused Relevance Feedback algorithms with the relevance information they need.

As such, a Focused Relevance Feedback algorithm can be mechanically evaluated without a need of a real user by simulating one, looking up the appropriate assessments for each document received from the algorithm and sending back the relevant passages.

To be able to accurately evaluate and compare the performance of different focused relevance feedback algorithms, it is necessary that the algorithms not be trained on the exact relevance assessments they are to receive in the evaluation. After all, a search engine isn't going to know in advance what the user is looking for. For this reason, it becomes necessary to evaluate an algorithm with data that was not available at the time the algorithm is written. Unlike in the Ad-Hoc track, the relevance submissions used to evaluate the plugins are also required for input to the plugins, so there is no way to provide participating organisations with enough information for them to provide submissions without potentially gaining an unrealistic advantage.

There are at least two potential ways of rectifying this. One is to require the submission of the algorithms a certain amount of time (for example, one hour) after the assessments for the Ad Hoc track were made available. This approach, however, is flawed as it allows very little margin for error and that it will unfairly advantage organisations that happen to be based in the right time zones, depending on when the assessments are released. In addition, it allows the relevance feedback algorithm to look ahead at relevance results it has not yet received in order to artificially improve the quality of the ranking. These factors make it unsuitable for the running of the track.

The other approach, and the one used in the Focused Relevance Feedback track, is to have the participating organisations submit the algorithms themselves, rather than just the results. The algorithms were submitted as dynamic libraries written in Java, chosen for its cross-platform efficiency. The dynamic libraries were then linked into an evaluation platform which simulated a user searching for a number of different topics, providing relevance results on each document given. The order in which the documents were submitted to the platform was then used to return a ranking, which could be evaluated like the results of any ranking algorithm.

4 Task

4.1 Overview

Participants were asked to create one or more Relevance Feedback Modules 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 Relevance Feedback Modules were implemented as dynamically linkable modules that implement a standard defined interface. The Evaluation Platform interacts with the Relevance Feedback Modules directly, simulating a user search session. The Evaluation Platform instantiates a Relevance Feedback Module object and provides it with a set of XML documents and a query.

The Relevance Feedback Module responds by ranking the documents (without feedback) and returning the ranking to the Evaluation Platform. 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 Evaluation Platform 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 Evaluation Platform passes relevance feedback (in the form of passage offsets and lengths) about the last document presented by the Relevance Feedback Module. This feedback is taken from the grels of the respective topic, as provided by the Ad-Hoc track assessors. The simulated user feedback may then be used by the Relevance Feedback Module to re-rank the remaining unseen documents and return the next most relevant document. The Evaluation Platform makes repeated calls to the Relevance Feedback Module until all relevant documents in the collection have been returned.

The Evaluation Platform retains the presentation order of documents as generated by the Relevance Feedback Module. 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.

4.2 Submission Format

Participating organisations submitted JAR files that implemented the following specification:

```
package rf;

public interface RInterface {
    public Integer[] first(String[] documentList, String query);
    public Integer next();
    public String getFOL();
    public String getXPath();
    public void relevant(Integer offset, Integer length,
                        String Xpath, String relevantText);
}
```

In the call to *first*, the algorithm is given the set of documents and the query used to rank them and must return an initial ranking of the documents. The purpose of this is to quantify the improvement gained from providing the relevance assessments to the Relevance Feedback Module. The Evaluation Platform then calls *next* to request the next document from the algorithm, making a call to *relevant* to provide feedback on any relevant passages in the document. The optional methods *getFOL* and *getXPath*, if implemented, allow the Relevance

Feedback Module to provide more focused results to the Evaluation Platform in order to gain better results from the focused evaluation. None of the submitted algorithms implemented these methods, however.

A subset of 10 topics that assessments were created for in the Ad-Hoc track were selected to evaluate the topics. These topics were chosen by taking the first 10 topics that were manually assessed during the assessment stage of the Ad Hoc track. Between them these topics covered a total of 7488 documents (with some overlap), of which 1421 were marked as *relevant* (containing at least one passage marked as relevant.) On average, in the *relevant* documents 36.43% of the text was marked as relevant, with the shortest relevant passage being 29 characters long.

Each Relevance Feedback Module receives a call to *first* for each topic, but is not otherwise reinitialised, giving the module opportunity to implement some form of *learning to rank* approach. It is difficult to tell if any of the submissions made use of this.

5 Results

5.1 Submissions

Three groups submitted a total of nine Relevance Feedback Modules to the INEX 2010 Relevance Feedback track. While this submission pool is small, it was to be expected given that this is the first time the track has been run. QUT resubmitted the reference Relevance Feedback Module. The Indian Statistical Institute (ISI) submitted three modules and Peking University submitted five.

To provide a starting point for participating organisations, a reference Relevance Feedback Module, both in source and binary form, was provided by QUT. This reference module used the ranking engine Lucene^[2] as a base for a modified Rocchio^[3] approach. The approach used was to provide the document collection to Lucene for indexing, then construct search queries based on the original query but with terms added from those selections of text nominated as relevant. A scrolling character buffer of constant size was used, with old data rolling off as new selections of relevant text were added to the buffer, and popular terms (ranked by term frequency) added to the search query. The highest ranked document not yet returned is then presented to the Evaluation Platform and this cycle continues until the collection is exhausted. The reference algorithm does not provide focused results and as such does not implement the *getFOL* or *getX-Path* methods.

The Indian Statistical Institute (ISI) submitted a relevance feedback algorithm that was designed around finding non-overlapping word windows in the relevant passages and modifying the query to include these terms.

Peking University submitted an algorithm that used a Rocchio-based algorithm revised to include negative feedback and criterion weight adjustments.

Table 1. Average precision and R-precision for submitted modules

Run	Organisation	Average Precision	R-Precision
Reference	QUT	0.4827	0.5095
1367	ISI	0.3770	0.3845
1366	ISI	0.3617	0.3503
1365	ISI	0.3445	0.3307
1368	Peking	0.2275	0.2171
1373	Peking	0.2210	0.2015
1369	Peking	0.2196	0.2015
1371	Peking	0.2185	0.2015
1370	Peking	0.2177	0.1962

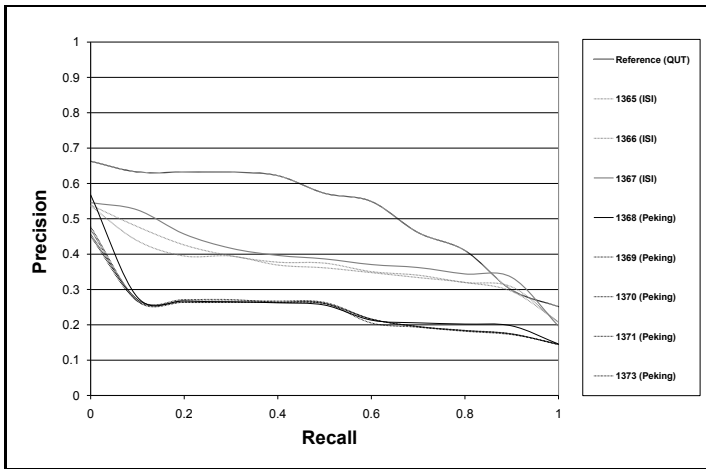


Fig. 1. Recall-precision comparison of Relevance Feedback Modules

Table 2. Average precision and R-precision for submitted modules, before and after feedback

Run	Organisation	Without feedback		With feedback	
		Average	R-Precision	Average	R-Precision
Reference	QUT	0.2796	0.3240	0.4827	0.5095
1367	ISI	0.2771	0.2718	0.3770	0.3845
1366	ISI	0.2771	0.2718	0.3617	0.3503
1365	ISI	0.2768	0.2718	0.3445	0.3307
1368	Peking	0.2286	0.2031	0.2275	0.2171
1373	Peking	0.2231	0.2158	0.2210	0.2015
1369	Peking	0.2231	0.2158	0.2196	0.2015
1371	Peking	0.2292	0.2159	0.2185	0.2015
1370	Peking	0.2292	0.2159	0.2177	0.1962

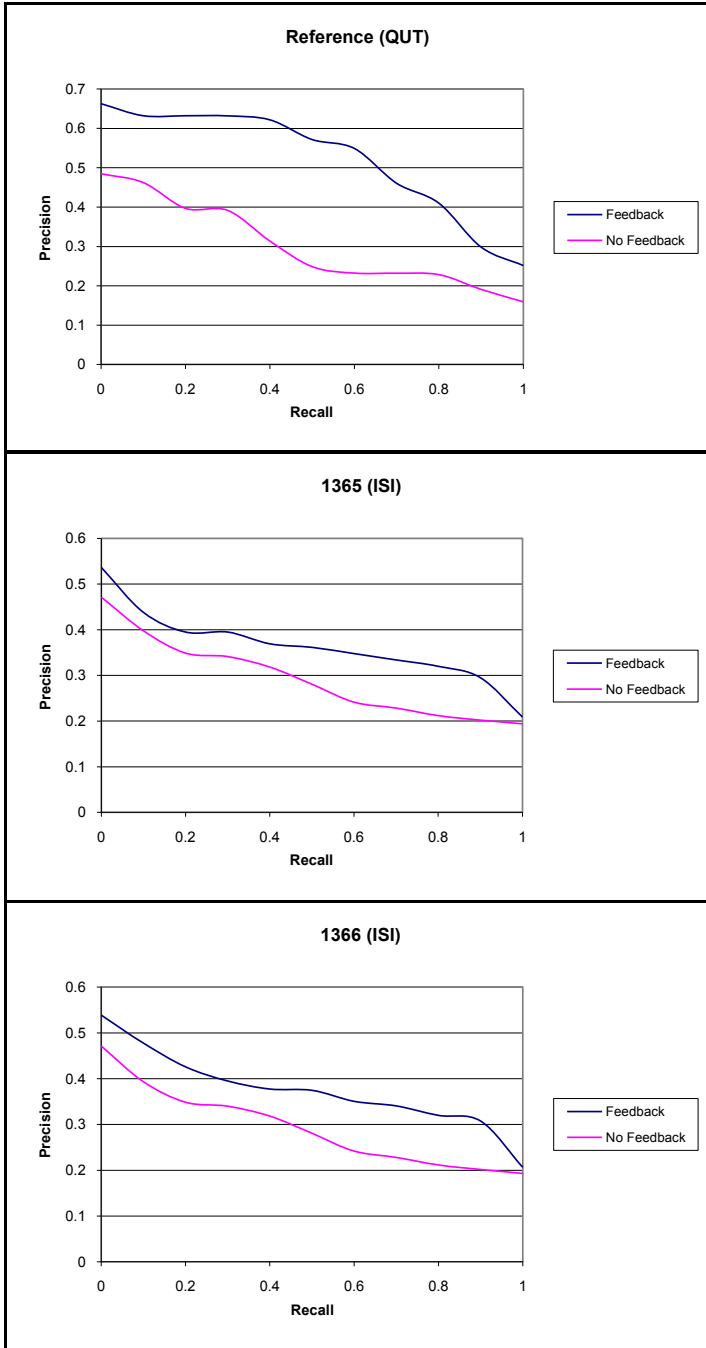


Fig. 2. Recall-precision comparison for track submissions

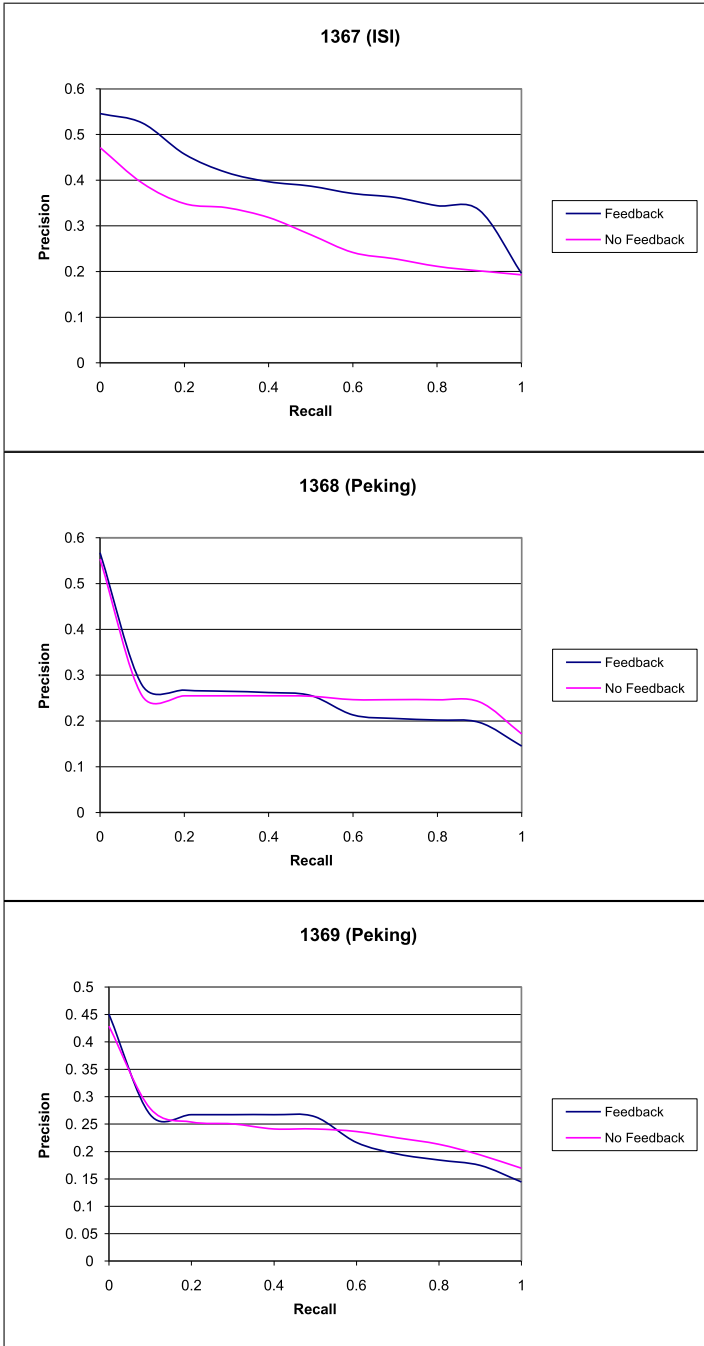


Fig. 3. Recall-precision comparison for track submissions

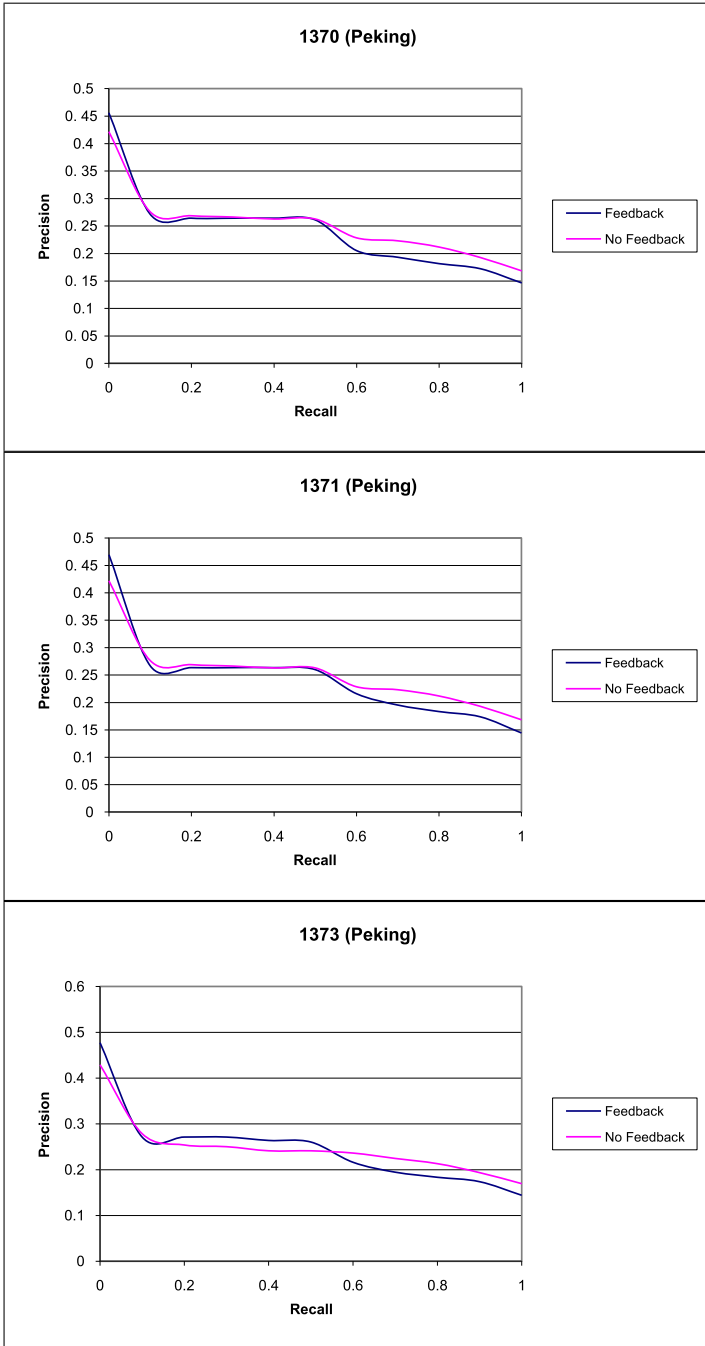


Fig. 4. Recall-precision comparison for track submissions

5.2 Evaluation

The Relevance Feedback Modules submitted by participating organisations were run through the Evaluation Platform. As none of the submitted Relevance Feedback Modules returned focused results, *trec eval* [1] was used to evaluate the results.

The two measures used in this evaluation are *average precision* and *R-precision*. R-precision is calculated as the precision (number of relevant documents) after R documents have been seen, where R is the number of relevant documents in the collection. Average precision is calculated from the sum of the precision at each recall point (a point where a certain fraction of the documents in the collection have been seen) divided by the number of recall points.

To see how the different Relevance Feedback Modules compared over the entire evaluation, a recall-precision curve can be plotted to show how they stack up. These recall-precision curves were created with *trec eval* and consist of precision values at 11 points from 0.0 to 1.0 recall. The precision value used is the highest precision within that recall point.

The ranking results returned by the Relevance Feedback Modules show how each module would have ranked the documents if feedback was not available. This information can then be used to show how use of feedback changes the results from each algorithm:

The results can also be compared on a recall-precision curve to show how the use of feedback data changed the results of each algorithm.

6 Conclusion

We have presented the Focused Relevance Feedback track at INEX 2010. Despite the limited pool of participating organisations, the track has provided a good starting point for further work in this area and it is hoped that the Focused Relevance Feedback track will be more successful at INEX 2011. While the INEX Ad Hoc track—used as the source of assessment data for the Focused Relevance Feedback track—will not be running in INEX 2011, there is plenty of assessment data that has not so far been used and as such future runs of this track will use assessments from previous runs of the Ad Hoc track.

Acknowledgements. We would like to thank all the participating organisations for their contributions and hard work.

References

1. Buckley, C.: The trec eval IR evaluation package (2004) (Retrieved January 1, 2005)
2. Goetz, B.: The Lucene search engine: Powerful, flexible, and free. Javaworld (2002), <http://www.javaworld.com/javaworld/jw-09-2000/jw-0915-lucene.html>
3. Rocchio, J.J.: Relevance feedback in information retrieval. In: Salton, G. (ed.) The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall Series in Automatic Computation, vol. 14, pp. 313–323. Prentice-Hall, Englewood Cliffs (1971)

Exploring Accumulative Query Expansion for Relevance Feedback

Debasis Ganguly, Johannes Leveling, and Gareth J.F. Jones

CNGL, School of Computing, Dublin City University, Dublin 9, Ireland
{dganguly,jleveling,gjones}@computing.dcu.ie

Abstract. For the participation of Dublin City University (DCU) in the Relevance Feedback (RF) track of INEX 2010, we investigated the relation between the length of relevant text passages and the number of RF terms. In our experiments, relevant passages are segmented into non-overlapping windows of fixed length which are sorted by similarity with the query. In each retrieval iteration, we extend the current query with the most frequent terms extracted from these word windows. The number of feedback terms corresponds to a constant number, a number proportional to the length of relevant passages, and a number inversely proportional to the length of relevant passages, respectively. Retrieval experiments show a significant increase in MAP for INEX 2008 training data and improved precisions at early recall levels for the 2010 topics as compared to the baseline Rocchio feedback.

1 Introduction

Query expansion (QE) is a popular technique to improve information retrieval effectiveness by extending the original query. The Relevance Feedback (RF) track at INEX 2010 attempts to simulate user interaction by communicating *true* relevance information between a *Controller module*, with access to the *qrels* file and simulates RF from a user, and a *Feedback module*. This allows re-ranking results by changing the set of retrieved documents in every retrieval iteration. In the RF track, the incremental reporting of relevant text segments from full documents allows the development of a feedback algorithm choosing feedback terms in different ways, compared to standard Blind Relevance Feedback (BRF). The exchange of relevance information between user and system denotes a retrieval iteration and can be repeated multiple times for the same query. In each iteration, the feedback algorithm or its parameters can be adapted to improve retrieval performance.

In this paper, we investigate the relationship between the length of relevant passages and the number of feedback terms. We explore three variants of selecting the number of feedback terms depending on the length of relevant test segments: i) choosing a constant number of feedback terms, ii) choosing a number directly proportional to the lengths of the relevant segments, and iii) choosing a number inversely proportional to the lengths of the relevant segments. All three approaches can be justified in their own way. One might want to choose more

terms from a smaller relevant segment in the hope that it has less or no noisy terms. It might be more effective to choose more terms from larger relevant segments on the assumption that the likelihood of finding useful expansion terms increases with the length of a relevant section. Finally, the length of a relevant passage may be unrelated to the best number of feedback terms so that a constant number of feedback terms is the best choice.

The rest of this paper is organized as follows: Section 2 describes the motivation of the RF experiments and related work, our RF algorithm is introduced in Section 3. Section 4 reports our results in the RF track and analyzes the results and we conclude the paper with directions for future work in Section 5.

2 Related Work

One of the problems of BRF is that *all* terms which meet the selection criterion for feedback terms are used for QE. This includes terms which are not related to the query, for example semantically unrelated, but highly frequent terms from long (pseudo-)relevant documents or text segments.¹ A number of experiments using small text passages instead of full documents for BRF have been conducted [1,2,3,4,5]. One assumption behind using small passages is that long documents can contain a wider range of discourse and noisy terms would be added to the original query, which can result in a topic shift. A wide range of discourse in long documents means that the relevant portion of such a document may be quite small and feedback terms should be extracted from relevant portions only. Another assumption behind these approaches is that even non-relevant documents can contain passages with useful feedback terms [6].

In contrast, our experiments for the RF track at INEX 2010 aim at investigating if *true* relevant text passages also contain noise so that a segmentation into smaller textual units (in this case: word windows) will improve IR effectiveness. The motivation behind our method is the assumption that even large *true* relevant text passages contain harmful terms for QE. Furthermore, the RF track provides the opportunity to explore how to use true relevant passages for QE.

LCA [7] involves decomposing the feedback documents into fixed length word windows to overcome the problem of choosing terms from unrelated portions of a long document. The word windows are ranked by a score which depends on the co-occurrence of a word with the query term. Similar to LCA, we presume that terms in close proximity to query terms are good candidates for QE. In our RF method, we select feedback terms from word windows which are maximally similar to the query, the similarity being measured by Lucene's default similarity which is a variant of $\text{tf} \cdot \text{idf}$, thus achieving the same effect of filtering out potentially irrelevant parts of a longer document as in LCA. A major difference with respect to LCA is that we do not compute term co-occurrences explicitly.

¹ We employ the term segment in its most general sense, denoting sentences, paragraphs, and other small text units such as word windows.

3 System Setup

In contrast to other evaluation tracks in IR, submissions to the RF track comprise of an implemented software module (a JAVA .jar file). We submitted 3 RF modules for the RF track at INEX 2010. As a baseline, we use standard Rocchio feedback [8], which was packaged as a default feedback module implementation by the INEX organizers. We use the Lucene API² for indexing and retrieval. The RF track simulates a user highlighting relevant passages if any for each document presented to him. The feedback module re-ranks the initial results based on relevance information.

The Term Selection Algorithm. We propose the following basic algorithm for RF. Three variations of this algorithm are realized by choosing the terms t_i in different ways (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. Tokenize the string R into words and break it up into fixed length windows of m words after applying stopword removal and stemming.
4. Let $\mathbf{tf}(t_i)$ be the term frequency of the i^{th} term in the window and $\mathbf{idf}(t_i)$ the inverse document frequency of t_i .
For each window $\mathbf{w} = (w_1, \dots, w_n)$, where $w_i = \mathbf{tf}(t_i)^{\frac{1}{2}} \log \mathbf{idf}(t_i)$, compute the cosine similarity of \mathbf{w} with $\mathbf{q} = (q_1, \dots, q_n)$, where $q_i = \mathbf{tf}(t_i)$.
5. Rank all windows w by similarity score and choose top p windows.
6. Extract the most frequent T terms from these windows and add them to the query. The three variants for choosing T terms are as follows:
 - RF_{const} : $T = t$, where t is a constant $\forall i$.
 - RF_{invsrl} : $T = \frac{(L_i - r_i)}{L_i} t$, 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.
 - RF_{rsl} : $T = \frac{r_i}{L_i} t$, with t , L_i and r_i defined as before.
7. Re-retrieve with the expanded query and return the topmost similar document not returned previously.

The first variant (RF_{const}) chooses a constant number of terms regardless of the segment length. For RF_{invsrl} and RF_{rsl} the number of terms added is inversely and directly proportional to the length of the relevant section, which corresponds to choosing a greater number of terms from shorter relevant segments, and choosing a smaller number of terms from shorter segments, respectively.

The default Rocchio feedback implementation serves as a baseline with parameters $(\alpha, \beta, \gamma) = (1, 0.75, 0)$ to weight original terms, positive, and negative feedback terms. The Rocchio feedback uses $T = 20$ terms for query expansion.

Two major differences between the baseline module and our implementation are: a) the baseline method adds expansion terms to the original query at each iteration, whereas we add expansion terms for the i^{th} iteration obtained during

² <http://www.apache.org/dyn/closer.cgi/lucene/java/>

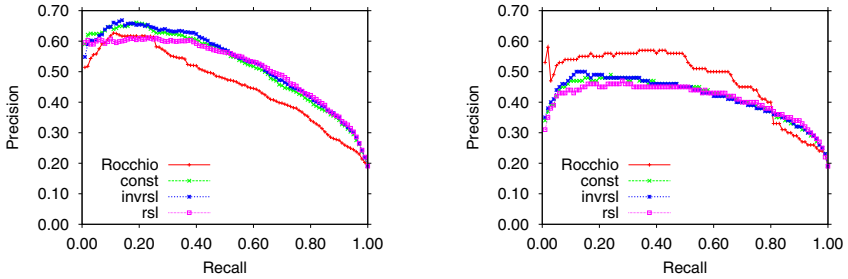


Fig. 1. Interpolated Precision-Recall graphs for INEX 2008 (left) and 2010 data (right)

the $(i-1)^{th}$ iteration; b) the step-size of the incremental feedback for the baseline method is 5, i.e. it expands the original query after every 5 iterations whereas our method uses a step-size of 1, i.e. we update the query after every iteration. Thus, our query expansion accumulates terms at every retrieval iteration in contrast to the baseline method, which generates a new query in each iteration. This is in contrast to the “save nothing” strategy [9] which was shown to be ineffective for incremental feedback.

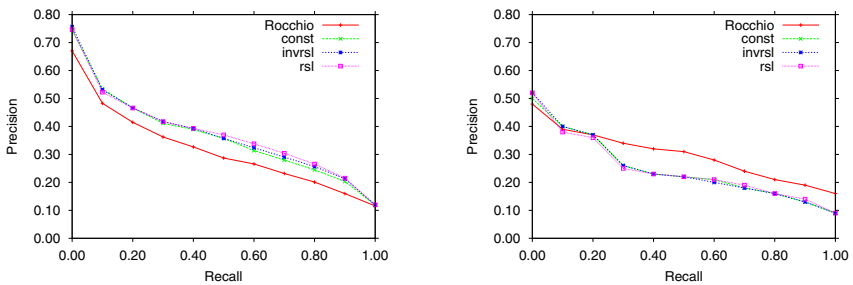
Training the System. The parameters as outlined in the feedback algorithm are the window length m , the number p of most similar windows to restrict the expansion terms to, and the variable T , which represents the number of feedback terms. After conducting a range of experiments on INEX 2008 topic set we chose the optimal settings of $(m, p, T) = (30, 10, 5)$. The results of these training experiments with the above settings are outlined in Table 1. Wilcoxon tests on the 11 point precision-recall curves reveal that the improvements for the three proposed methods over $\text{RF}_{\text{Rocchio}}$ are statistically significant.

4 Results and Analysis

For our official submission to the RF track, we used the optimal parameters obtained from INEX 2008 training topics. The graphs of Figure 1 show the document level interpolated precision-recall curves for the four approaches on INEX 2008 and 2010 topics. The graphs of Figure 2 show the interpolated geometric means of per-topic precision values measured at 11-point recall levels on 2008 and 2010 data. The left graph of Figure 1 reveals some interesting characteristics of the two feedback methods RF_{rs1} and $\text{RF}_{\text{invrs1}}$. While it can be seen that RF_{rs1} yields low precision for lower levels of recall, it outperforms $\text{RF}_{\text{invrs1}}$ for higher levels of recall which suggests that it might be worth trying a combination of the above two techniques as a part of our future work. The right graph of Figure 1 suggests that $\text{RF}_{\text{Rocchio}}$ starts off with a better precision and outperforms the focused methods until a recall level of 80% is reached. For the focused methods, although the initial retrieval precision (precision at less than 10% recall level) is

Table 1. Results for Relevance Feedback on INEX 2008 training topics

Methodology	Evaluation Metric		
	MAP	GMAP	MAiP
No feedback	0.3610	0.3087	0.3952
RF _{Rocchio}	0.4744	0.4292	0.5011
RF _{const}	0.5366	0.4687	0.5519
RF _{invrsl}	0.5442	0.4805	0.5611
RF _{rsl}	0.5307	0.4596	0.5477

**Fig. 2.** RIC curves for INEX 2008 (left) and INEX 2010 data (right)

lower, precision picks-up steadily and does not suffer from a steep down-hill as observed for the Rocchio method. For the RIC metric, we see a different trend for the INEX 2010 topics. The focused methods have a higher precision (thus suggesting that it is more appropriate for precision oriented retrieval tasks such as the focused task) at recall levels of less than 20% after which the Rocchio feedback outperforms each of them.

The fact that the focused RF methods are outperformed by the Rocchio feedback as measured by the standard document level retrieval metric MAP, leads to the question of what changes in the characteristics of the topics and the relevant set, if any, from 2008 to 2010, caused this trend reversal. The corresponding answers should explain the differences in the training results and the official submissions. A possibility is that the average length of relevant passages (average being computed by accumulating the number of relevant characters per document averaged over the number of topics) for the INEX 2010 topic set is higher (453.6 characters) as compared to INEX 2008 (409.9 characters), which means that further reducing the length of relevant passages may be required. This suggests using smaller values for the number of windows, e.g. decreasing p , could possibly improve results.

5 Conclusions and Future Work

For our participation in the RF track at INEX 2010, we implemented a new feedback method which selects feedback terms from maximally similar word windows extracted from reported relevant text passages.

The proposed method significantly outperforms the baseline Rocchio feedback method on the INEX 2008 training data and yields better precision at early recall levels when measured with the RIC metric, but does not show improvement for the INEX 2010 data when evaluated with MAP.

Future work includes optimizing feedback parameters m and p keeping in mind that the average length of relevant segments is higher for INEX 2010 topics. In addition based on the observation from INEX-2010 results that Rocchio gives better precision at early recall levels and our method gives better precision at higher recall levels, we plan to explore a combination of feedback strategies, i.e. selecting or switching the feedback strategies at some retrieval iteration.

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. Callan, J.P.: Passage-level evidence in document retrieval. In: SIGIR 1994, pp. 302–310. ACM/Springer (1994)
2. Allan, J.: Relevance feedback with too much data. In: SIGIR 1995, pp. 337–343. ACM Press, New York (1995)
3. Ganguly, D., Leveling, J., Jones, G.J.F.: Exploring sentence level query expansion in the language model. In: Proceedings of ICON 2010, pp. 18–27 (2010)
4. Murdock, V.: Aspects of Sentence Retrieval. PhD thesis, University of Massachusetts - Amherst (2006)
5. Losada, D.E.: Statistical query expansion for sentence retrieval and its effects on weak and strong queries. *Inf. Retr.* 13, 485–506 (2010)
6. Wilkinson, R.: Effective retrieval of structured documents. In: SIGIR 1994, pp. 311–317. Springer-Verlag Inc., New York (1994)
7. Xu, J., Croft, W.B.: Query expansion using local and global document analysis. In: SIGIR 1996, pp. 4–11. ACM, New York (1996)
8. Rocchio, J.J.: Relevance feedback in information retrieval. In: The SMART retrieval system – Experiments in automatic document processing, pp. 313–323. Prentice-Hall, Englewood Cliffs (1971)
9. Allan, J.: Incremental relevance feedback for information filtering. In: SIGIR 1996, pp. 270–278. ACM, NY (1996)

Combining Strategies for XML Retrieval

Ning Gao¹, Zhi-Hong Deng^{1,2,*}, Jia-Jian Jiang¹, Sheng-Long Lv¹, and Hang Yu¹

¹ Key Laboratory of Machine Perception (Ministry of Education),
School of Electronic Engineering and Computer Science, Peking University

² The State Key Lab of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, China
{ninggao,zhdeng,jiangjiajian}@pku.edu.cn,
{davidfracs,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 Two-Step retrieval, MAXLCA query results, BM25, distribution measurements and learn-to-optimize method are combined to form a more effective search engine. In Data Centric track, to gain a set of closely related nodes that are collectively relevant to a given keyword query, we promote three factors, correlation, explicitness and distinctiveness. In Relevance Feedback track, to obtain useful information from feedbacks, our implementation employs two techniques, a revised Rocchio algorithm and criterion weight adjustment.

Keywords: INEX, Ad Hoc, Data Centric, Relevance Feedback.

1 Introduction

INEX Ad Hoc Track [1] aims at evaluating performance in retrieving relevant results (e.g. XML elements or documents) to a certain query. In Ad Hoc 2010, four different tasks are addressed: (1) Efficiency task requires a thorough run to estimate the relevance of documents components. (2) Relevant in Context task requires a ranked list of non-overlap XML elements or passages grouped by their corresponding parent articles. (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 elements or passages. And for each element a ranked list result covers its relevant material, at most 500 characters for each result document.

Initially we only consider the effectiveness of retrieval, regardless of the efficiency and restricted length. Five different querying strategies, BM25 [2], Two-Step retrieval, Maximal Lowest Common Ancestor (MAXLCA) [3] query results, distribution measurements and learn-to-optimize method, are combined to form a more efficient search engine. Detailed definitions of these technologies is introduced in section 2. The thorough retrieval results are submitted to efficiency task. Furthermore, the results for other

* The corresponding author.

three tasks are all obtained based on thorough run task. 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 with top relevance-length ratio, in which relevance score for each results are computed by the aforementioned combined strategies module. For Relevance in Context task, the process scans the thorough runs and integrates the elements belonging to the same document. The orders for these integrated elements 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 is pruned to maximal 500 characters. Similar to the Restricted Focused task, only passages with top relevance-length ratio are retrieved.

Data Centric track is to provide a common forum for researchers or users to compare different retrieval techniques on data-centric XML documents, whose structure is plentiful and carries important information about objects and their relationships. In order to generate a good snippet [4] 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: (1) whether the attribute is distinguishable or not; (2) whether the attribute is explicit or implicit; (3) whether the attribute describes the entity directly or indirectly.

Relevance feedback [1] is a new track in INEX 2010. IR system with relevance feedback permits interactivities between the users and the system. Users provide relevance (irrelevance) information of search result to IR system, which is utilized by IR system to return more effective results. Relevance feedback track in INEX2010 simulates a single user searching for 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 result 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. Consequently, our team pay more attention to acquiring more information through feedback rather than optimizing results as what we did 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, the methods applied for Relevance Feedback track are presented.

2 Ad Hoc Track

Figure 1 describes the framework of our search engine. The Inverted Index of the Data Collection is initially processed in the background. Afterwards, when a user submits a Query to the Interface, the search engine first retrieves the relevant documents. Based on the definition of MAXimal Lowest Common Ancestor (MAXLCA) and All Common Ancestor (ACA) query results, the relevant elements are extracted from the relevant documents. Furthermore, we use the ranking model BM25 to rank these extracted disordered element results, the output of the processing on a Ranked List. To further improve the effect of the ranking module, the top several results in the Ranked List are re-ranked by Distribution Measurements. In Distribution Measurements, there are four criterions based on the distribution of keywords that are taken into consideration, explicitly introduced in section 4. The weights of these four criterions in the re-ranking function

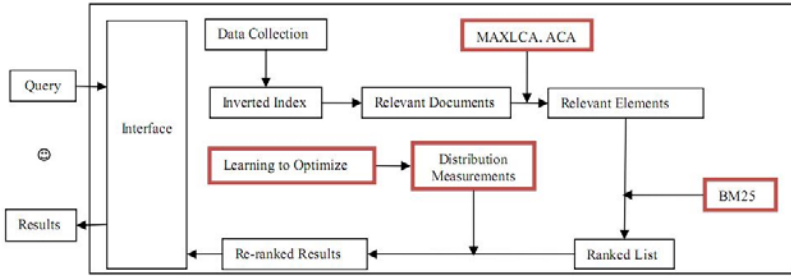


Fig. 1. System Framework

are trained by a Learning to Optimize method. Finally, the Re-ranked Results are returned to the user as searching Results. In the search engine, five different strategies are used, Two-Step search, MAXLCA query results, BM25, Distribution Measurements and Learn to Optimize method.

- **Two-Step Retrieval:** Different from HTML, the retrievable unit for the INEX focused task is XML elements rather than the whole document text. Therefore, the core idea of Two-Step retrieval is splitting the searching process into two steps. The first level starts from the traditional article retrieval. Then taking the top 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. In INEX 2009, one of the most effective search engines proposed by Queensland used Two-Step as retrieval strategy [19].
- **BM25:** Based on various research and comparative experiments, BM25 is confirmed to be an effective ranking method. It takes both text and structure information into consideration. Additionally, evaluation results of Ad Hoc Track show that BM25 performs better than some other frequently cited ranking models, such as $tf*idf$ [5] etc. Motivated by BM25's excellent performance, we implant it into the search engine as a basic ranking method.

In the remainder of the section, we apply other three technologies in the search engine. MAXLCA and ACA defines which elements are relevant and to be returned as results. Distribution measurements are re-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 the fact that the returned results of XML retrieval are elements, an adaptive XML search engine should define which elements in the XML tree are relevant and to be retrieved. Several approaches have been proposed for identifying relevant results, such as XRANK [6], SLCA [7] and XSeek [8] and so on. In this paper, our definition of query results are called MAXLCA. Furthermore, we compare it with another widely used definition of query results, naming All Common Ancestor (ACA).

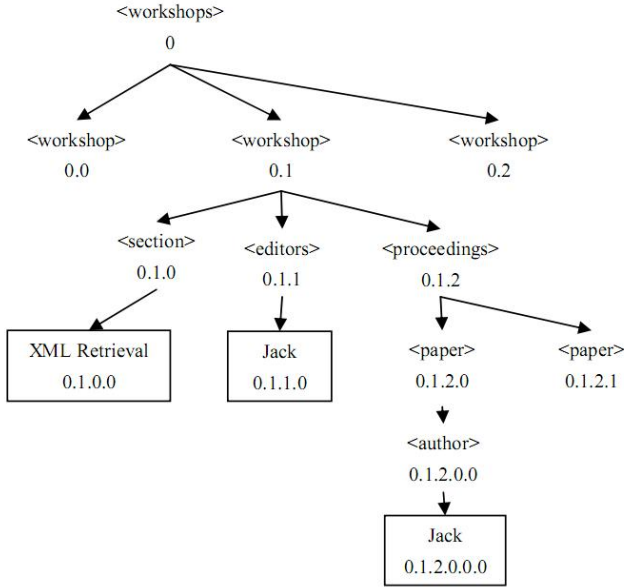


Fig. 2. Sample XML Tree

- **ACA:** In an XML tree, the nodes containing all matches of keywords are returned as relevant results. For example, there is a submitted query {XML retrieval, Jack} and an XML tree presented as figure 2, in which the matches of keywords have been marked. Node 0 and 0.1 contains all the three matches in the tree, so that these two nodes should be returned.
- **MAXLCA:** In an XML tree, the lowest node that contains all matches of keywords is defined as a query result. For example, within the nodes containing all matches of keywords in XML tree, 0.1 is the lowest one. Therefore, node 0.1 is the unique query result.

2.2 Distribution Measurements

By observing a large amount of over 2000 query-result pairs, we 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, which present advantage capability on distinguishing relevant and irrelevant passages.

- **Distance Among Keywords (DAK).** Zhao etc. considers the proximity of query terms in [20]. Here in this paper the minimum distance among the keywords is calculated. The closer the matches of keywords, the more relevant an element is.
- **Distance Among Keyword Classes (DAKC).** Xu and Croft discuss term clustering in [21]. In this paper, the matches of a certain keyword in passages are firstly clustered into several subsets. The closer the keywords subsets are, the more relevant this passage is.

- **Degree of Integration Among Keywords (DIAK).** The passage with higher degree of integration is considered as more concentrating on one certain theme and should be given a higher priority in the returned list.
- **Quantity Variance of Keywords (QVK).** The passages whose numbers of different keywords vary significantly should be penalized.

2.3 Learn to Optimize the Parameters

Neural Network method in machine learning is introduced 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 $Q = \{q_1, q_2, \dots, q_m\}$ extracted from the INEX 2009 Ad Hoc track. Each query q_i is associated with a list of candidate elements $E_i = (e_i^1, e_i^2, \dots, e_i^{n(i)})$, where e_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 $G_i = (g_i^1, g_i^2, \dots, g_i^{n(i)})$, indicating the relevance score of each elements in E_i . Given that the wiki collection only contains information of whether or not the passages in a document is relevant, the F-measure [22] is applied to evaluate the ground truth score. Given a query q_i , the ground-truth score of the j -th candidate element is defined as follows:

$$precision = \frac{relevant + irrelevant}{relevant} \quad (1)$$

$$recall = \frac{relevant}{REL} \quad (2)$$

$$g_i^j = \frac{(1 + 0.1^2) \cdot precision \cdot recall}{0.1^2 \cdot precision + recall} \quad (3)$$

In the formula, *relevant* is the length of relevant contents highlighted by user in e , while *irrelevant* stands for the length of irrelevant parts. *REL* indicates the total length of relevant contents in the data collection. The general bias parameter is set as 0.1, denoting that the weight of precision is ten times as much as recall.

Furthermore, for each query q_i , we use the distribution criterions defined in section 2.2 to get the predicted relevant scores of each candidate element, recorded in $R_i = (r_i^1, r_i^2, \dots, r_i^{n(i)})$. In formula (4), 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.

$$r_i^j = \alpha S_{DAK} + \beta S_{DAKC} + \gamma S_{DIAK} + \delta S_{QVK} \quad (4)$$

Then each ground truth score list G_i and predicted score list R_i form a “instance”. The loss function L is defined as the Euclidean distance between standard results lists D_i and search results lists R_i . 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 when reaching the limit cycle index or the parameters do not change. Precise descriptions see [9].

2.4 Comparison Results

According to (1) two-step strategy or simple element retrieval; (2) ACA results or MAXLCA results; (3) BM25, Distribution or BM25+Distribution, there should be 12 kinds of combination methods altogether. However, due to some previous experiments, we only submit 5 different combinations which are predicted as effective to Ad Hoc track, illustrated in table 1.

Table 1. Results Submitted to Ad Hoc Track

	MAXLCA	ACA	Two-Step	BM25	Distribution
AcaBM25		✓		✓	
MaxBM25	✓			✓	
RefMaxDis	✓		✓		✓
RefMaxBM25	✓		✓	✓	
RefMaxBM25Dis	✓		✓	✓	✓

Figure 3,4,5 illustrate the evaluation results of Efficiency task, Relevance In Context task and Restricted Relevance In Context task respectively under measure as focused retrieval. For Restricted Focused task, since we only submitted the RefMaxBM25Dis results, there is no useful and convincing comparison results that can be shown. However, as can be concluded from the other three tasks:

- Two-Step search performs better than simple element search. According to tradition ranking methods, such as BM25 and $tf*idf$, the elements with high ($tf / \text{passage length}$) ratio are marked as high relevance, emphasizing on small fragments even if they are extracted from irrelevant documents. Two-Step strategy eliminates such bias, since the candidate elements are all extracted from documents predicted to be relevant.
- MAXLCA performs better on wiki collection than ACA. Though according to our experiments on the data collection of INEX2010 and INEX2009, the MAXLCA results perform better than ACA results, we still support the definition of ACA for its apparently high flexibility. On the other hand, the definition of MAXLCA is only suitable for short documents, such as web pages.
- Rather than completely abandoning BM25, distribution measurement is suitable for improving the performance and modifying the drawbacks of it. Our approach originally aims at modifying the disadvantage parts of BM25, since it has been proved effective by many searching systems in INEX. The distribution measurement is a re-ranking method, where each standards only focuses on one single point. Accordingly, we use a learning method to learn the optimal weights of these standards for a certain data collection and only in this way, the final re-ranking method are actually determined by the data collection.
- The method using Two-Step as retrieving strategy, MAXLCA as query results, BM25 and distribution measurement as ranking functions shows the best performance.

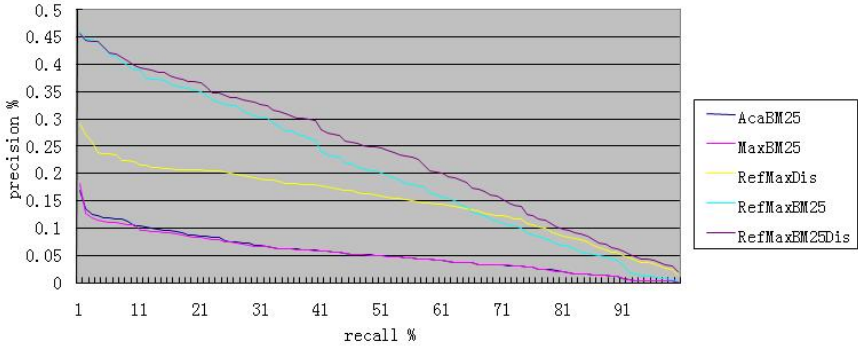


Fig. 3. Evaluation Results of Efficiency Task

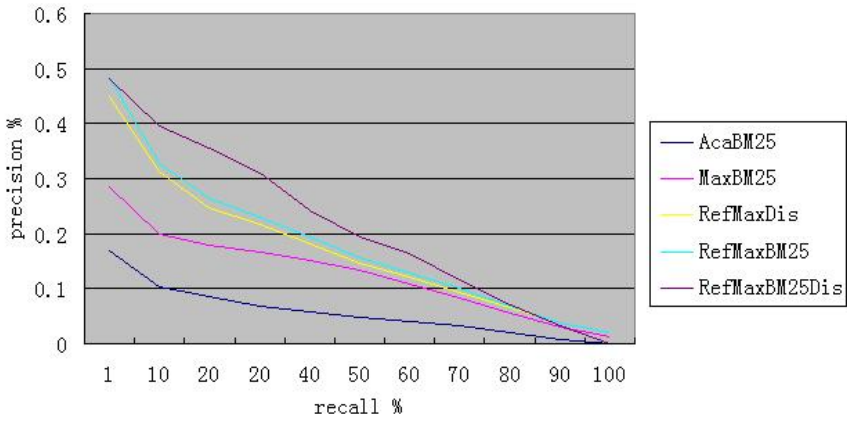


Fig. 4. Evaluation Results of Relevance in Context Task

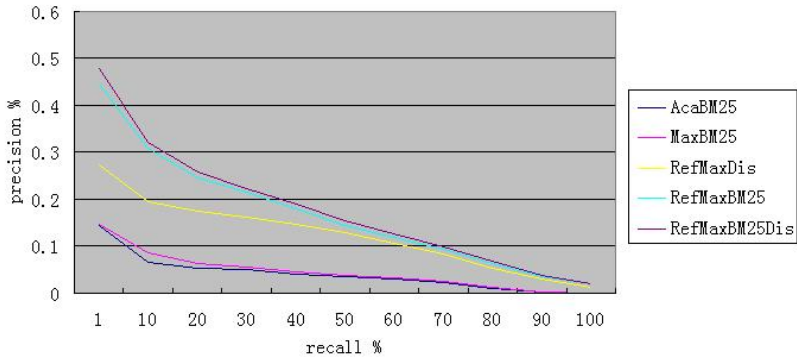


Fig. 5. Evaluation Results of Restricted Relevance in Context Task

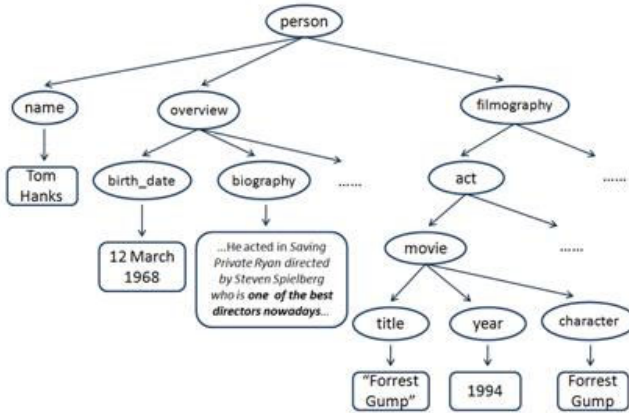


Fig. 6. IMDB data segment

3 Data Centric Track

In XML keyword search, there are quite numbers of results returned, only by the snippet of each result can the users judge whether it meets the search intention. Therefore, it is important to generate a good snippet for each result. [8] [10] 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.

3.1 Related Work

[10] [11] pointed out 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 [4]. The size limitation of a snippet requires 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.

3.2 Extracting the Most Representative Attributes

In this paper, we use a semantic model MRepA (Most Representative Attribute) [12] to evaluate the importance of an attribute to its corresponding entity. Further, three main principles are proposed to judge the importance of attributes, presented as, (1) whether the attribute is distinguishable or not; (2) whether the attribute is explicit or implicit; (3) whether the attribute describes the entity directly or indirectly.

3.2.1 Correlation between Entities and Attributes

To evaluate whether an attribute describes an entity directly, we analyze the 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 *correlation* between an attribute and its corresponding entity.

Definition 1. An entity-attribute path is a set of nodes which are on the path from an entity to its attribute (including the entity and the attribute).

Definition 2. We define the correlation between entity e and attribute a $R(e, a)$ as follows

$$R(e, a) = k^{\text{length}(e,a)} \cdot \prod_{i=1}^n \frac{1}{m_i} \quad (5)$$

Where $n = \text{length}(e, a)$ 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. k is a parameter set less than 1.

For example, in figure 4, suppose that there are 10 movie nodes, the entity-attribute path from node person to title *Forrest Gump* is {person, filmography, act, movie, title}. There are two entities *person* and *movie* on the path. The number of person is 1, while the number of the movie is 10.

3.2.2 Explicitnesses of Attributes

In XML keyword search, the length of the value of an attribute is usually associated with the explicitness of the attribute. 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. We judge the explicitness of an attribute by the complicacy (length) of its value, and we denote the explicitness of attribute a as $E(a)$.

3.2.3 Distinctiveness of Attributes

A distinguishable attribute should match the following two conditions, (1) the attribute appears in all of the entities; (2) the values of the attribute are different in different entities. Due to the possibility that two different person share the same name, in this section we promote the formula to calculate how much an attribute meets the demands.

Definition 4. We use distinctiveness of attributes to evaluate the distinguish ability of the attributes.

$$W_a = \exp(p_a) \cdot H(a) \quad (6)$$

$$H(a) = - \sum_{i=1}^n p(a_i) \cdot \log[p(a_i)] \quad (7)$$

In the above formulas, W_a is the distinctiveness of attribute a . p_a refers to the percentage of the correlative entities where attribute a appears. $H(a)$ is the entropy of attribute a , which estimates the variety intensity of attribute a .

3.2.4 MRepA Model

In MRepA model, we take the above three factors into consideration. Given a keyword query, we firstly return a set of entities as results, and then pick up the top- k most important attributes of each entity into the snippet.

Definition 5. *The importance of an attribute a to an entity e $S(e,a)$ is defined as follows*

$$S(e,a) = W_a \times E(a)^{R(e,a)} \quad (8)$$

3.2.5 Comparison Results

In Data Centric track, there are three assessments, TREC MAP metric, INEX thorough retrieval MAiP metric and INEX Relevant-in-Context MAgP T2I(300). In MAP metric, our results perform the best, using the description and narrative of the topics as extra information. However, poor responses are got under MAiP and MAgP metric.

Table 2. Results Submitted to Data Centric Track

	MAP	MAgP	MAiP
MRepA	0.5046	NA	0.0157

4 Relevance Feedback Track

In relevance feedback track, we employs two techniques, a revised Rocchio algorithm and criterion weight adjustment. In section 4.1, we briefly introduce Rocchio algorithm [13]. In section 4.2 the revised algorithm is proposed. We discuss the adjustment of the criterion weights in section 4.3.

4.1 Rocchio Algorithm

Rocchio algorithm operates on vector space model, in which a document is represented by a vector of n weights such as $d = (t_1, t_2, t_3, t_4, \dots, t_n)$. Where n is the number of unique terms in document collections and t_i is the weight of the i -th term in document d . The keyword query is also presented as a vector.

The general idea of Rocchio algorithm is that the initial query may not express the purpose of a IR system user completely and effectively. Rocchio algorithm's goal is to define an optimal query that maximize the difference between the average vector of the relevant documents and the average vector of the irrelevant documents. To achieve this, Rocchio algorithm adds new query terms and re-weight 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

$$Q_t = \alpha Q_0 + \beta \frac{1}{n_1} \sum_{i=0}^{n_1} R_i - \gamma \frac{1}{n_2} \sum_{i=0}^{n_2} S_i \quad (9)$$

where Q_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, S_i is a irrelevant document vector, and α, β, γ control the influence of each part.

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 get relatively low weight if they appear in both relevant and irrelevant documents and have less discriminative power.

Some researchers have modified and extended the formula such as assigning different weight to original query terms [14] and added query terms or make constraints of number of documents used in the modification [15]. There are some other feedback methods based on probabilistic model and language model. The feedback method in classical probabilistic models is to select expanded terms primarily based on Robertson and Sparck-Jones Weight [16]. In language model, Zhai et al [17] estimate a model for all relevant documents together and expand original query model with it. There are also many other methods directly estimating a relevance model with relevance document [18].

4.2 Revised Rocchio Algorithm

Due to the fact that relevant information about each part of a relevant document is accessible in INEX2010 we divided a document into several paragraphs (we use " $\langle p \rangle$ " and " $\langle /p \rangle$ " to identify 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. We also assign a weight to each paragraph according to its length. The score of a document is the weighted sum of its paragraphs' scores. In our implementation, we define query expansion formula as

$$Q_t = Q_0 + \frac{1}{n} \sum_{i=0}^n P_i \quad (10)$$

Where P_i donates the vector of term weights calculated from a paragraph. For term t_j in P_i , we define its weight as follows

$$w_j = \begin{cases} score_{P_i} & \text{if } P_i \text{ is a relevant paragraph in relevant document} \\ 0 - score_{P_i-t_j} & \text{if } score_{P_i} \text{ is a paragraph in irrelevant document} \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

Where $score_{P_i}$ denotes the score of paragraph P_i , and $score_{P_i-t_j}$ denotes the score of paragraph after removing all t_j from it.

Here we 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 preparing a presentation about the *Nobel Prize*. Therefore, he or she wants to collect information about *Nobel Prize* as much as possible. Assume that 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 because it only contains three words and two of them are keywords. Intuitively, we can figure out that the term *Ig* is a bad word for this topic since it turns

a high-score paragraph to a irrelevant one. In addition, term *Nobel* or *Prize* has no contribution to the irrelevance of this paragraph. However, if we use the formula in section 4.1, no difference between I_g and I_{ir} is reflected in the values of R_i or S_i . While in the revised model, the weights of I_g 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 belongs to.

4.3 Criterion Weight Adjustment

To calculate score of a paragraph, we make three criterions, 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 the alternation of keywords. It assigns low score to a paragraph if it talks about one of the keyword at beginning while talks about another keyword at the end without a mixture of them. Each criterion makes contrition to the final score of a paragraph.

However, it is hard 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. When updating the criterion weights, the discriminative power of each criterion and each keyword are computed. The discriminative power is computed as follows

$$DP = \frac{(\mu_r - \mu_{ir})^2}{d_r^2 - d_{ir}^2} \quad (12)$$

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 indicates a criterion of keyword that is not suitable for the 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* is raised to 0.89 but the *Prize* is only 0.11. This is understandable. *Prize* is relatively a more widely used word because there are a lot of prizes such as Fields Medal Prize, Turing Prize.

Acknowledgments. This work was supported by the National High-Tech Research and Development Plan of China under Grant No.2009AA01Z136.

References

1. <http://www.inex.otago.ac.nz/>
2. Carmel, D., Maarek, Y.S., Mandelbrod, M., et al.: Searching XML documents via XML fragments. In: SIGIR 2003, pp. 151–158 (2003)

3. Gao, N., Deng, Z.H., Jiang, J.J., Xiang, Y.Q., Yu, H.: MAXLCA A Semantic XML Search Model Using Keywords. Technical Report
4. Huang, Y., Liu, Z., Chen, Y.: eXtract: A Snippet Generation System for XML Search. In: VLDB 2008, pp. 1392–1395 (2008)
5. Theobald, M., Schenkel, R., Wiekum, G.: An Efficient and Versatile Query Engine for TopX Search. In: VLDB 2005, pp. 625–636 (2005)
6. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: Ranked Keyword Search over XML Documents. In: SIGMOD 2003, pp. 16–27 (2003)
7. Xu, Y., Papakonstantinou, Y.: Efficient Keyword Search for Smallest LCAs in XML Databases. In: SIGMOD 2005, pp. 537–538 (2005)
8. Liu, Z., Chen, Y.: Identifying Meaningful Return Information for XML Keyword Search. In: SIGMOD 2007, pp. 329–340 (2007)
9. Gao, N., Deng, Z.H., Yu, H., Jiang, J.J.: ListOPT: A Learning to Optimize Method for XML Ranking. In: PAKDD 2010 (2010)
10. Liu, Z., Chen, Y.: Identifying Meaningful Return Information for XML Keyword Search. In: SIGMOD 2007, pp. 329–340 (2007)
11. Huang, Y., Liu, Z.Y., Chen, Y.: eXtract: A Snippet Generation System for XML Search. In: VLDB 2008, pp. 1392–1395 (2008)
12. Jiang, J., Deng, Z.H., Gao, N., Lv, S.L., Yu, H.: MRepA: Extracting the Most Representative Attributes in XML Keyword Search. Technical Report
13. Ruthven, I., Lalmas, M.: A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review* 18(2), 95–145 (2003)
14. Ide, E.: New experiments in relevance feedback. In: Salton, G. (ed.) *The SMART Retrieval System Experiments in Automatic Document Processing*, ch. 16, pp. 337–354 (1971)
15. Ide, E., Salton, G.: Interactive search strategies and dynamic file organization in information retrieval. In: Salton, G. (ed.) *The SMART Retrieval System - Experiments in Automatic Document Processing*, ch.18, pp. 373–393 (1971)
16. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms. *Journal of the American Society of Information Science* 27(3), 129–146 (1976)
17. Zhai, C., Lafferty, J.D.: Model-based feedback in the language modeling approach to information retrieval. In: *CIKM 2001*, pp. 403–410 (2001)
18. Lavrenko, V., Bruce Croft, W.: Relevance-based language models. In: *SIGIR 2001*, pp. 120–127 (2001)
19. Geva, S., Kamps, J., Lethonen, M., Schenkel, R., Thom, J.A., Trotman, A.: Overview of the INEX 2009 ad hoc track. In: Geva, S., Kamps, J., Trotman, A. (eds.) *INEX 2009*. LNCS, vol. 6203, pp. 4–25. Springer, Heidelberg (2010)
20. Zhao, J., Yun, Y.: A proximity language model for information retrieval. In: *SIGIR 2009*, pp. 291–298 (2009)
21. Xu, J., Croft, W.B.: Improving the effectiveness of information retrieval with local context analysis. In: *TOIS 2000*, pp. 79–112 (2000)
22. van Rijsbergen, C.J.: *Information Retrieval*. Butterworths, London (1979)

Overview of the INEX 2010 Web Service Discovery Track*

James A. Thom¹ and Chen Wu²

¹ RMIT University, Melbourne, Australia
james.thom@rmit.edu.au

² University of Western Australia, Perth, Australia
chen.wu@uwa.edu.au

Abstract. 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 contributed to topic development and to the evaluation, which allows them to compare the effectiveness of their XML retrieval techniques for the discovery of Web services. This has led to the initial development of a test collection that will allow future comparative experiments.

Keywords: Web Service discovery, WSDL.

1 Introduction

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).

There were five active groups participating in the task in 2010, and they contributed to providing topics, submitting runs and assessing results.

2 WSDL Collection

The Web Service Discovery track used 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.

The original dataset for track contained 1987 separate documents, however after some duplicates were removed, a revised version of the dataset was released containing 1738 documents (original document numbering was retained).

* This work was done while the second author was at Curtin University in Perth.

3 Topics

The participating groups were asked to create a set of candidate topics, representative of a range of realistic web service discovery needs. The submitted topics were in the same format as the ad hoc track. A sample topic is shown in Fig. 1. Out of the 31 topics submitted by five groups, 25 topics were selected for the track in 2010.

```
<topic id="2010023" ct_no="30">
  <title>airline flights</title>
  <castitle>//*[about(., airline flights)]</castitle>
  <description>Given an airline flight number I would like
    to find details of the flight.</description>
  <narrative>A service for any airline that can provide
    the status of airline flights is relevant.</narrative>
</topic>
```

Fig. 1. Sample topic

4 Submissions

The submission format was the same as the ad hoc track, submission were allowed in one of three formats:

- XML elements using XPath syntax
- passages in File-Offset-Length (FOL) format
- ranges of elements (for backward compatibility with previous INEX formats)

However all groups only submitted document level runs and had difficulty following the submission format, so some formatting corrections were required.

Five groups submitted a total of 15 runs, although all runs were used to contribute documents into the pool for each topic, three runs were excluded from the final evaluation as they included more serious errors (such as duplicate answers).

5 Assessment

The pooling of documents from the submitted included approximately 100 results for each topic. As the XML structure was important for assessing whether a document (or parts of a document), the INEX evaluation tool for the ad hoc track was used but the WSDL documents' XML markup was displayed along the content of the elements. In some cases, there was little or no text content in the XML elements, so having the XML markup was essential in assessing the relevance. Of the 25 topics, only 20 were assessed.

6 Evaluation and Results

Since only document level runs were submitted, evaluation was only performed for document retrieval using Mean Average Precision as calculated by `trec_eval`. The results for the 12 valid runs are shown in Table 1, with the 11 point average precision shown in Fig. 2.

Table 1. Mean Average Precision for all runs

	map	Institute	Run
1	0.3469	Kasetsart University	Kas_I138BM25ESS010
2	0.3239	RMIT University	RMIT10WS
3	0.2946	Hasso-Plattner-Institut	HPI2010
4	0.2798	Kasetsart University	Kas_I138ANYSS025
5	0.2798	Kasetsart University	Kas_I138ANYBM25SS015
6	0.2348	Queensland University of Technology	QUT_BM25WordNetComposition
7	0.2233	Queensland University of Technology	QUT_BM25WordNet
8	0.2042	Queensland University of Technology	QUT_BM25WordNetCompositionWordNet
9	0.1451	Benemrita Universidad Autnoma de Puebla	BUAPFCCWSD01
10	0.1268	Queensland University of Technology	QUT_Wikipedia
11	0.1095	Queensland University of Technology	QUT_WikipediaComposition
12	0.0937	Queensland University of Technology	QUT_WikipediaCompositionWordNet

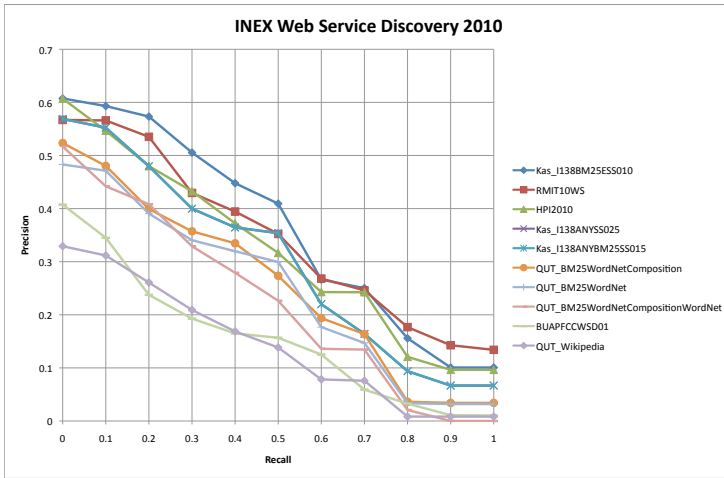


Fig. 2. Performance of 10 best Web Service Discovery runs

7 Conclusion

Now that the basics are in place for this track, we hope to get more groups participating in 2011.

Long term goal of track is given a description of a workflow (e.g. in YAWL) representing a scientific or business process, find web services that can meet the steps within the process.

Specific plans for 2011 include the following.

- Improving the collection by modifying the existing documents by extracting smaller documents from the WSDL documents to represent individual operations which will make assessments easier, and possibly adding more documents, including newer documents (e.g. more WSDL 2 documents)
- Requiring topics to be a sequence/graph of queries for a service rather than isolated information needs, which will allow two tasks: (i) document/passage retrieval (similar to 2010 but hopefully explore passage retrieval) of service components, and (ii) combining operations to meet service requirements (new).

Acknowledgements. 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.

Semantics-Based Web Service Discovery Using Information Retrieval Techniques

Jun Hou, Jinglan Zhang, Richi Nayak, and Aishwarya Bose

Faculty of Science and Technology, Queensland University of Technology
jun.hou@student.qut.edu.au,
{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. The main goal of this research is to evaluate algorithms for semantic web service discovery. The evaluation is comprehensively benchmarked using more than 1,700 real-world WSDL documents from INEX 2010 Web Service Discovery Track dataset. For automatic search, we successfully use Latent Semantic Analysis and BM25 to perform Web service discovery. Moreover, we provide linking analysis which automatically links possible atomic Web services to meet the complex requirements of users. Our fusion engine recommends a final result to users. Our experiments show that linking analysis can improve the overall performance of Web service discovery. We also find that keyword-based search can quickly return results but it has limitation of understanding users' goals.

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 unprecedented attention from the computer world. Web services can be discovered by matchmaking requirements of service requesters with providers. Web service discovery plays a key role in finding appropriate Web services. Although a number of Web services are provided by different organizations, there are still no standards for Web service design and provision. Many Web services have the same or similar functionalities but they are described in various ways. It is a challenging task to discover accurate Web services in accordance with users' requirements.

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 Web services. Web services requesters and providers then communicate with each other by SOAP message, a XML format communication language based on HTTP. The WSDL and UDDI search mechanism utilizes syntactic search based on keywords because it can return a large number of

Web services in a relatively short time. However, exact keyword match may miss actually relevant services and semantic search is proposed to enhance the search accuracy. In addition, since different organizations design services in enclosed circumstances, atomic Web services cannot satisfy different users' requirements [13]. One service can invoke other services to achieve goals with complicated requirements. Therefore, a set of Web services need to be composed to fulfill given tasks.

Two methods are used in this paper, namely Latent Semantic Analysis (LSA) 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.

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 [12]. OWL-S is more concentrated on the "Upper ontology" (not domain-specific ontology) for describing Web services [3]. Compared to OWL-S, WSMO is more focused on producing a reference implementation of an execution environment, the Web Service modeling execution environment and specifying mediators [13]. Mediators are not the significant consideration in OWL-S conceptual and implementation [17]. However, the discovery mechanism in WSMX is based on keyword and simple semantic description [17]. 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 the preferred ontology language. On top of that, the existing tool can be updated relatively easy. The objectives of WSDL-S are to be of compatibility with OWL-S with emphasizes on a more lightweight and incremental approach [14]. 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 documents into bags of words and create a term-document matrix. Then Webs services are ranked by Term Frequency–Inverse Document Frequency (TF-IDF) according to the term frequency of search query in each document. A binning & merging-based Latent Semantic Kernel [2] is proposed to enhance the semantics of LSA. The experiment result shows that the LSA approach can be acceptable both in scalability and complexity [19]. A method using surface parsing of sentences to add structural relations [4] are proposed to improve the performance on single sentences

in LSA. However, there still are some issues related to LSA. The pre-process including stop word removal and stemming reduces common terms and outliers, it also breaks WSDL structure at the same time. Nayak & Iryadi [15] and Hao & Zhang [7] 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 [8] is introduced. This approach concentrates on improving the semantic relationships between important terms by applying text clustering. Above approaches are only keyword-based and simple keywords may not represent the preference of users very well. Users' selection of services is highly impacted by non-functionality such as response time, price, throughout, availability, reliability etc.

Researchers are devoted to dig more semantic information from current Web resources. Ding, Lei, Jia, Bin, & Lun [5] 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 languages. It is limited by the scope of comment on Web services and the variety between different comment styles. Semantic Web Services Clustering (SWSC) [16] makes use of pre-conditions 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, scalability is still a problem.

3 Discovery Approach

We propose a novel three-phase approach for Web service discovery. Figure 1 shows an overview of this 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 have been executed.

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 [18] is used to strip suffix.

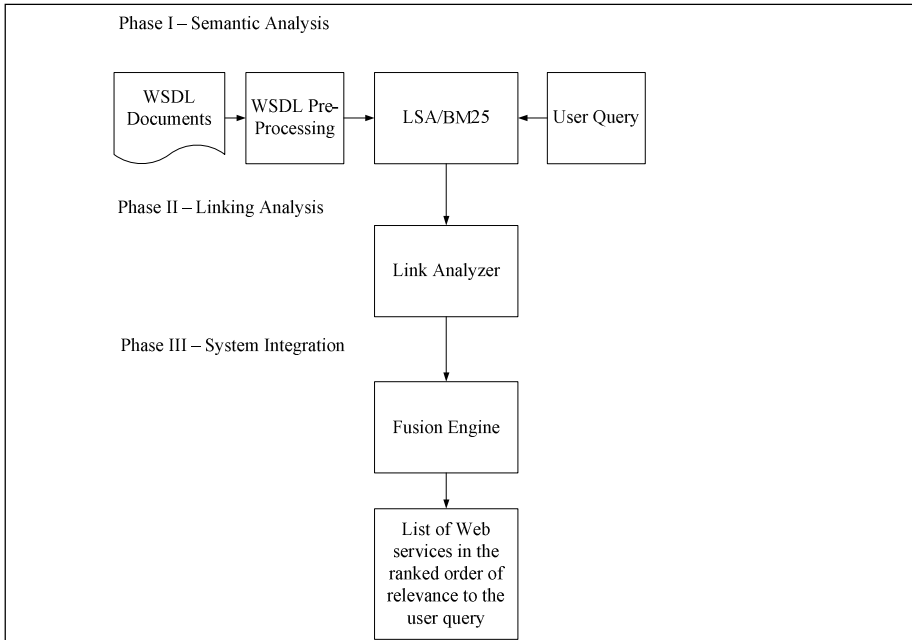


Fig. 1. Overview of Web Service Discovery Methodology

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. The wikipedia dataset [2] is chosen because it is not domain-specific and covers various topics. Figure 2 shows the overview of LSA in phase I.

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 similarity between Q and W .

$$\text{Sim}(Q, W) = \text{Cos}(Q, W) = \frac{Q \cdot W}{\|Q\| \|W\|} \quad (1)$$

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.

$$\text{Sim}(Q, W) = \text{Cos}(Q, W) = \frac{Q^T K \cdot K^T W}{\|Q^T K\| \|K^T W\|} \quad (2)$$

Finally, the top- k Web services are returned to users (k is set to 25).

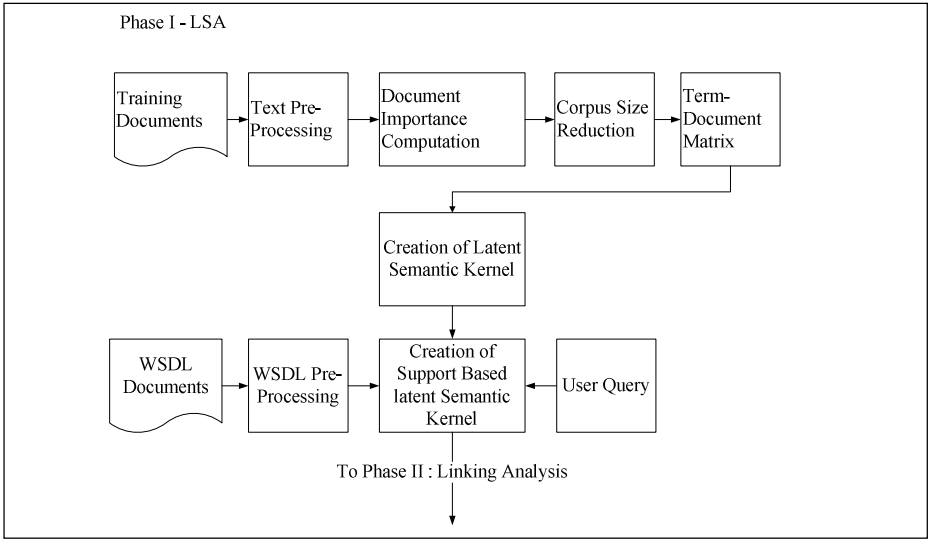


Fig. 2. Overview of LSA

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.

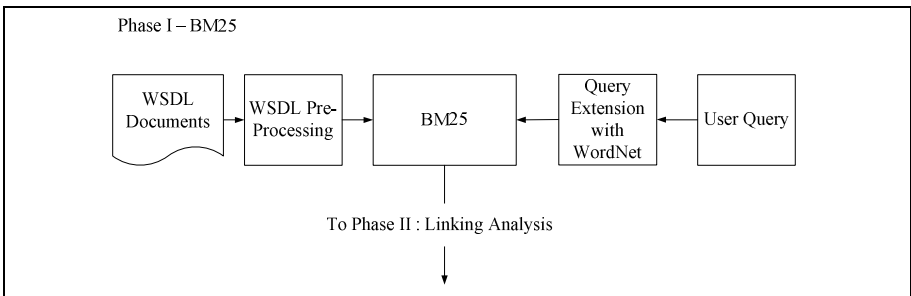


Fig. 3. Overview of BM25

After pre-processing, WSDL documents (W) are computed with users’ queries (Q) for similarity. Equation 3 shows the major equation of BM25.

$$\text{Score } (Q, W) = \sum_i^n \text{IDF } (q_i) \cdot \frac{f_{(q_i, W)} \cdot (k_i + 1)}{f_{(q_i, W)} + k_i \cdot (1 - b + b \cdot \frac{|W|}{\text{avgdl}})} \tag{3}$$

$f_{(q_i,W)}$ is q_i 's 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).

$$IDF(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \tag{4}$$

N represents the total number of WSDL documents 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 to 25).

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 incorporate 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.

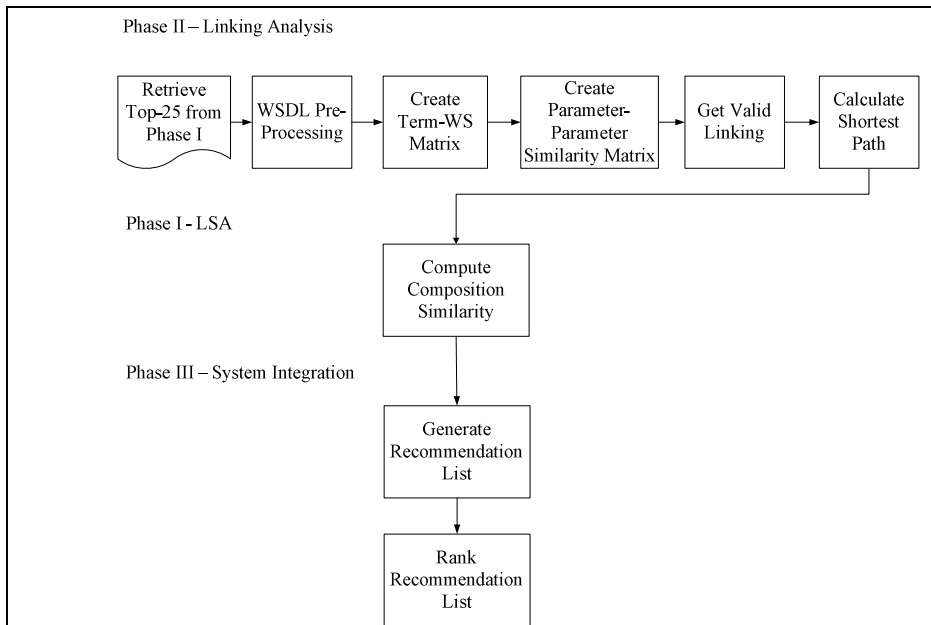


Fig. 4. Overview of Linking Analysis

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 $\langle PortType \rangle$ tag consists of sets of $\langle Operation \rangle$ 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 best 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.

$$\text{Sim}(P_1, P_2) = \frac{n \cdot \text{Sum}(\text{sim}(W_{P_1}, W_{P_2}))}{N} \quad (5)$$

$\text{Sum}(\text{sim}(W_{P_1}, W_{P_2}))$ 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 . n is the minimum of the number of parameter tokens in P_1 and P_2 . For example, if P_1 has 2 tokens and P_2 has 3 tokens, n will be 2 and N will be 5. If $\text{Sim}(P_1, P_2)$ 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.

$$\text{Link Strength} = \frac{N_L}{N_I} \quad (6)$$

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 Warshall algorithm [6] 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 as 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 results from composition of Web services with the 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 an atomic one. In addition, if a Web service is the component of a composition, it will not appear in the final result. Therefore, 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.

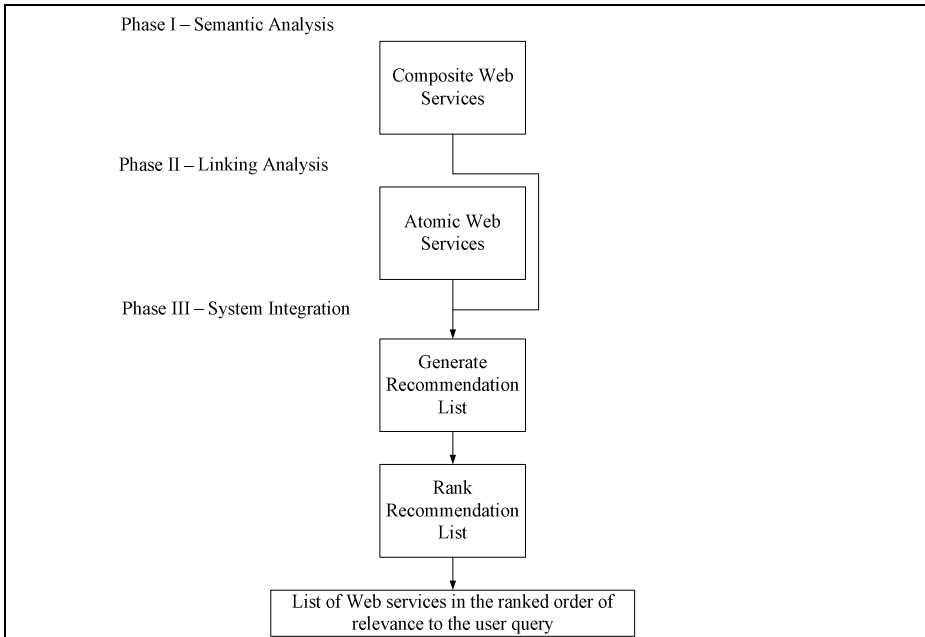


Fig. 5. Overview of System Integration

4 Data Set

The document collection is provided by the INEX 2010 organizing committee. The dataset [11] 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 from different domains. User queries are created by competition participants to ensure the variety. Figure 6 demonstrates the precision & recall curve under the query term “map”.

In Figure 6, we have four runs with the initial QUT and the best of them is QUT_BM25WordNetCompositon. In the BM25WordNetComposition run, we use BM25 supported by WordNet and the linking analysis. This submission outperforms BM25WordNet, which only applies BM25 supported by WordNet. It suggests that the linking analysis improves the accuracy of Web service discovery. The submission QUT_Wikipedia is created using LSA and it does not perform very well. One reason may be that the query term, “map”, is simple and LSA cannot find semantic services. Another reason might be that the semantic services found by LSA are not closely relevant to the query term.

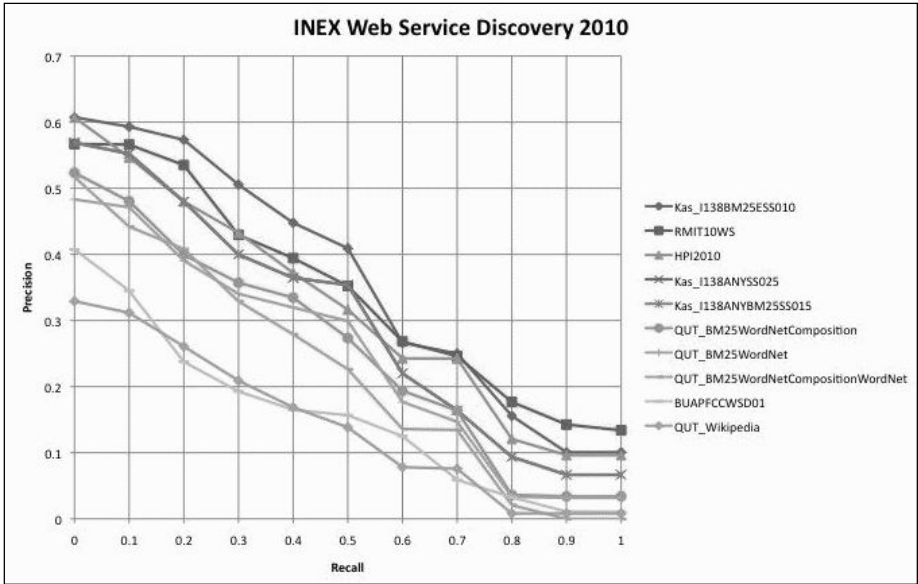


Fig. 6. INEX Web Service Discovery Results [10]

As we can see from Figure 6, the run Kas_I138BM25ESS010 (from Kasetsart University) has the highest precision when recall is less than 0.6. According to the result released from INEX, Kas_I138BM25ESS010 has the highest score, which is 0.3469 [10] with the query term “map”. Kasetsart University makes use of pre-processing techniques to boost search accuracy [9]. In our method, we only use simple pre-processing techniques due to time-constraint. Our approach can be improved by applying sophisticated pre-processing techniques. Our linking analysis is proposed under the situation of multiple query items. For example, if a user types “weather by postcode”, a combination of services “weather by city” and “city to postcode” will be retrieved. As a result, it almost has no effect with the simple query “map”. We believe our algorithm will have better performance for more complicated multiple term queries, especially for queries that can only be satisfied with the linking of multiple atomic services.

Figure 6 has also shown that even by utilizing sophisticated pre-processing techniques, the score is only 0.3469, which means there are still a lot of irrelevant retrievals. That is because textual information occupies only a small part of the overall size of WSDL documents so it is not sufficient enough to answer service queries on the single basis of query terms [1]. Moreover, WSDL documents describe the interfaces of Web services in an abstract way. The small size of textual information makes it relatively difficult to understand what the service offers. For example, sometimes, the service having a service name “map” in the service name tag may provide the map of a hospital or the map of a city. Keyword-based search with simple query term may not be suitable enough to answer users’ queries.

6 Conclusions and Future Work

The experiment result shows that both the Latent Semantic Analysis and BM25 boosted by WordNet approaches work for web service discovery. BM25 outperforms the Latent Semantic Analysis approach. Link analysis automatically composes Web services to fulfill complex tasks. Unfortunately this cannot be demonstrated by simple queries that can be satisfied by atomic services.

In this paper, Web services are converted to bags of words and then compared with users' queries for similarity. However, we find that 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, WSDL describes services in an abstract way sometimes just a single term in one tag. The single term cannot describe the function very well and investigating semantics by single words may cause more false negatives by misunderstanding the service functionality. In addition, discovering web services by considering only query terms is overly simple because 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. Non-functional parameters such as response time, price, throughout, availability, reliability etc have become significant factors on selecting services. As a result, more practical situations need to be investigated to effectively retrieve and select Web services.

References

1. Al-Masri, E., Mahmoud, Q.H.: Identifying Client Goals for Web Service Discovery. In: 2009 IEEE International Conference on Services Computing, Bangalore, India, pp. 202–209 (2009)
2. Bose, A., Nayak, R., Bruza, P.: Improving Web Service Discovery by Using Semantic Models. In: 9th International Conference on Web Information Systems Engineering, Auckland, New Zealand, pp. 366–380 (2008)
3. Burstein, M.H., McDermott, D.V.: Ontology Translation for Interoperability among Semantic Web Services. *AI Magazine* 26, 71–82 (2005)
4. Dennis, S.: *Handbook of Latent Semantic Analysis*, Mahwah, N.J (2007)
5. Ding, Z., Lei, D., Jia, Y., Bin, Z., Lun, A.: A Web Service Discovery Method Based on Tag. In: 2010 International Conference on Complex, Intelligent and Software Intensive Systems, Krakow, Poland, pp. 404–408 (2010)
6. Floyd, R.W.: Algorithm 97: Shortest Path. *Communications of the ACM* 5, 345 (1962)
7. Hao, Y., Zhang, Y.: Web Services Discovery Based on Schema Matching. In: 13th Australasian Conference on Computer Science, Ballarat, Australia, pp. 107–113 (2007)
8. Hu, J., Fang, L., Cao, Y., Zeng, H.-J., Li, H., Yang, Q., et al.: Enhancing Text Clustering by Leveraging Wikipedia Semantics. In: 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Singapore, pp. 179–186 (2008)
9. INEX Workshop Pre-proceedings (2010), <http://inex.otago.ac.nz/data/publications.asp>

10. INEX Web Service Discovery Results,
http://goanna.cs.rmit.edu.au/~jat/INEX2010_WS_results.html
11. INEX Web Service Track (2010), <http://www.inex.otago.ac.nz/tracks/webservices/webservices.asp>
12. Lara, R., Roman, D., Polleres, A., Fensel, D.: A Conceptual Comparison of WSMO and OWL-S. In: Zhang, L.-J., Jeckle, M. (eds.) *Web Services*, pp. 254–269. Springer, Heidelberg (2004)
13. Li, Q., Liu, A., Liu, H., Lin, B., Huang, L., Gu, N.: Web Services Provision: Solutions, Challenges and Opportunities (invited paper). In: *3rd International Conference on Ubiquitous Information Management and Communication*, Suwon, Korea, pp. 80–87 (2009)
14. Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., et al.: Bringing Semantics to Web Services with OWL-S. *World Wide Web* 10, 243–277 (2007)
15. Nayak, R., Iryadi, W.: XML Schema Clustering with Semantic and Hierarchical Similarity Measures. *Knowledge-Based Systems* 20, 336–349 (2007)
16. Nayak, R., Lee, B.: Web Service Discovery with Additional Semantics and Clustering. In: *9th IEEE/WIC/ACM International Conference on Web Intelligence*, Fremont, USA, pp. 555–558 (2007)
17. Shafiq, O., Moran, M., Cimpian, E., Mocan, A., Zaremba, M., Fensel, D.: Investigating Semantic Web Service Execution Environments: A Comparison between WSMX and OWL-S Tools. In: *2nd International Conference on Internet and Web Applications and Services*, Morne, Mauritius, pp. 31–36 (2007)
18. Van Rijsbergen, C.J., Robertson, S.E., Porter, M.F.: *New Models in Probabilistic Information Retrieval*. British Library, London (1980)
19. Wu, C., Potdar, V., Chang, E.: 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. Springer, Heidelberg (2009)

The BUAP Participation at the Web Service Discovery Track of INEX 2010*

María Josefa Somodevilla, Beatriz Beltrán, David Pinto,
Darnes Vilariño, and José Cruz Aaron

Faculty of Computer Science
Benemérita Universidad Autónoma de Puebla, México
{mariasg,bbeltran,dpinto,darnes}@cs.buap.mx

Abstract. A first 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. It calculates the degree of similarity between words and their relative importance to support the task of web services discovering. The first 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 information retrieval system that index terms by using an inverted index structure was also used. Both algorithms are applied in order to evaluate 25 topics in a set of 1947 real web services (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 [2]. The Web Services Description Language (WSDL)^[3] 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

* This work has been partially supported by projects: CONACYT #106625, VIEP #SOGJ-ING11-I, #BEMB-ING11-I, as well as by the PROMEP/103.5/09/4213 grant.

¹ <http://opengroup.org/projects/soa/doc.tpl?gdid=10632>

² <http://www.w3.org/TR/wsdl.html>

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 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 appropriate web service (WSDL file) to a given topic were developed.

The description of the approach that uses a clustering method follows.

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: The Maximum Matching Algorithm (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. 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.
7. Content word recognition: Thereafter, we removed the words of the cluster with minimal elements (i.e. *service*, *SOA*, *array* and *data*).
8. 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.
9. Query answering: Using the two corpus constructed, a query can be answered by applying Eq. (2), and then sorting the results from lowest to highest.

$$\text{NGD}(x, y) = \frac{\max \{ \log f(x), \log f(y) \} - \log f(x, y)}{\log M - \min \{ \log f(x), \log f(y) \}} \quad (1)$$

$$O(S_i, S_j) = 0.5 * S'(S_i, S_j) + 0.5 * S''(S_i, S_j) \quad (2)$$

where:

$$S'(S_i, S_j) = \frac{\sum_{a \in S_i} \sum_{b \in S_j} \text{Sim}(a, b)}{|S_i| |S_j|} \quad (3)$$

$$S''(S_i, S_j) = 1 - \text{NGD}(S_i, S_j) \tag{4}$$

Following we describe the approach that uses information retrieval for finding the corresponding web services files that satisfies the user needs.

The implementation based on NLP 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 the Algorithm 1 (taken from [3]).

Algorithm 1. Intersection of two posting lists

```

Input: Posting lists  $p_1$  and  $p_2$ 
Output: Relevant documents  $D_1, D_2, \dots$ 
1  $answer = \langle \rangle$ 
2 while  $p_1! = NIL$  and  $p_2! = NIL$  do
3   if  $docID(p_1) = docID(p_2)$  then
4      $ADD(answer, docID(p_1));$ 
5      $p_1 = next(p_1);$ 
6      $p_2 = next(p_2);$ 
7   else
8     if  $docID(p_1) < docID(p_2)$  then
9        $p_1 = next(p_1)$ 
10    else
11       $p_2 = next(p_2)$ 
12    end
13  end
14 end
15 return  $answer$ 

```

3 Experimental Results

In Figure 1 we may see the interpolated precision, whereas in Table 1 we show the mean average precision obtained by the teams at the competition. The clustering-based approach obtained a low performance, and, therefore, we will discard the use of this technique in future experiments. With respect to the other approach, we did not obtained an official evaluation due to a format problem with our output file. However, a preliminar evaluation (using our own evaluation tools) show a very good performance. In summary, we consider that the use of techniques of information retrieval obtains the best performance in this kind of task, when the correct features are extracted from wsdl files.

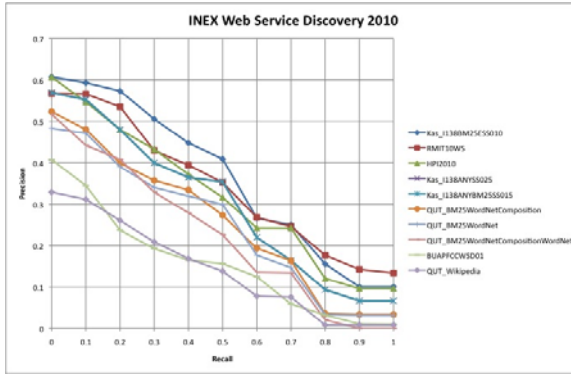


Fig. 1. Interpolated precision at standard recall levels

Table 1. Results at the INEX Web Service Discovery Track

Num	map	Institute	Run
1	0.3469	Kasetsart University	Kas_I138BM25ESS010
2	0.3239	RMIT University	RMIT10WS
3	0.2946	Hasso-Plattner-Institut	HPI2010
4	0.2798	Kasetsart University	Kas_I138ANYSS025
5	0.2798	Kasetsart University	Kas_I138ANYBM25SS015
6	0.2348	Queensland Univ. of Technology	QUT_BM25WordNetComposition
7	0.2233	Queensland Univ. of Technology	QUT_BM25WordNet
8	0.2042	Queensland Univ. of Technology	QUT_BM25WordNetCompWordNet
9	0.1451	B. Univ. Automa de Puebla	BUAPFCCWSD01
10	0.1268	Queensland Univ. of Technology	QUT_Wikipedia
11	0.1095	Queensland Univ. of Technology	QUT_WikipediaComposition
12	0.0937	Queensland Univ. of Technology	QUT_WikipediaCompositionWordNet

4 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. In general we may see that the second approach behaves better than the one based on clustering.

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), 136–158 (2004)
2. Guodong, Z.: A chunking strategy towards unknown word detection in chinese word segmentation. In: Dale, R., Wong, K.-F., Su, J., Kwong, O.Y. (eds.) *IJCNLP 2005*. LNCS (LNAI), vol. 3651, pp. 530–541. Springer, Heidelberg (2005)
3. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, Cambridge (2009)

XML Retrieval More Efficient Using Double Scoring Scheme

Tanakorn Wichaiwong and Chuleerat Jaruskulchai

Department of Computer Science,
Faculty of Science, Kasetsart University,
Bangkok Thailand
{g5184041, fscichj}@ku.ac.th

Abstract. This is the first year of Kasetsart University's participation in INEX. We participated in three tracks: the Ad Hoc, the Data Centric, and the Web Service Discovery tracks. In the Ad hoc and Data Centric tracks, the BM25F function has performed well in the past evaluations. However, there is an issue that needs more attention; what is the weight for each field? Previously, the weights are given manually to the fields. In general, many experts feel that uncontrolled selected fields. We proposed an unsupervised implementation of the BM25F scoring function, which we call the Double Scoring function. This scoring function assigns the weight for each field by an extended indexing scheme that handles the tuned weight parameter for each selected field. In the Web Service Discovery track, the standard tokens cannot be utilised directly by IR models, and need to be converted to natural language before indexing. We applied the Capitalization function to solve the tokenisation issue.

Keywords: XML Retrieval, Information Retrieval, Indexing Units, Ranking Schemes.

1 Introduction

The widespread use of Extensible Markup Language (XML) [1] documents in digital libraries has 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 the documents. Therefore, users who utilise an XML-IR system could potentially receive highly relevant and precise material.

In the Initiative for the Evaluation of XML Retrieval (INEX) [2] reports, the BM25F function has performed well in past evaluations. However, there is an issue that needs more attention: what is the weight for each field? Previously, the weights were obtained manually, tuning to the fields that affect the cost, the time pre-processing complexity and the heterogeneity of collections. Furthermore, these steps require training data and an evaluation set for parameter tuning, and difficulty often arises when evaluating new collections. Therefore, we have developed automatic methods to assign the weight for each field with an extended indices scheme.

This paper is organised as follows: Section 2 reviews related works, and Section 3 explains the implementation of our system. Section 4 presents the experiment. Conclusions and recommendations for further work are provided in Section 5.

2 Related Works

Prior work found in [3, 4, 5] presents the BM25F as the fielded version of the BM25 scoring function. Compared to the classical BM25, improvements have been discovered of up to 65% as measured by nxCG on the INEX-IEEE collection, with a different task where overlap is allowed. Authors of [7] applied the BM25F to INEX-Wikipedia’s collection; on each element, they constructed two fields, one for “title” and another for “body”. The title field consists of a concatenation of an article title and any section titles. The body field contains another context. In the INEX-2009 reports [8], the results showed the best ranking, with an iP [0.01] of 0.6333.

2.1 BM25F Function Overview

Robertson et al. [3, 4, 5, 6] presented the BM25F as the fielded version of the BM25 scoring function, which is considered to be composed of several document fields with possibly different degrees of importance, called selected fields. Using the BM25F scheme, an element’s score is computed as follows:

$$BM25F(e) = \sum_{t \in q \cap e} \frac{X_{e,t}}{K + X_{e,t}} * W_t \tag{1}$$

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 normalised term frequency.

K is a common tuning parameter for BM25.

W_t is the inverse document frequency weight of a term t .

The weighted normalised term frequency is obtained by first performing length normalisation, on the term frequency $W_{e,f,t}$ of a term t of a field f in an element e ,

$$W_{e,f,t} = \frac{X_{e,f,t}}{1 + B_f (\frac{l_{e,f}}{l_f} - 1)} \tag{2}$$

Note that:

B_f is a parameter to tune.

$l_{e,f}$ is a length of a field f in an element e .

l_f is the average length of elements in the whole collection multiplied the normalised term frequency $W_{e,f,t}$ by a field weight W_f ,

$$X_{e,t} = \sum_f W_f * W_{e,f,t} \tag{3}$$

3 XML Retrieval Model

3.1 Inverted File Definition

The Zettair search engine has good performance in [9], but this engine only supports the document level. We have to convert the element level to the document level by the absolute XPath [10], with context replacement to the <DOCNO> tag in the Zettair TREC format; our system is then able to use the best features of Zettair [11]. When we replace all tags to <DOCNO> by the absolute XPath, then leaf-only indexing is closest to traditional IR because each XML node is a bag of words itself and can be scored as an ordinary plain text document. Then we calculate the leaf element score for its context using BM25. For instance, take a document named x1.

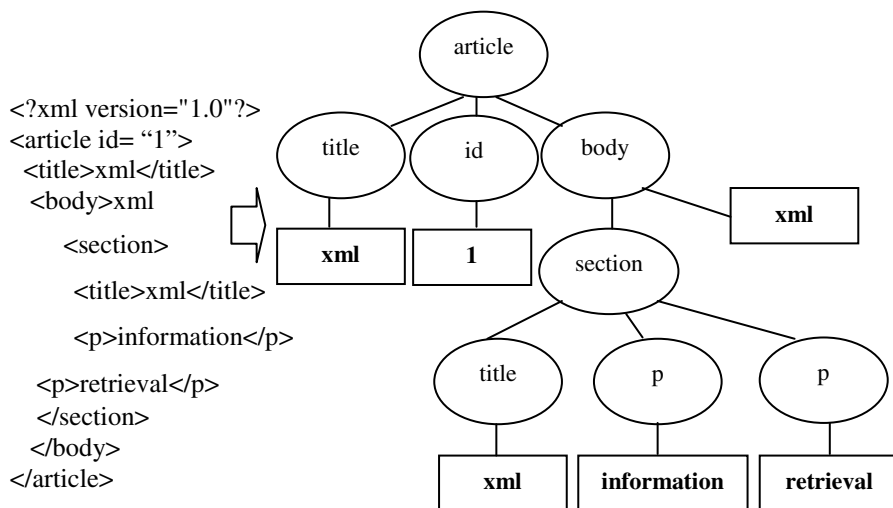


Fig 1. Example of an XML Element Tree

Fig. 1 depicts an example, the XML element tree of x1. We can build an index using the absolute XPath expression to identify the leaf node that has the text contained within the document, relative to the document and its parents, as shown in Table 1.

Table 1. Example of an Inverted File

<i>Term</i>	<i>Inverted List</i>
xml	x1/article[1]/title[1], x1/article[1]/body[1], x1/article[1]/body[1]/section[1]/title[1]
1	x1/article[1]/@id[1]
Information	x1/article[1]/body[1]/section[1]/p[1]
retrieval	x1/article[1]/body[1]/section[1]/p[2]

Finally, the term position identifies the ordinal position of the term within the XPath context. The next step is data preprocessing for the Zettair search engine application, and then we convert XPath into TREC format as follows:

```
<DOC><DOCNO>x1/article[1]/title[1]</DOCNO>xml</DOC>
<DOC><DOCNO>x1/article[1]/@id[1]</DOCNO>1</DOC>
<DOC><DOCNO>x1/article[1]/body[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]/p[1]</DOCNO>information</DOC>
<DOC><DOCNO>x1/article[1]/body[1]/section[1]/p[2]</DOCNO>retrieval</DOC>
```

3.2 The Leaf Node Scoring Function

Leaf-only indexing is closest to traditional information retrieval because each XML node is a bag of words and can be scored as an ordinary text document, after that we calculate the leaf element score using BM25 as follows:

$$LeafScore(e, Q) = \sum_{t \in Q} W_t * \frac{(k_1 + 1) * tf_e}{k_1 * ((1 - b) + b * \frac{len(e)}{avel}) + tf_e} \quad (4)$$

$$W_t = \log \left[\frac{N - e_t + 0.5}{e_t + 0.5} \right]$$

Note that:

$LeafScore(e, Q)$ measures the relevance of element e in the leaf-node index 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 the whole collection.

N is the total number of an element in the collection.

e_t is the total element of a term t occurrence.

k_1 and b are used to balance the weight of term frequency and element length.

3.3 Double Scoring Function

In previous reports [5], the authors raise the issue of the field weight W_f of the BM25F function (3). Because all of the weights need to be tuned for each selected field, this issue contributes to the document's weight in BM25F. In this report, the authors show that the tuning values for W_f are all integers, and they tuned W_f {at1, abs st} from {1, 1, 1} to {x, x, x}, using increments of 1. The result shows that the values of {2356, 4, 22} for W_f get the highest average precision score. In the same way, the report [7] shows the tuning values for W_f , and the authors tuned W_f {title, body} from {1, 1} to {x, x} using increments of 0.1. The result shows that the values of {4.0, 1.2} for W_f get the highest result on ip[0.01].

Our assumption of using the content at the document level has unfairness for all elements. For instance, the content in the "article[1]/body[1]/section[1]" element that

has not reflected to the “article[1]/title[1]” element with respect to the absolute XPath, and it has to reflect to their descendants are follows:

```

article[1]/body[1]/section[1]/title[1]
article[1]/body[1]/section[1]/p[1]
article[1]/body[1]/section[1]/p[2]

```

We have developed an automatic method to assign the weight for each field by an extended indices scheme, namely the *Double Scoring function*. This function is based on extending new indices to store all of the selected fields. The selected indexing is closest to traditional information retrieval, where each XML node is a bag of words by itself and can be scored as an ordinary plain text document; then we calculate the Selected Weight (SW) indices using the BM25 scoring function. After this step, we can compute an element’s final score, as follows:

```

For each swList in SWRelList
    swListWeight ← Score(swList);
    For each leafList in LeafNodeRelList
        leafListWeight ← Score(leafList);
        If leafList.StartsWith(swList) Then
            leafListWeight ← leafListWeight
            * swListWeight;
        Else
            leafListWeight ← leafListWeigh;
        End If
    End For
End For

```

Fig. 2. Details on the Double Scoring Algorithm

3.4 Double Scoring on BM25 (BM25W)

From function (3), we can see that the linear combination of weighted field frequencies is used instead of the original term frequency in selected fields. We assume that this method could be applied to all of the elements that we propose for SW indices. Our basic view is that an element is to be treated like a document.

Suppose that we have n selected fields in a given collection C . Given a field weight W_f of each element n in the selected fields, this contributes to a given weight in the SW indices, which we call W_{nf} . Then these indices can be captured in the weight for each selected field in the ordinary text document. Next, we calculate the SW using the BM25 scoring functions. For each weight W_f of the function of BM25F (3), we applied W_{nf} for each relevance list of SW, then we captured the element score using:

$$W_{nf} = SW(e, Q)$$

$$SW(e, Q) = \sum_{t \in Q} W_t * \frac{(k_1 + 1) * tf_e}{k_1 * ((1 - b) + b * \frac{\text{len}(e)}{\text{avel}}) + tf_e} \quad (5)$$

Note that:

W_{nf} is the field weight for each selected field in the Selected Weight indices.
 $SW(e, Q)$ measures the relevance of element e in the Selected Weight indices to a query Q .

Given a query Q , we run the query in parallel on each index (Selected Weight and Leaf-Node indices in Figure 2) and then integrate the Double Scoring by using the weight from $SWRelList$ of SW indices applied to each $LeafNodeRelList$ result set from the Leaf-Node indices. The weighting for each element in each $LeafNodeRelList$ result set is a linear combination of $SWRelList$ when the prefix of the result set is the same as the $SWRelList$ path, as shown in Figure 2. Then the new score for each $LeafNodeRelList$ list can compute BM25W, as follows:

$$BM25W(e, Q) = \sum_{t \in Q} LeafScore(e, Q) * SW(e, Q) \quad (6)$$

Note that;

$BM25W(e, Q)$ measures the relevance of element e to a query Q .

For the initial step, we consider a simplified XML data model but disregard any kind of Meta mark-up, including comments, links in the form of XLink or ID/IDRef, and attributes. Referring to an example of an XML element tree, we classify tag by manual, and then we can build new indices, as follows.

SW Indices;

x1/article[1]/title[1]: “xml”
 x1/article[1]/body[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”

3.5 Score Sharing Function

In previous reports [12], 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 nodes and their parents in the document’s XML tree according to the following scheme:

$$Score(PNode) \leftarrow Score(PNode) + [BM25W(e, Q) * \beta^n] \quad (7)$$

Note that:

$PNode$ is a current parent node.

β is a tuning parameter.

If $\{0 - 1\}$, then the preference is given to the leaf node over the parents.

Otherwise, the 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 on 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 an Intel Pentium i5 4 * 2.79 GHz with a memory of 6 GB, Microsoft Windows 7 Ultimate 64-bit Operating System and using Microsoft Visual C#.NET 2008 for the development system.

4.1 INEX Collections

The INEX document collections are the following: On the Ad hoc track, the collection is the Wikipedia XML Corpus of the English Wikipedia in early 2009 [13], which contain 2,666,190 articles and a total size of 50.7 GB.

1. On the Data Centric track, Information about one movie or person is published in one XML file [14]; thus, each generated XML file represents a single object, i.e., a movie or a person. In total, about 4,418,102 XML files were generated, including 1,594,513 movies, 1,872,492 actors, 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 that did not produce or direct or act in any movies, and the total size is 1.40 GB.

2. On the Web Service Discovery track, this track will use a collection of WSDL documents. These WSDL documents were directly taken 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 of the structures of each XML document with an XML parser and parses all of the selective nodes of each XML document. After that, our system uses the Leaf-Only indexing scheme in experiments.

4.2 Ad Hoc Track

In this section, we tuned parameters using INEX-2008 Ad hoc track evaluation scripts were distributed by the INEX organisers. Our tuning approach was such that the sum of all relevance scores was maximised 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 of $K1 = 1.80$ and $B = 0.40$ to evaluate the sensitivity of the element length in BM25 on both indices. For selected fields, we classify elements manually, including “article”, “name”, “caption”, “title”, “body”, “st”, “sec”, and “ss”.

We submitted three runs; I138BM25ESS010, I138BM25ESS015 and I138BM25ESS020. Our results showed that I138BM25ESS010 ranked 78th with MAiP at 0.1296, I138BM25ESS015 ranked 87th with MAiP at 0.1132, and our other run I138BM25ESS020 ranked 91st with MAiP at 0.1057. Table 2 and Figure 3 depict the measurement for Focused Retrieval.

Table 2. The Effectiveness of the Focused Task on INEX-Wikipedia

<i>RUN ID</i>	<i>B</i>	<i>iP[0.00]</i>	<i>iP[0.01]</i>	<i>iP[0.05]</i>	<i>iP[0.10]</i>	<i>MAiP</i>
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

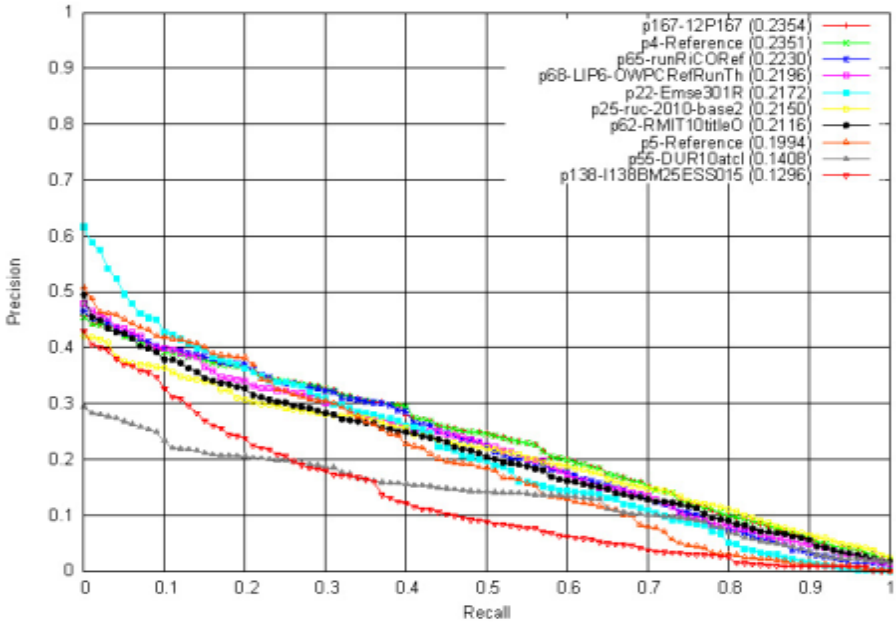


Fig. 3. INEX-2010 Ad hoc Result on Focused Retrieval

4.3 Data Centric Track

On the Data Centric track, we have used the values $K1 = 1.80$ and $B = 0.40$ to evaluate the sensitivity of the element length of 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 at this point of β parameters, after we analysed this collection. We found that all of the content had quite a few data, and then we turned to the document level as the better answer for the user.

Table 3. The Effectiveness of Data Centric

<i>RUN ID</i>	β	<i>iP[0.00]</i>	<i>iP[0.01]</i>	<i>iP[0.05]</i>	<i>iP[0.10]</i>	<i>MAiP</i>
I138BM25ESS060	0.60	0.5821	0.5113	0.3244	0.2719	0.1211

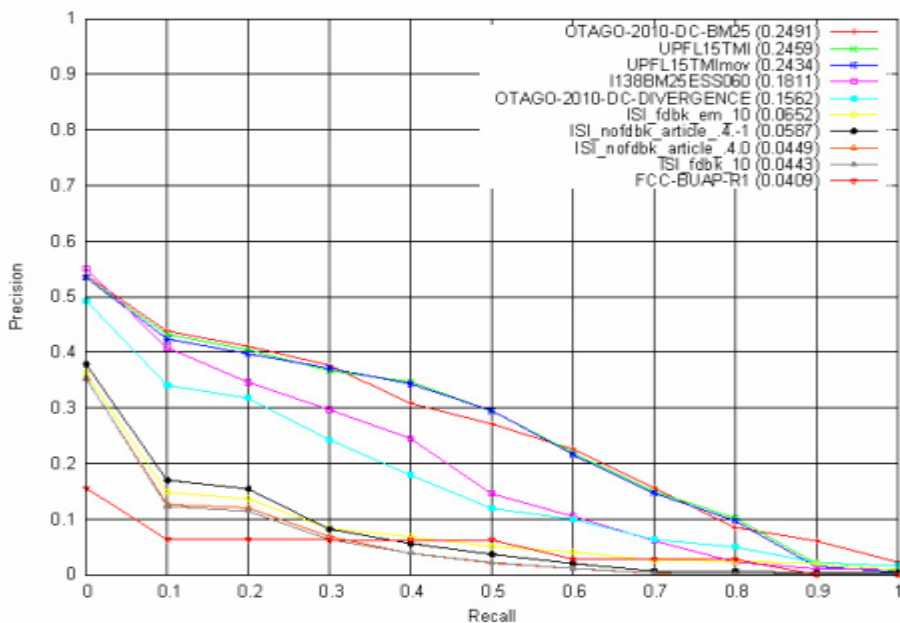


Fig. 4. INEX-2010 Data Centric Result on Focused Retrieval

We submitted only one run; our result showed that I138BM25ESS060 ranked 4th with MAgP at 0.1811 and ranked 12th with MAiP at 0.1211. Table 3 and Fig. 4 depict the measurement of Focused Retrieval on the Data Centric track.

4.4 Web Service Discovery Track

On the Web Service Discovery track, standard tokens cannot be utilised directly due to various reasons such as sublanguage patterns and programming conventions. Therefore, these tokens need to be converted to natural languages before being indexed using IR models.

The Capitalization Styles in [15] utilised three types of capitalisation styles: first, UpperCamelCase (Pascal), whereby the first letter is the identifier and the first letters of each subsequent concatenated word are capitalised, such as BackColor DataSet; second, lowerCamelCase (Camel), which means that the first letter of an identifier is lowercase and the first letter of each subsequent concatenated word is capitalised, such as backColor, dataSet; and last, UpperCase (Upper), in which all of the letters in the identifier are capitalised, such as BACKCOLOR, DATASET. Almost all of the web service methods have already utilised the form of CamelCase, such as naming conventions in several programming languages. For instance, take a part of a document name 0001.wsdl as follows.

In Figure 5, for example, applying the Capitalisation function [16] for the “msdGetXsdBase64BinData” and then applying the above function gives the result “msd”, “Get”, “Xsd”, “Base64”, “Bin” and “Data”. We applied this function to solve the tokenisation issue for both elements and attributes, and then we used the More

```
</operation>  
<operation name="msdGetXsdBase64BinData">  
  <SOAP:operation soapAction=""/>  
  <input>  
    <SOAP:body use="encoded" namespace="urn:msd_soap_service" encodingStyle="http://schemas.xml  
  </input>  
  <output>  
    <SOAP:body use="encoded" namespace="urn:msd_soap_service" encodingStyle="http://schemas.xml  
  </output>  
</operation>
```

Fig. 5. The Example of WDSL

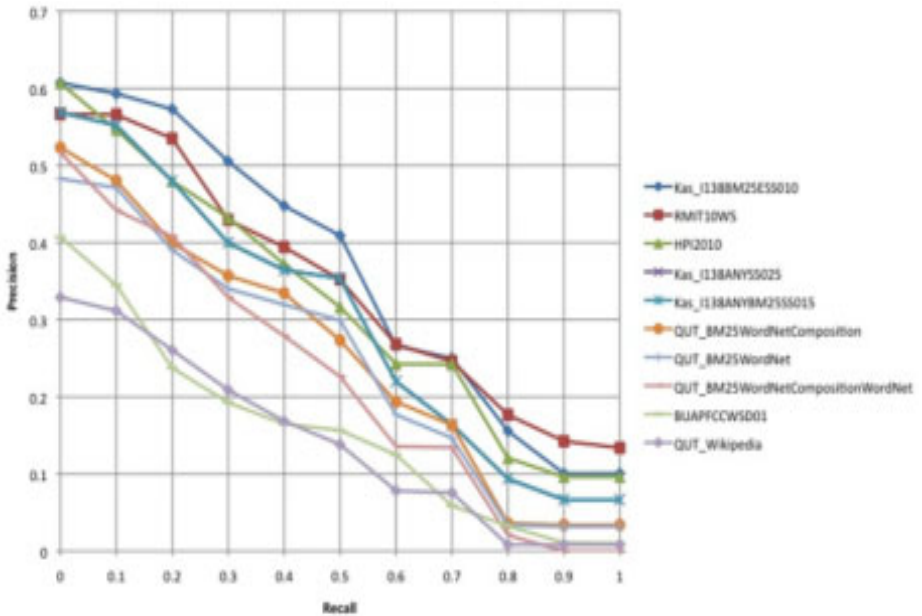


Fig. 6. INEX-2010 Web Service Discovery Result

Efficient XML Information Retrieval (MEXIR) [17], an XML information retrieval system that uses MySQL [18] and Sphinx [19].

We submitted three runs: Kas_I138BM25ESS010 using the Sphinx mode BM25 and Kas_I138ANYBM25SS015 and Kas_I138ANYSS025 using Sphinx mode ANY, as shown in Fig. 5. Our results showed that Kas_I138BM25ESS010 ranked 1st with MAgP at 0.3469, Kas_I138ANYBM25SS015 ranked 4th with MAgP at 0.2798 and our other run I138ANYSS025 ranked 5th with MAgP at 0.2798, as shown in Fig. 6.

5 Conclusions

Due to the ever-increasing information available electronically, the amount of information is growing rapidly in size and requires efficient and effective indexing and retrieval methods. Structured retrieval breaks away from the traditional retrieval unit and aims to implement focused retrieval. This focused retrieval strategy aims to return document components, i.e., XML elements, instead of whole documents in response to a user query.

In this paper, we report experimental results of our approach using the Double Scoring function base on the BM25 model for the retrieval of large-scale XML collections. This strategy can process by using only common parameters on BM25 by extending new indices to store all of the selected fields in order to reduce the amount of parameter-tuned weights for each selected field. In addition, we applied the Capitalization function to solve the tokenisation issue on the web service discovery track.

In our future work, we plan to: 1) extend Anchor text to study the sensitivity of the BM25W scoring function, 2) study the sensitivity of the evaluation to the k_1 and b parameters for each index, and 3) show how to make inferences regarding structural aspects based on CAS queries.

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. Craswell, N., Zaragoza, H., Robertson, S.: Microsoft Cambridge at TREC 14: Enterprise track. In: Proceedings of the TREC 14 (2005)
4. Robertson, S., Zaragoza, H., Taylor, M.: Simple BM25 extension to multiple weighted fields. In: Proceedings of CIKM, pp. 42–49 (2004)
5. Lu, W., Robertson, S., MacFarlane, A.: Field-Weighted XML Retrieval Based on BM25. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 161–171. Springer, Heidelberg (2006)
6. Itakura, K.Y., Clarke, C.L.A.: A Framework for BM25F-based XML Retrieval. In: SIGIR 2010, pp. 843–844 (2010)
7. Itakura, K.Y., Clarke, C.L.A.: University of Waterloo at INEX 2009: Ad Hoc, Book, Entity Ranking, and Link-the-Wiki Tracks. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 331–341. Springer, Heidelberg (2010)

8. Geva, S., Kamps, J., Lethonen, M., Schenkel, R., Thom, J.A., Trotman, A.: Overview of the INEX 2009 Ad Hoc Track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 4–25. Springer, Heidelberg (2010)
9. Middleton, C., Baeza-Yates, R.: A Comparison of Open Source Search Engines (2009), <http://wrg.upf.edu/WRG/dctos/Middleton-Baeza.pdf>
10. XML Path Language (XPath) Version 1.0., <http://www.w3.org/TR/xpath>
11. Zettair. The Zettair search engine (2009), <http://www.seg.rmit.edu.au/zettair/>
12. Wichaiwong, T., Jaruskulchai, C.: A Simple Approach to Optimize XML Retrieval. In: The 6th International Conference on Next Generation Web Services Practices, Goa, India, November 23-25 (2010)
13. Schenkel, R., Suchanek, F., Kasneci, G.: YAWN: A semantically annotated Wikipedia XML corpus. In: 12. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web, pp. 277–291 (2007)
14. Information courtesy of The Internet Movie Database, <http://www.imdb.com>
15. 3Suns, CamelCase, <http://everything2.com/e2node/CamelCase>
16. Wichaiwong, T., Jaruskulchai, C.: A Simple Approach to Optimize Text Compression's Performance. In: The 4th International Conference on Next Generation Web Services Practices, Seoul, Korea, October 20-22 (2008)
17. Wichaiwong, T., Jaruskulchai, C.: MEXIR: An Implementation of High-Speed and High-Precision XML Information Retrieval. In: The 3rd International Conference on Machine Learning and Computing, Singapore, February 26-28 (2011)
18. MySQL Full-Text Search Functions, <http://dev.mysql.com/>
19. Sphinx Open Source Search Server, <http://www.sphinxsearch.com/>

Overview of the INEX 2010 XML Mining Track: Clustering and Classification of XML Documents

Christopher M. De Vries¹, Richi Nayak¹, Sangeetha Kutty¹, Shlomo Geva¹,
and Andrea Tagarelli²

¹ Faculty of Science and Technology,
Queensland University of Technology, Brisbane, Australia

² University of Calabria, Italy
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.

Keywords: 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 classification tasks.

2 Corpus

Working with XML documents is a particularly challenging task for ML and IR. XML documents are defined by their logical structure and content. The

current Wikipedia collection contains structure as (1) document structure such as sections, titles and tables, (2) semantic structure as entities mined by YAWN, and (3) 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 judgements 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.

Table 1. Relevant Documents Missing from the XML Mining Subset

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%)

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 for 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”¹. 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 by 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.

For INEX 2010 the category graph from the 22nd of June 2010 Wikipedia dump was used. The graph consists 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 were not used. Table 2 lists the categories that were extracted.

In Figure 2, 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 where the vertices consist of pages P and categories C . Where $P \subset V$, $C \subset V$, $M \subset V$ and $M \subset C$. Moreover, t is a parameter indicating the broadest t levels to consider as categories; if t is 1 then only the main topic

¹ http://en.wikipedia.org/wiki/Category:Main_topic_classifications

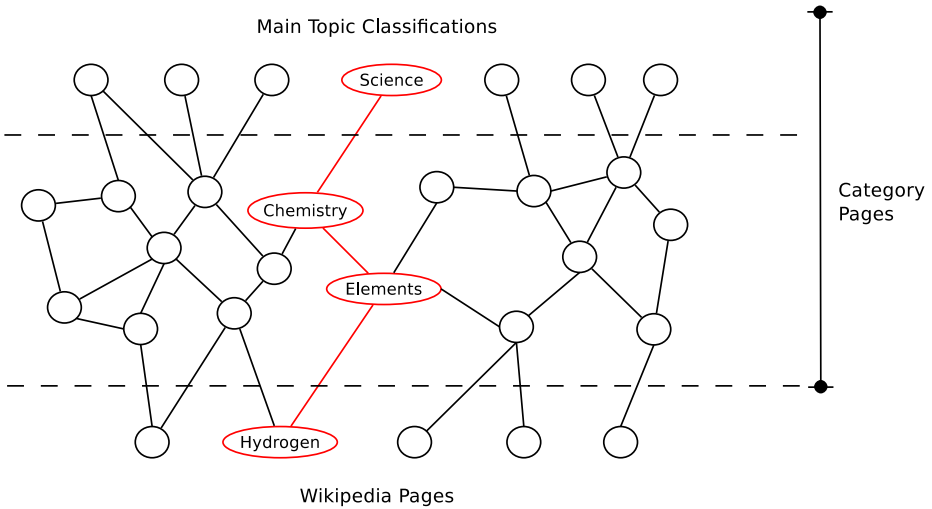


Fig. 1. Complicated and Noisy Wikipedia Category Graph

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 hypothetical path for the Wikipedia document Hydrogen.

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 topic classifications” appears to have processed the category graph as well. Using this post-processed graph could also improve the categories.

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

Table 2. XML Mining Categories

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

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 clusters. The categories extracted contained 36 categories due to only using categories with greater than 3000 documents. This choice was arbitrary and the decision for cluster sizes was made based on the number of documents in the collection before the categories were extracted. As there are not really 36 “true” categories, a direct comparison of 36 clusters with 36 categories is not necessary. The number of categories in a document collection is extremely subjective. Measuring how the categories behave over multiple cluster sizes indicates the quality of clusters and the trend can be visualized.

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 [5] 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.

Purity. The standard criterion of purity is used to determine the quality of clusters by measuring the extent to which each cluster contains documents primarily from one category. The simplicity and the popularity of this measure means that it has been used as the only evaluation measure for the clustering task in the INEX 2006 and INEX 2009. In general, the larger the value of purity, the better the clustering solution.

Let $\omega = \{w_1, w_2, \dots, w_K\}$, denote the set of clusters for the dataset D and $\xi = \{c_1, c_2, \dots, c_J\}$ represent the set of categories. The purity of a cluster w_k is defined as:

$$P(w_k) = \frac{\max_j |w_k \cap c_j|}{|w_k|} \quad (1)$$

where w_k is the set of documents in cluster w_k and c_j is the set of documents that occurs in category c_j . The numerator indicates the number of documents in cluster k that occurs most in category j and the denominator is the number of documents in the cluster w_k .

The purity of the clustering solution ω can be calculated based on micro-purity and macro-purity. Micro-purity of the clustering solution ω is obtained as a weighted sum of individual cluster purity. Macro-purity is the unweighted arithmetic mean based on the total number of categories [5].

$$\text{Micro-Purity}(\omega, \xi) = \frac{\sum_{k=0}^K P(w_k) * |w_k|}{\sum_{k=0}^K |w_k \cap c_j|} \quad (2)$$

$$\text{Macro-Purity}(\omega, \xi) = \frac{\sum_{k=0}^K P(w_k)}{J} \quad (3)$$

Entropy. It is used to measure the distribution of the documents on various categories. Given a particular cluster ω_k of size n_k , the entropy of this cluster is defined to be:

$$E(\omega_k) = -\frac{1}{\log J} \sum_{i=1}^J \frac{n_k^i}{n_k} \log \frac{n_k^i}{n_k} \quad (4)$$

where J is the number of categories in the dataset, and n_k^j is the number of documents of the j th category that were assigned to the k th cluster [6]. The

clustering solution can then be measured by the sum of the individual cluster entropies weighted according to the clustering size as defined below:

$$Entropy = \sum_{k=1}^K \frac{n_k}{K} E(\omega_k) \tag{5}$$

It is scaled from 0 to 1. A perfect clustering solution will have an entropy value of 0.

F1-measure. Another standard measure that is used to evaluate the clustering solution is the F1-measure. It helps to calculate not only the number of documents that are correctly classified together in a cluster but also the number of documents that are misclassified from the cluster.

In order to calculate the F1-measure, three types of decisions are used. Among them there are two types of correct decisions: True Positives (TP) and True Negatives (TN). A TP decision assigns two similar documents to the same cluster; a TN decision assigns two dissimilar documents to different clusters. On the other hand, a False Positive (FP) is an error decision that assigns two dissimilar documents to the same cluster [7]. Though there is another error decision, FN, that assigns two similar documents to different clusters, it is not used in calculating F1-measure.

Using the TP, TN and FP decisions, the precision and the recall for the micro-F1 are defined as:

$$precision_{micro-F1} = \frac{\sum_{j=1}^J TP_j}{\sum_{j=1}^J TP_j + FP_j} \tag{6}$$

$$recall_{micro-F1} = \frac{\sum_{j=1}^J TP_j}{\sum_{j=1}^J TP_j + TN_j} \tag{7}$$

The precision and the recall for the macro-F1 are defined as

$$precision_{macro-F1} = \frac{\sum_{j=1}^J \frac{TP_j}{TP_j + FP_j}}{J} \tag{8}$$

$$recall_{macro-F1} = \frac{\sum_{j=1}^J \frac{TP_j}{TP_j + TN_j}}{J} \tag{9}$$

where TP_j is the number of documents in category c_j that exists in cluster w_k , FP_j is the number of documents that is not in category c_j but that exists in cluster w_k and TN_j is the number of documents that is in category c_j but does not exist in cluster w_k .

F1 can now be defined as:

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{10}$$

$$Micro-F1 = \frac{2 \times precision_{micro-F1} \times recall_{micro-F1}}{precision_{micro-F1} + recall_{micro-F1}} \tag{11}$$

$$Macro-F1 = \frac{2 \times precision_{macro-F1} \times recall_{macro-F1}}{precision_{macro-F1} + recall_{macro-F1}} \quad (12)$$

Normalized Mutual Information (NMI). Another evaluation measure is the Normalized Mutual Information (NMI) which helps to identify the trade-off between the quality of the clusters against the number of clusters [7].

NMI [7] is defined as,

$$NMI(\omega, \xi) = \frac{I(\omega; \xi)}{[H(\omega) + H(C)]/2} \quad (13)$$

$$I(\omega; \xi) = \frac{\sum_k \sum_j P(w_k \cap c_j) \log \frac{P(w_k \cap c_j)}{P(w_k)P(c_j)}}{\sum_k \sum_j \frac{|w_k \cap c_j|}{N} \log \frac{N|w_k \cap c_j|}{|w_k||c_j|}} \quad (14)$$

where $P(w_k)$, $P(c_j)$ and $P(w_k \cap c_j)$ indicate the probabilities of a document in cluster w_k , category c_j and in both w_k and c_j .

$H(\omega)$ is the measure of uncertainty given by,

$$H(\omega) = \frac{-\sum_k (P(w_k) \log P(w_k))}{-\sum_k \frac{|w_k|}{N} \log \frac{|w_k|}{N}} \quad (15)$$

Collection Selection Evaluation Using NCCG Measure

This evaluation measure was used in evaluating the INEX 2009 dataset [4] and is based on Van Rijsbergen's clustering hypothesis. Van Rijsbergen and his co-workers [8] conducted intensive study on the use of the clustering hypothesis test on information retrieval, which states that documents which are similar to each other may be expected to be relevant to the same requests; dissimilar documents, conversely, are unlikely to be relevant to the same requests. If the hypothesis holds true, then relevant documents will appear in a small number of clusters and the document clustering solution can be evaluated by measuring the spread of relevant documents for the given set of queries.

To test this hypothesis on a real-life dataset, the INEX 2009 dataset, the clustering task was evaluated by determining the quality of clusters relative to the optimal collection selection [4]. Collection selection involves splitting a collection into subsets and recommending which subset needs to be searched for a given query. This allows a search engine to search fewer documents, resulting in improved runtime performance over searching the entire collection.

The evaluation of collection selection was conducted using the manual query assessments for a given set of queries from the INEX 2009 Ad Hoc track [4]. The manual query assessment is called the relevance judgment in Information Retrieval (IR) and has been used to evaluate ad hoc retrieval of documents. It involves defining a query based on the information need, a search engine returning results for the query and humans judging whether the results returned by the search engine are relevant to the information need.

Better clustering solutions in this context will tend to (on average) group together relevant results for (previously unseen) ad hoc queries. Real ad hoc

retrieval queries and their manual assessment results are utilised in this evaluation. This approach evaluates the clustering solutions relative to a very specific objective – clustering a large document collection in an optimal manner in order to satisfy queries while minimising the search space. The metric used for evaluating the collection selection is called the Normalized Cumulative Cluster gain (NCCG) [4].

The NCCG is used to calculate the score of the best possible collection selection according to a given clustering solution of n number of clusters. The score is better when the query result set contains more cohesive clusters. The Cumulative Gain of a Cluster (CCG) is calculated by counting the number of documents of the cluster that appear in the relevant set returned for a topic by manual assessors.

$$CCG(c, t) = \sum_{i=1}^n (Rel_i) \tag{16}$$

For a clustering solution for a given topic, a (sorted) vector CG is created representing each cluster by its CCG value. Clusters containing no relevant documents are represented by a value of zero. The cumulated gain for the vector CG is calculated, which is then normalized on the ideal gain vector. Each clustering solution c is scored for how well it has split the relevant set into clusters using CCG for the topic t .

$$SplitScore(t, c) = \sum^{|CG|} \frac{cumsum(CG)}{nr^2} \tag{17}$$

where nr = Number of relevant documents in the returned result set for the topic t .

A scenario with worst possible split is assumed to place each relevant document in a distinct cluster. Let CG1 be a vector that contains the cumulative gain of every cluster with each document.

$$MinSplitScore(t, c) = \sum^{|CG1|} \frac{cumsum(CG1)}{nr^2} \tag{18}$$

The normalized cluster cumulative gain (nCCG) for a given topic t and a clustering solution c is given by,

$$nCCG(t, c) = \frac{SplitScore(t, c) - MinSplitScore(t, c)}{1 - MinSplitScore(t, c)} \tag{19}$$

The mean and the standard deviation of the nCCG score over all the topics for a clustering solution are then calculated.

$$Mean(nCCG(c)) = \frac{\sum_{t=0}^n nCCG(t, c)}{Total\ Number\ of\ topics} \tag{20}$$

$$Std\ Dev\ (nCCG(c)) = \frac{\sum_{t=0}^n [nCCG(t, c) - Mean(nCCG(c))]^2}{Total\ Number\ of\ topics} \tag{21}$$

The NCCG value varies from 0 to 1. A larger value of NCCG for a given clustering solution is better, since it represents the fact that an increased number of relevant documents are clustered together in comparison to a smaller number of relevant documents. Further details of this metric can be found in [4].

Divergence from Random. Most measures of cluster quality can be tricked by changing the number of clusters or documents in the submission. 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. The NCCG measure orders clusters by the number of relevant documents they contain. A large cluster containing most documents will almost always be ranked first. Therefore, almost 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 adjusted 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 to be generated. This is done by shuffling the document IDs uniformly randomly and splitting them into clusters the same size as the solution being measured. The score for the uniform random solution is 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 7.

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 4 to 7 graph the best performing submissions from each participant for Purity, Negentropy, NMI and NCCG. The divergence from random for each metric is also graphed. Figure 3 contains the legend for all these graphs.

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.

Participants

- QUT RI 1024 bit
- △ Peking 4
- BUAP MS
- + Random

Fig. 3. Legend

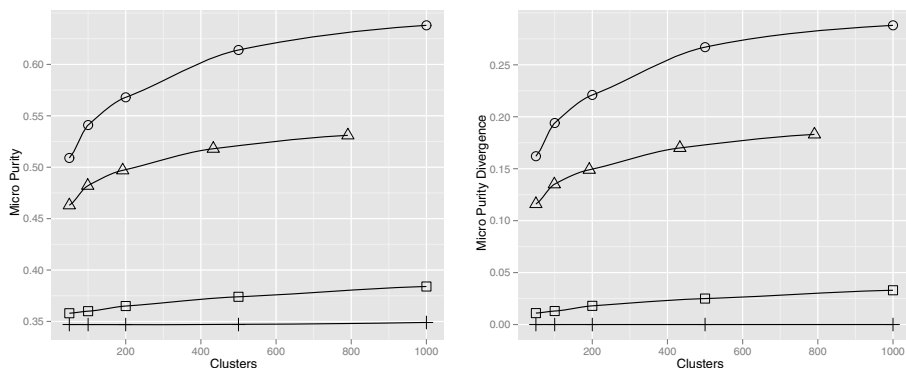
The group from Peking University [9] made a submission based on the structured link vector model (SLVM). It incorporated document structure, links and content. This year they focused on the preprocessing step for document structure and links. They modified two popular clustering algorithms, AHC clustering algorithm and K-means algorithm, to work with this model.

The group from BUAP [10] 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, they 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.

**Fig. 4.** Micro Purity

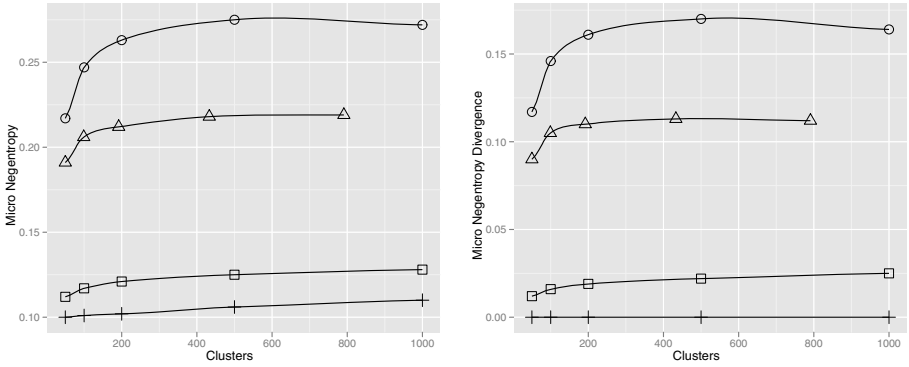


Fig. 5. Micro Negentropy

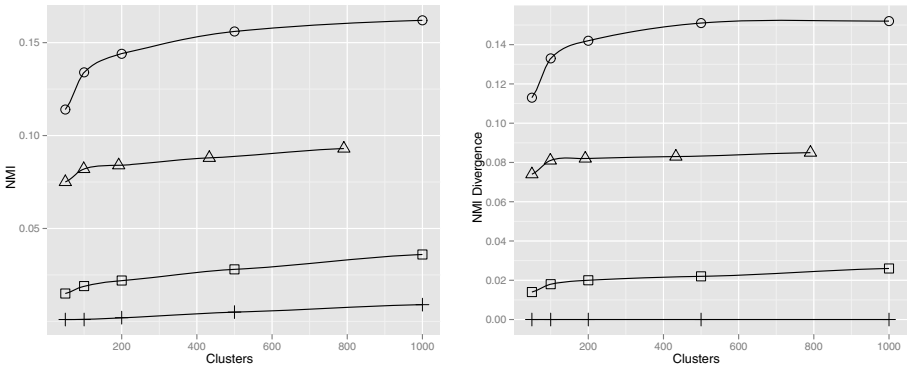


Fig. 6. NMI

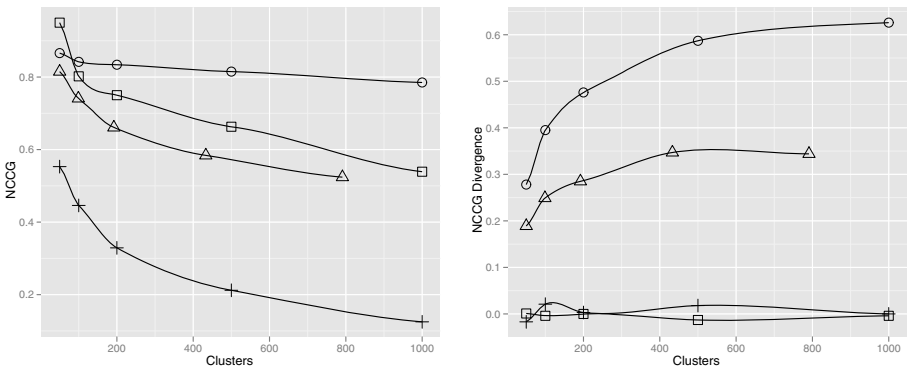


Fig. 7. NCCG

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.

Table 3. Type I and II Classification Errors

	In Category	Not in Category
Predicted in Category	True Positive (<i>tp</i>)	False Positive (<i>fp</i>)
Predicted not in Category	False Negative (<i>fn</i>)	True Negative (<i>tn</i>)

The F1, Precision and Recall scores are calculated as described in Equations 22 to 24. F1 is the harmonic mean of precision and recall.

$$F1 = \frac{2 \times tp}{2 \times tp + fn + fp} \tag{22}$$

$$Precision = \frac{tp}{tp + fp} \tag{23}$$

$$Recall = \frac{tp}{tp + fn} \tag{24}$$

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.

Table 4. Classification Results

Submission	F1		Precision		Recall	
	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

The group from Peking University [9] 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 participants to explore algorithmic, theoretical and practical issues regarding the classification of interdependent XML documents.

References

1. Denoyer, L., Gallinari, P., Vercoestre, A.-M.: Report on the XML mining track at INEX 2005 and INEX 2006. In: Fuhr, N., Lalmas, M., Trotman, A. (eds.) INEX 2006. LNCS, vol. 4518, pp. 432–443. Springer, Heidelberg (2007)
2. Denoyer, L., Gallinari, P.: Report on the XML mining track at INEX 2007 categorization and clustering of XML documents. In: ACM SIGIR Forum, vol. 42, pp. 22–28. ACM, New York (2007)
3. Denoyer, L., Gallinari, P.: Overview of the INEX 2008 XML mining track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 401–411. Springer, Heidelberg (2009)
4. Nayak, R., De Vries, C.M., Kutty, S., Geva, S., Denoyer, L., Gallinari, P.: Overview of the INEX 2009 XML mining track: Clustering and classification of XML documents. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 366–378. Springer, Heidelberg (2010)
5. De Vries, C.M., Geva, S.: Document clustering with K-tree. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 420–431. Springer, Heidelberg (2009)
6. Zhao, Y., Karypis, G.: Criterion functions for document clustering: experiments and analysis. Technical Report Technical Report 01-40, Department of Computer Science (November 2001)
7. Manning, C.D., Raghavan, P., Scütze, H.: Introduction to Information Retrieval. Cambridge University Press, NY (2008)
8. Rijsbergen, C.J.v.: Information Retrieval. Butterworths, London (1979)
9. Wang, S., Liang, F., Yang, J.: PKU at INEX 2010 XML mining track. In: Geva, S., et al. (eds.) INEX 2010. LNCS, vol. 6932, pp. 383–395. Springer, Heidelberg (2010)
10. Tovar, M., Cruz, A., Vázquez, B., Pinto, D., Vilariño, D.: An iterative clustering method for the XML-mining task of the INEX 2010. In: Geva, S., et al. (eds.) INEX 2010. LNCS, vol. 6932, pp. 377–382. Springer, Heidelberg (2010)

An Iterative Clustering Method for the XML-Mining Task of the INEX 2010*

Mireya Tovar^{1,4}, Adrián Cruz^{2,4}, Blanca Vázquez^{3,4},
David Pinto¹, Darnes Vilariño¹, and Azucena Montes⁴

¹ Benemérita Universidad Autónoma de Puebla, México

² Instituto Tecnológico de Cerro Azul, México

³ Instituto Tecnológico de Tuxtla Gutiérrez, México

⁴ Centro Nacional de Investigación y Desarrollo Tecnológico, México
{`mtovar`,`dpinto`,`darnes`}@`cs.buap.mx`, `abadrector@gmail.com`,
`blanca_tec@hotmail.com`, `amr@cenidet.edu.mx`

Abstract. In this paper we propose two iterative clustering methods for grouping Wikipedia documents of a given huge collection into clusters. The recursive method 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 scalable algorithms which may be used with huge collections that in other way (for instance, using the classic clustering methods) would be computationally expensive 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 for measuring 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 that 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.

* This work has been partially supported by the CONACYT project #106625, VIEP #PIAD-ING11-I, as well as by the PROMEP/103.5/09/4213 grant.

The first approach consider two main modules: the K-biX and the K-biN.

The K-biX 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 of the similarity matrix). The procedure of K-biX is given in Algorithm [1](#).

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 ϵ

Output: A set of clusters C_1, C_2, \dots

```

1  $D = \{d_1, d_2, \dots\};$ 
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      $tmp = d_i; d_i = d_j; d_j = tmp;$ 
7   if ( $d_i \notin rel$  and  $d_j \notin rel$ ) then
8      $fusion_{d_i} = \{d_i, d_j\};$ 
9      $rel_{d_i} = \{d_i\}; rel_{d_j} = \{d_i\};$ 
10  else if ( $d_i \in rel$ ) and ( $d_j \notin rel$ ) then
11     $fusion_{rel_{d_i}} = fusion_{rel_{d_i}} \cup \{d_j\};$ 
12     $rel_{d_j} = \{d_i\};$ 
13  else if  $d_j \in rel$  and ( $d_i \notin rel$ ) then
14     $fusion_{rel_{d_j}} = fusion_{rel_{d_j}} \cup \{d_i\};$ 
15     $rel_{d_i} = \{d_j\};$ 
16   $loop = loop + 1;$ 
17  $cluster = 1;$ 
18 foreach  $d_x \in fusion$  do
19    $first\_d\_representative(fusion_{d_x});$ 
20    $C_{cluster} = fusion_{d_x};$ 
21    $cluster = cluster + 1;$ 
22 foreach  $d_x \in |D|$  do
23   if  $d_x \notin rel$  then
24      $C_{cluster} = \{d_x\};$ 
25      $cluster = cluster + 1;$ 
26 return  $C_1, C_2, \dots$ 

```

K-biX is unable to find a fixed number of clusters; instead, it tries to discover the optimal number of clusters. Therefore, we have executed an additional clustering method in order to fix the number of clusters to exactly those required in the competition (50, 100, 200, 500 and 1000).

On the other hand, the K-biN clustering method considers the clustering with a fixed number of clusters. This number depends of criteria given in advance, and

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, \dots, C_n

```

1   $D = \{d_1, d_2, \dots\}$ ;
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           $tmp = d_i$ ;  $d_i = d_j$ ;  $d_j = tmp$ ;
9      if ( $d_i \notin rel$  and  $d_j \notin rel$ ) then
10          $fusion_{d_i} = \{d_i, d_j\}$ ;
11          $rel_{d_i} = \{d_i\}$ ;  $rel_{d_j} = \{d_i\}$ ;
12          $cn = cn + 2$ ;
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          $cn = cn + 1$ ;
17     else if  $d_j \in rel$  and ( $d_i \notin rel$ ) then
18          $fusion_{rel_{d_j}} = fusion_{rel_{d_j}} \cup \{d_i\}$ ;
19          $rel_{d_i} = \{d_j\}$ ;
20          $cn = cn + 1$ ;
21     if  $|D| - n \geq cn$  then
22          $f = 0$ ;
23          $cd = 0$ ;
24         foreach  $d_x \in fusion$  do  $f = f + 1$ ;
25         foreach  $d_x \in |D|$  do
26             if  $d_x \in rel$  then  $cd = cd + 1$ ;
27          $gs = f + cd$ ;
28      $loop = loop + 1$ ;
29  $cluster = 1$ ;
30 foreach  $d_x \in fusion$  do
31      $first\_d\_representative(fusion_{d_x})$ ;
32      $C_{cluster} = fusion_{d_x}$ ;
33      $cluster = cluster + 1$ ;
34 foreach  $d_x \in D$  do
35     if  $d_x \notin rel$  then
36          $C_{cluster} = \{d_x\}$ ;
37          $cluster = cluster + 1$ ;
38 return  $C_1, C_2, \dots, C_n$ 

```

Algorithm 3: Algorithm used for clustering the INEX 2010 with K-Means

Input: A corpus D , n number of clusters (50, 100, 200, 500 or 1000)

Output: A set of clusters C_1, C_2, \dots, 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 $D_i = \{d_{i,1}, d_{i,2}, d_{i,3}, \dots, d_{i,\frac{|D|}{m}}\}$ **do**
- 4 Calculate the similarity matrix M_i of D_i by using the cosine measure;
- 5 Apply the K-Means clustering method to M_i in order to obtain k clusters;
- 6 $\{C_{i,1}, C_{i,2}, \dots, C_{i,k}\}$;
- 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 $d_{i,j}$;
- 12 Calculate the similarity matrix M'_i of D'_i by using the cosine measure;
- 13 Apply the K-Means clustering method to M'_i in order to obtain n clusters
 $\{C'_{i,1}, C'_{i,2}, \dots, C'_{i,n}\}$;
- 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
 $j \ll j'$;
- 15 $Loop = Loop + 1$;
- 16 **end**
- 17 **return** C_1, C_2, \dots, C_n

the similarity degree among the documents processed. The Algorithm 2 presents the K-bin method.

2.1 The K-Means Based Clustering Approach

The second approach has used 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. The rationale of the K-Means clustering method is shown in (3). The main advantage of this algorithm is 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 is depicted in Algorithm 3.

The clustering methods aforementioned assume that a matrix that represents the similarity degree among all the documents of the collection is given in advance. For this purpose, we construct this matrix by using the cosine similarity measure over a vectorial representation of each document 4.

The obtained results are presented and discussed in the following section.

3 Experimental Results

The experiments were carried out on the clustering task of the INEX 2010, which evaluated unsupervised machine learning solutions against the ground

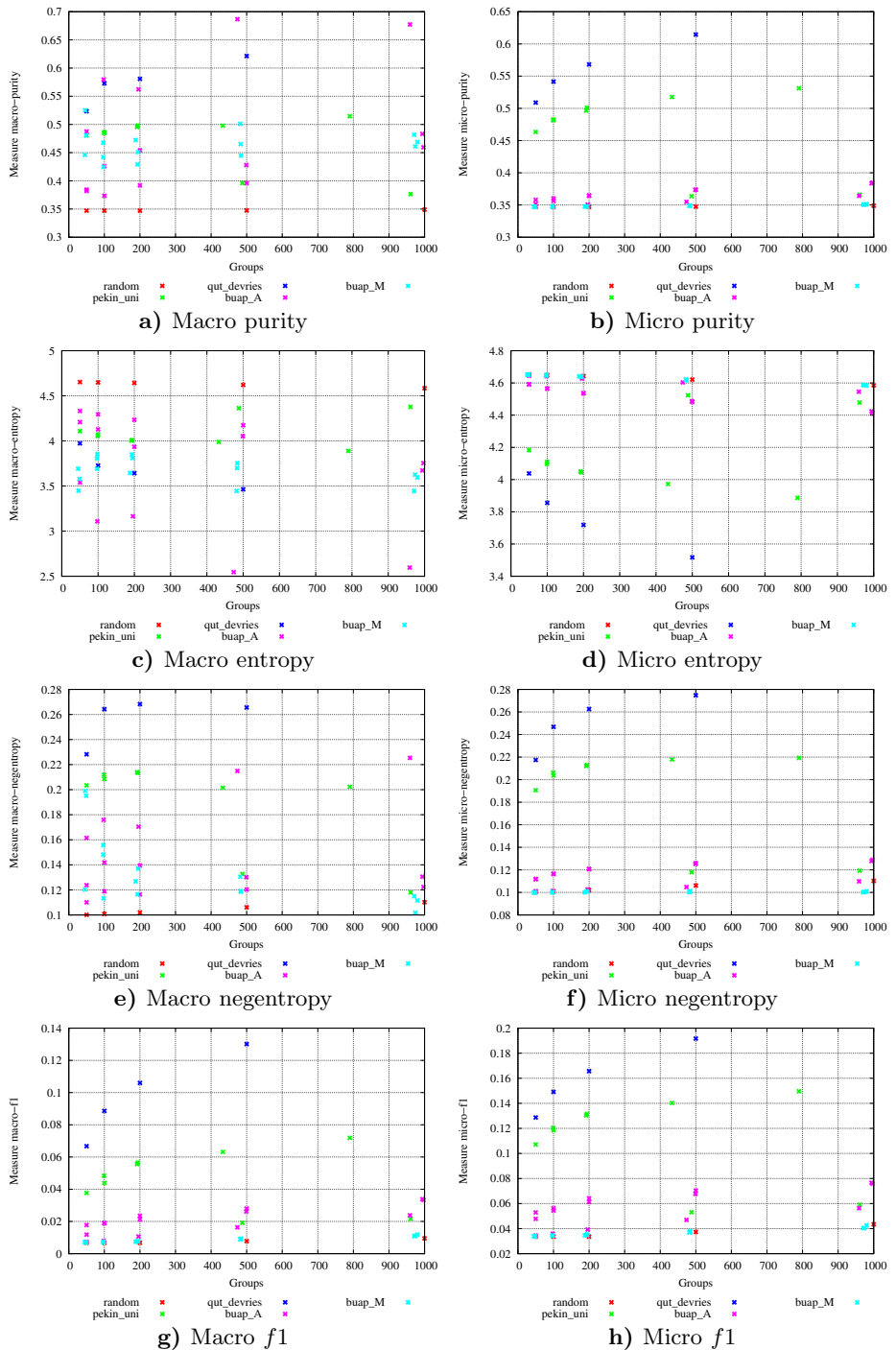


Fig. 1. Evaluations of the runs with Purity, Entropy, negentropy and f_1 (macro and micro) at the INEX 2010 clustering task

truth categories by using standard evaluation criteria such as Purity, Entropy, Negentropy and F-score ($f1$).

The evaluation is performed by using 146,225 documents that have been pre-processed in order to provide various representations of these documents such as, vector space representation of terms, frequent bi-grams, XML tags, trees, links and named entities. In this paper, we have used unigrams and frequent bigrams (original terms and stemmed terms). The obtained results are shown in Figure 1.

In general, it can be noticed that the presented approaches performs slightly better than the random assignment. Our thought is that we have not sufficiently 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 these hypotheses.

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 made up of a random assignment. We would like to analyze this behavior when the gold standard is released by the task organizers.

References

1. MacKay, D.J.C.: Information Theory, Inference and Learning Algorithms. Cambridge University Press, Cambridge (2003)
2. Mirkin, B.G.: Mathematical Classification and Clustering. Springer, Heidelberg (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, pp. 281–297. University of California Press, Berkeley (1967)
4. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Communications of the ACM 18(11), 613–620 (1975)

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 used the WordNet and the link information for better performance, and applied 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 track, 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 on the feature selection, frequent subtrees selection, and 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 how those links add relevant information for the categorization of documents.

* Corresponding author.

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 INEX 2009 and INEX2010. 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 d_x of D is represented by a vector $d_x = (d_{(x,1)}, d_{(x,2)}, d_{(x,3)} \dots d_{(x,n)})$, where $d_{(x,i)}$ is the weight of the term t_i in the document d_x . SLVM is extended from VSM. The basic idea is to extract structure units from XML documents, then to 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 d_x can be represented by a document feature matrix $\chi \in R^{m \times n}$,

$$d_x = (d_{x(1)}, d_{x(2)}, d_{x(3)} \dots d_{x(m)}) \quad (1)$$

$$d_{x(i)} = (d_{x(i,1)}, d_{x(i,2)}, d_{x(i,3)} \dots d_{x(i,n)}) \quad (2)$$

Where m is the number of unit of document d_x , and $d_{x(i,j)}$ is the weight of the term t_j in the unit $d_{x(i)}$ of document 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].

$$BM25(q, d) = TF(q, d) \times IDF(q) \quad (3)$$

$$TF(q, d) = \frac{f(q, d) \times (k_1 + 1)}{f(q, d) + k_1 \times \left(1 - b + b \times \frac{d}{d_{avg}}\right)} \quad (4)$$

$$IDF(q) = \log \frac{N - n(q) + 0.5}{n(q) + 0.5} \quad (5)$$

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.
- $\frac{d}{D_{avg}}$: the ratio between the length of articles d and the average article length.
- k₁ and b: the classical BM25 parameters.

Parameter k₁ is able to control the term frequency saturation. Parameter b allows setting the importance of $\frac{d}{D_{avg}}$.

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 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

We have used both INEX2010 and INEX2009 collection of XML Mining Track for this experiment. As the collection of INEX2010 has been described above, the collection of INEX2009 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 average the number of category per 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, where 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 that about 2,000 features for each categorization can acquire a good value.

The INEX 2009 tuning results are shown in table1, where all the value with the BM25 have been improved, so that our system is based on the BM25 weights function for this year task.

Table 1. The result of BM25(INEX2010)

	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 INEX2009 and INEX2010 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 structure. We obtain the document of INEX2009 vectors at three different pruning levels, 0, 1/3, and 1/2, which yield a better performance for almost all the evaluation measures, and INEX2010 has five as shown in table 3.

Table 2 presents the effectiveness comparison of classification structures at various pruning levels of INEX2009, while used no more than 40,000 features Table 3 presents the effectiveness comparison of classification structures at various pruning levels of INEX2010, 80,000 features for all categories, as concede that calculation of the similarity between the words is very slowly, we have not used the information of WordNet in this part.

Table 2. The results of pruning the document tree (2009)

Prune	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

Table 3. The results of pruning the document tree (2010)

	Prune	Mi_p	Ma_p	Mi_r	Ma_r	Mi_f	Ma_f
Split_0	0%	0.453	0.39	0.632	0.55	0.524	0.454
Split_3	35%	0.458	0.409	0.635	0.555	0.534	0.466
Split_5	47%	0.467	0.407	0.639	0.558	0.535	0.467
Split_10	58%	0.474	0.412	0.635	0.560	0.539	0.472
Split_20	68%	0.454	0.392	0.633	0.557	0.525	0.457

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 $D_{i_{t_k}}(w'_1, w'_2, w'_3 \dots w'_m)$ and $D_{j_{t_k}}(w''_1, w''_2, w''_3 \dots w''_n)$.

The similarity between the t_k parts of the documents D_i and D_j is given as:

$$S_{ij_{t_k}} = \sum_{a=1}^{\text{Max}\{m,n\}} w'_a * w''_a \tag{6}$$

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:

$$S_{ij_{t_k}} = \sum_{a=1}^m \sum_{b=1}^n S_{w'_a w''_b} w'_a * w''_b \tag{7}$$

Where $S_{w'_a w''_b}$ is the similarity between the word w'_a and w''_b .

Given the similarity matrix between the words, we can change the $D_{j_{t_k}}$ as follow:

$$D_{j_{t_k}}(w''_1 + \sum_a^m S_{w''_1 w'_a}, w''_2 + \sum_a^m S_{w''_2 w'_a}, w''_3 + \sum_a^m S_{w''_3 w'_a} \dots w''_n + \sum_a^m S_{w''_n w'_a}) \tag{8}$$

When $D_{j_{t_k}}(w''_1, w''_2, w''_3 \dots w''_n)$ 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 collection, 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 $D_{j_{t_k}}$ don't contain word A, but it contains word B and word C, which the similarity with A is greater than 0, then we can use $S_{w_A w_B} + S_{w_A w_C}$ to smoothing the weight of word A in $D_{j_{t_k}}$.

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>” tag contain sthe 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_information>”, 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 $T(t_1, t_2, t_3 \dots t_s)$, and a given document D , which contain the tree t_k of T , and suppose the part of the t_k is

$$D_{tk}(w'_1, w'_2, w'_3 \dots w'_m) \tag{9}$$

Where m is the total number of features appearing in the tree t_k part of the document collection, w'_i is the weight of term i in document D and $0 \leq w'_i \leq 1$. Hence, if $w'_i > 0$, it means that word i exist in the tree t_k part of the document D . Therefore, document D will be represented by a vector of features

$$D_{tk}(w'_1, w'_2, w'_3 \dots w'_m, l'_1, l'_2, l'_3 \dots l'_n) \tag{10}$$

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 $0 \leq l'_i \leq 1$. Hence, if $l'_i > 0$, 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 calculate form the text information:

$$S(c|d) = \alpha \times S_l(c|d) + (1 - \alpha) \times S_t(c|d) \tag{11}$$

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 4 and 5. In the following tables, we can notice that, out-linked features work very well in the experiment, and this simply means that adding in out-link features can improve the performance of classification.

Table 4. The result of using the link for tuning (2009)

	Ma_acc	Mi_acc	Ma_roc	Mi_roc	Ma_prf	Mi_prf	Map
$\alpha=0$	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

Table 5. The result of using the link for tuning (2010)

	Prune	Mi_p	Ma_p	Mi_r	Ma_r	Mi_f	Ma_f
Split_10	58%	0.474	0.412	0.635	0.560	0.539	0.472
Split_10_tuning	58%	0.491	0.433	0.644	0.572	0.554	0.490

3.4 Other Tuning

In addition to the above methods, we also made some other optimization, for example smoothing IDF value in the formula of BM25, normalized the vector of the units in the document, and in particular the result of normalized the matrix of the documents performance better, so using this case, we pruning the trees of documents and using the out-link information, and the we get our result in the table 6, as follow.

Table 6. The result of using the link for tuning

runs	idf	Norm	Mi_p	Ma_p	Mi_r	Ma_r	Mi_f	Ma_f
	log	N	0.473	0.414	0.646	0.566	0.544	0.476
	1+log	N	0.476	0.416	0.641	0.563	0.543	0.476
	1+ log(f-v)/v	N	0.474	0.414	0.636	0.559	0.541	0.473
	1+log	Y	0.528	0.485	0.647	0.572	0.578	0.521
Result_link_ tuning	1+log	Y	0.543	0.499	0.656	0.583	0.591	0.535

3.5 Compare the Results

Table 7. 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.

Table 8. Compare the result with INEX 2010

Team	Run	Mi_p	Ma_p	Mi_r	Ma_r	Mi_f	Ma_f
Peking	T1_S3_linkN ⁺	0.432	0.356	0.656	0.613	0.517	0.444
	T1_S1_link0 ⁺	0.582	0.570	0.363	0.252	0.400	0.320
Qut	Inex10_sub ⁺	0.561	0.527	0.523	0.440	0.536	0.473
	Our result*	0.543	0.499	0.656	0.583	0.591	0.535

+ Result of INEX2010; * 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.

Another algorithm is based on AHC Clustering Algorithm, and the process of the Local AHC Clustering Algorithm is present as Figure 2.

- Given
- X: a set of N vectors
- K: desired number of clusters
- Begin
 - Select K random seeds $\{\vec{s}_1, \vec{s}_2, \dots, \vec{s}_k\}$ 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

Fig. 1. An algorithm for clustering

- 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 ,then continue, until the number of clusters meet our requirement:
 - Given:
 - Subsets D_i and a threshold τ
 - Calculate similarity matrix M
 - Do (if max element in M > threshold τ)
 - 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_if
 - End_do
 - Update row i and column j, set $m_{k,j} = m_{i,k} = \text{avg}(m_{c,k})$, 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

Fig. 2. The Local AHC Clustering Algorithm

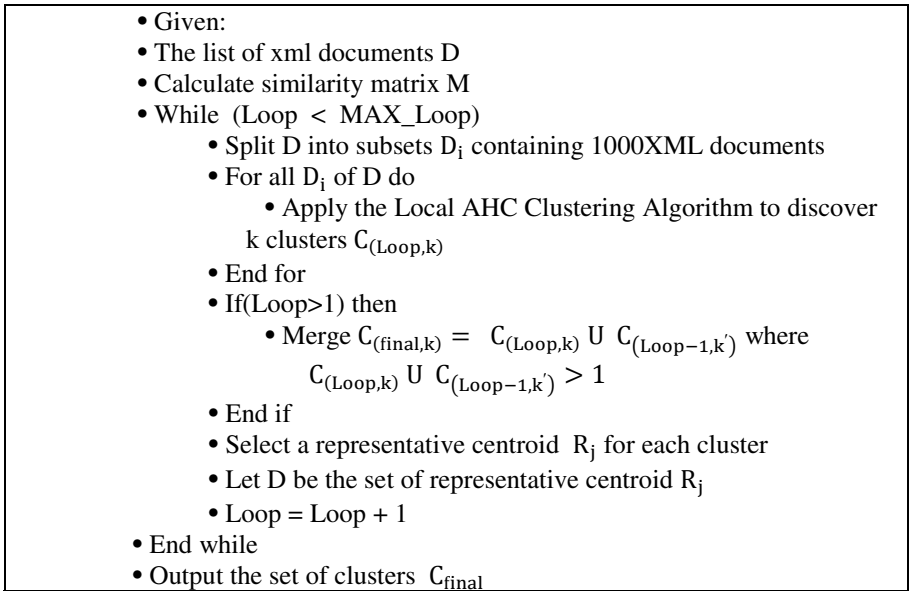


Fig. 3. Simple algorithm for clustering

Experimental results and analysis of these clustering algorithms will be showed in section 5.

5 INEX 2010 Results

We present in this section the official results obtained by our system during INEX 2010 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.

5.1 System Settings

Most of the settings given in section 3 have been reused for our INEX 2010 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.

5.2 Results

Table 9 present the official results of classification:

Table 9. The results of classification

Team	Run	Mi_p	Ma_p	Mi_r	Ma_r	Mi_t
Peking	T1_S3_link0	0.553	0.525	0.436	0.334	0.931
	T1_S3_link67	0.422	0.345	0.653	0.612	0.874
	T1_S3	0.433	0.368	0.635	0.582	0.886
	T1_S3_linkN	0.432	0.356	0.656	0.613	0.878
	T1_S2_linkN	0.456	0.389	0.615	0.559	0.893
	T1_S1_link0	0.582	0.570	0.363	0.252	0.934
	T2_S2_link0	0.562	0.536	0.422	0.321	0.934
	T2_S2	0.48.	0.414	0.574	0.510	0.910
	T2_S3_linkN	0.436	0.359	0.652	0.614	0.882
Qut	Inex10_sub	0.561	0.527	0.523	0.440	0.932
Team	Run	Ma_t	Mi_a	Ma_a	Mi_f	Ma_f
Peking	T1_S3_link0	0.974	0.891	0.942	0.518	0.446
	T1_S3_link67	0.928	0.860	0.914	0.508	0.435
	T1_S3	0.936	0.869	0.920	0.518	0.444
	T1_S3_linkN	0.931	0.864	0.917	0.517	0.444
	T1_S2_linkN	0.944	0.873	0.926	0.522	0.454
	T1_S1_link0	0.980	0.888	0.943	0.400	0.320
	T2_S2_link0	0.976	0.892	0.943	0.452	0.372
	T2_S2	0.954	0.883	0.932	0.521	0.452
	T2_S3_linkN	0.933	0.886	0.918	0.518	0.446
Qut	Inex10_sub	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 selected 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 category, 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 used, 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 selected 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 roe, 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, the number of categories that each document belonging to is more than for 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 10 present the official results of clustering.

Table 10. The results of clustering

	Runs	Mi_p	Ma_e	Mi_e	Ma_n	Mi_n
Pek	Result_1000_4	0.531	3.89	3.89	0.202	0.219
	Result_500_4	0.518	3.99	3.97	0.201	0.218
	Result_200_4	0.467	4.00	4.05	0.314	0.212
	result502_0.15	0.364	4.36	4.52	0.133	0.118
	Result_100_4	0.481	4.06	4.10	0.212	0.206
	result_0.18_1004	0.336	4.38	4.48	0.118	0.119
	Result_50_4	0.463	4.11	4.18	0.203	0.191
	Result_100_3	0.483	4.07	4.11	0.209	0.204
	Result_200_2	0.501	4.01	4.00	0.213	0.213
	Runs	Ma_fl	Mi_fl	Nmi	M_NC CG	S_NCC G
Pek	Result_1000_4	0.072	0.150	0.093	0.524	0.220
	Result_500_4	0.063	0.140	0.088	0.584	0.235
	Result_200_4	0.056	0.130	0.084	0.661	0.219
	result502_0.15	0.019	0.053	0.020	0.391	0.221
	Result_100_4	0.048	0.120	0.082	0.741	0.193
	result_0.18_1004	0.022	0.060	0.024	0.290	0.169
	Result_50_4	0.038	0.101	0.075	0.185	0.157
	Result_100_3	0.040	0.119	0.081	0.745	0.190
	Result_200_2	0.057	0.132	0.085	0.662	0.212

In table 10, 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 using link information can improve the result of classification by 2%, but the results in part 5 that we presented are not good, so we 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 included 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.: Chen, X.: A semi_structured document model for text mining. *Journal of computer science and Technology* 17(5), 603–610 (2002)
2. Salton, G., McGill, M.: *Introduction to Modern information Retrieval*. McGraw-Hill, New York (1983)
3. Robertson, S.E., Spark Jones, K.: Relevance weighting of search terms. *JASIST* 27(3), 129–146 (1976)
4. Chi, Y., Nijssen, S., Muntz, R.R., Kok, J.N.: Frequent Subtree Mining – An Overview. *Fundamenta Information* (2005)
5. Chi, Y., Yang, Y., Xia, Y., Muntz, R.R.: CMTreeMiner: Mining Both Closed and Maximal Frequent Subtrees. In: *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) 0-7695-2841-4/07
7. Yang, J., Zhang, F.: XML Document Classification using Extended VSM. In: *Pre-Proceedings of the Sixth Workshop of Initiative for the Evaluation of XML Retrieval*, Dagstuhl, Germany (2007)
8. Berry, M.: *Survey of Text Mining: Clustering*. Springer, Heidelberg (2003)
9. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: Nédellec, C., Rouveirol, C. (eds.) *ECML 1998*. LNCS, vol. 1398, Springer, Heidelberg (1998)
10. Yang, J., Wang, S.: Extended VSM for XML document classification using frequent subtrees. In: Geva, S., Kamps, J., Trotman, A. (eds.) *INEX 2009*. LNCS, vol. 6203, pp. 441–448. Springer, Heidelberg (2010)
11. Shin, K., Han, S.Y.: Fast clustering algorithm for information organization. *Proc. of the CILing 2003 Conference*. In: Gelbukh, A. (ed.) *CILing 2003*. LNCS, vol. 2588, pp. 619–622. Springer, Heidelberg (2003)

Author Index

- Aaron, José Cruz 347
Alexander, David 241, 250
Arvola, Paavo 1, 33
- Balderas, Carlos 219
Beckers, Thomas 227, 236
Beigbeder, Michel 44
Bellot, Patrice 118, 269
Beltrán, Beatriz 347
Bose, Aishwarya 336
Boudin, Florian 118
Buffoni, David 54
- Chappell, Timothy 303
Cherukuri, Ramakrishna 63
Crouch, Carolyn J. 63
Crouch, Donald B. 63
Cruz, Adrián 377
- da C. Hummel, Felipe 194
da Cunha, Iria 295
da Silva, Altigran S. 194
Deng, Zhi-Hong 319
Deveaud, Romain 118
De Vries, Christopher M. 363
Doucet, Antoine 98
- Fuhr, Norbert 227
- Gagnon, Michel 290
Gallinari, Patrick 54
Ganguly, Debasis 182, 313
Gao, Ning 319
Géry, Mathias 44
Geva, Shlomo 1, 241, 303, 363
Giguët, Emmanuel 128
- Hatano, Kenji 71
Hou, Jun 336
- Jaruskulchai, Chuleerat 351
Jia, Xiang-Fei 250
Jiang, Jia-Jian 319
Jones, Gareth J.F. 182, 313
- Kamps, Jaap 1, 98, 140
Kazai, Gabriella 98
Keyaki, Atsushi 71
Koolen, Marijn 98, 140
Korbar, Dennis 236
Kutty, Sangeetha 363
- Laender, Alberto H.F. 194
Landoni, Monica 98
Largerón, Christine 44
Larson, Ray R. 154
Lehtonen, Miro 82
León, Saul 219
Leveling, Johannes 182, 313
Li, Rongmei 89
Liang, Feng 383
Lucas, Nadine 128
Lv, Sheng-Long 319
- Mahule, Abhijeet 63
Medina-Urrea, Alfonso 282
Méndez-Cruz, Carlos-Francisco 282
Mitra, Mandar 182
Miyazaki, Jun 71
Montes, Azucena 377
Moriceau, Veronique 269
Moro, Mirella M. 194
- Nayak, Richi 336, 363
Nordlie, Ragnar 164, 227
- Pal, Sukomal 182
Palchowdhury, Sauparna 182
Pharo, Nils 227
Pinto, David 219, 347, 377
Preminger, Michael 164
- Ramírez, Georgina 206
Ramírez, Javier 295
- SanJuan, Eric 269
Schenkel, Ralf 1
Seck, Howard 44
Sierra Martínez, Gerardo 282
Somodevilla, María Josefa 347
Soriano-Morales, Edmundo-Pavel 282

- Tagarelli, Andrea 363
Tannier, Xavier 269
Thom, James A. 332
Torres-Moreno, Juan-Manuel 290
Tovar, Mireya 219, 377
Trotman, Andrew 1, 171, 241, 250

Usunier, Nicolas 54

Vadlamudi, Sandeep 63
Vainio, Johanna 1, 33
van der Weide, Theo 89
Vázquez, Blanca 377

Vilariño, Darnes 219, 347, 377
Vivaldi, Jorge 295

Wang, Qiuyue 171
Wang, Songlin 383
Wichaiwong, Tanakorn 351
Wood, Vaughn 250
Wu, Chen 332

Yang, Jianwu 383
Yu, Hang 319

Zhang, Jinglan 336