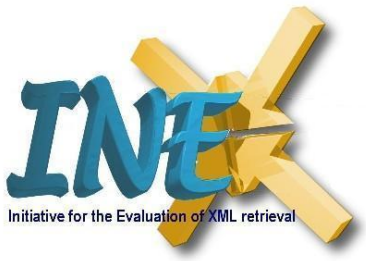


INEX 2011 Workshop Pre-proceedings

Shlomo Geva, Jaap Kamps, Ralf Schenkel (editors)

December 12–14, 2011
Hofgut Imsbach, Saarbrücken, Germany
http://inex.mmci.uni-saarland.de/

Attribution
http://creativecommons.org/licenses/by/3.0/

Copyright ©2011 remains with the author/owner(s).

The unreviewed pre-proceedings are collections of work submitted before the December workshops. They are not peer reviewed, are not quality controlled, and contain known errors in content and editing. The proceedings, published after the Workshop, is the authoritative reference for the work done at INEX.

Published by: IR Publications, Amsterdam. ISBN 978-90-814485-8-1.
INEX Working Notes Series, Volume 2011.

Preface

Welcome to the tenth 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. Now, in its tenth year, INEX is an established evaluation forum, with over 100 organizations worldwide registered and over 30 groups participating actively in at least one of the tracks.

INEX 2011 was an exciting year for INEX in which a number of new tasks and tracks started, including Social Search, Faceted Search, Snippet Retrieval, and Tweet Contextualization. In total five research tracks were included, which studied different aspects of focused information access:

Books and Social Search Track investigating techniques to support users in searching and navigating books, metadata and complementary social media. The *Social Search for Best Books Task* studies the relative value of authoritative metadata and user-generated content using a collection based on data from Amazon and LibraryThing. The *Prove It Task* asks for pages confirming or refuting a factual statement, using a corpus of the full texts of 50k digitized books.

Data Centric Track investigating retrieval over a strongly structured collection of documents based on IMDb. The *Ad Hoc Search Task* has informational requests to be answered by the entities in IMDb (movies, actors, directors, etc.). The *Faceted Search Task* asks for a restricted list of facets and facet-values that will optimally guide the searcher toward relevant information.

Question Answering Track investigating tweet contextualization, answering questions of the form "what is this tweet about?" with a synthetic summary of contextual information grasped from Wikipedia and evaluated by both the relevant text retrieved, and the "last point of interest."

Relevance Feedback Track investigate the utility of incremental passage level relevance feedback by simulating a searcher's interaction. An unconventional evaluation track where submissions are executable computer programs rather than search results.

Snippet Retrieval Track investigate how to generate informative snippets for search results. Such snippets should provide sufficient information to allow the user to determine the relevance of each document, without needing to view the document itself.

Two more tracks were announce, continuations of the Interactive Track and the Web Service Discovery Track, but these failed to complete in time for INEX 2011.

The aim of the INEX 2011 workshop is to bring together researchers who participated in the INEX 2011 campaign. During the past year participating organizations contributed to the building of a large-scale test collection by creating topics, performing retrieval runs and providing relevance assessments. The workshop concludes the results of this large-scale effort, summarizes and addresses encountered issues and devises a work plan for the future evaluation of XML retrieval systems.

All INEX tracks start from having available suitable text collections. We gratefully acknowledge the data made available by: Amazon and LibraryThing (Books and Social Search Track), Microsoft Research (Books and Social Search Track), the Internet Movie Database (Data Centric Track), and the Wikimedia Foundation (Question Answering Track and Relevance Feedback Track).

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

December 2011

Shlomo Geva
Jaap Kamps
Ralf Schenkel

Organization

Steering Committee

Charles L. A. Clarke (University of Waterloo)
Norbert Fuhr (University of Duisburg-Essen)
Shlomo Geva (Queensland University of Technology)
Jaap Kamps (University of Amsterdam)
Mounia Lalmas (Yahoo! Research)
Stephen E. Robertson (Microsoft Research Cambridge)
Ralf Schenkel (Max-Planck-Institut für Informatik)
Andrew Trotman (University of Otago)
Ellen M. Voorhees (NIST)
Arjen P. de Vries (CWI)

Chairs

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

Track Organizers

Books and Social Search

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

Data Centric

Jaap Kamps (University of Amsterdam)
Maarten Marx (University of Amsterdam)
Georgina Ramírez Camps (Universitat Pompeu Fabra)
Martin Theobald (Max-Planck-Institut für Informatik)
Qiuyue Wang (Renmin University of China)

Question Answering

Patrice Bellot (University of Avignon)
Véronique Moriceau (LIMSI-CNRS, University Paris-Sud 11)
Josiane Mothe (IRIT, Toulouse)
Eric SanJuan (University of Avignon)
Xavier Tannier (LIMSI-CNRS, University Paris-Sud 11)

Relevance Feedback

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

Snippet Retrieval

Shlomo Geva (Queensland University of Technology)
Mark Sanderson (RMIT)
Falk Scholer (RMIT)
Andrew Trotman (University of Otago)
Matthew Trappett (Queensland University of Technology)

Table of Contents

Front matter.

Books and Social Search Track.

Data Centric Track.

Question Answering Track.

Relevance Feedback Track.

Snippet Retrieval Track.

Back matter.

Overview of the INEX 2011 Book Track

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

[1] Microsoft Research, United Kingdom
v-gabkaz@microsoft.com
[2] University of Amsterdam, Netherlands
{marijn.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 2011 Book Track is to evaluate approaches for supporting users in reading, searching, and navigating book metadata and full texts of digitized books. The investigation is focused around four tasks: 1) the Social Search for Best Books task aims at comparing traditional and user-generated book metadata for retrieval, 2) the Prove It task evaluates focused retrieval approaches for searching books, 3) the Structure Extraction task tests automatic techniques for deriving structure from OCR and layout information, and 4) the Active Reading task aims to explore suitable user interfaces for eBooks enabling reading, annotation, review, and summary across multiple books. We report on the setup and the results of the track.

1 Introduction

Prompted by the availability of large collections of digitized books, e.g., the Million Book project[5] and the Google Books Library project,[6] the Book Track was launched in 2007 with the aim to promote research into techniques for supporting users in searching, navigating and reading book metadata and full texts of digitized books. Toward this goal, the track provides opportunities to explore research questions around four areas:

- The relative value of professional and user-generated metadata for searching large collections of books,
- Information retrieval techniques 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.

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

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

Table 1. Active participants of the INEX 2011 Book Track, the task they were active in, and number of contributed runs (SB = Social Search for Best Books, PI = Prove It, SE = Structure Extraction, AR = Active Reading)

ID	Institute	Tasks	Runs
4	University of Amsterdam	SB	6
7	Oslo University College	PI	15
18	Universitat Pompeu Fabra	SB	6
34	Nankai University	SE	4
50	University of Massachusettes	PI	6
54	Royal School of Library and Information Science	SB	4
62	University of Avignon	SB	6
113	University of Caen	SE	3
	Microsoft Development Center Serbia	SE	1
	Xerox Research Centre Europe	SE	2

1. The Social Search for Best Books (SB) task, framed within the user task of searching a large online book catalogue for a given topic of interest, aims at comparing retrieval effectiveness from traditional book descriptions, e.g., library catalogue information, and user-generated content such as reviews, ratings and tags.
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 2011. First, in Section 2, we give a brief summary of the participating organisations. The four task are described in detail in the following sections: the SB task in Section 3, the PI task in Section 4, the SE task in Section 5 and the ART in Section 6. We close in Section 7 with a summary and plans for INEX 2012.

2 Participating Organisations

A total of 47 organisations registered for the track (compared with 82 in 2010, 84 in 2009, 54 in 2008, and 27 in 2007). At the time of writing, we counted 10 active groups (compared with 16 in 2009, 15 in 2008, and 9 in 2007), see Table 1.[7]

[7] The last two groups participated in the SE task via ICDAR but did not register for INEX, hence have no ID.

3 The Social Search for Best Books Task

The goal of the Social Search for Best Books (SB) task is to evaluate the relative value of controlled book metadata, such as classification labels, subject headings and controlled keywords, versus user-generated or social metadata, such as tags, ratings and reviews, for retrieving the most relevant books for a given user request. Controlled metadata, such as the Library of Congress Classification and Subject Headings, is rigorously curated by experts in librarianship. It is used to index books to allow highly accurate retrieval from a large catalogue. However, it requires training and expertise to use effectively, both for indexing and for searching. On the other hand, social metadata, such as tags, are less rigorously defined and applied, and lack vocabulary control by design. However, such metadata is contributed directly by the users and may better reflect the terminology of everyday searchers. Clearly, both types of metadata have advantages and disadvantages. The task aims to investigate whether one is more suitable than the other to support different types of search requests or how they may be fruitfully combined.

The SB task aims to address the following research questions:

- How can a system take full advantage of the available metadata for searching in an online book collections?
- What is the relative value of social and controlled book metadata for book search?
- How does the different nature of these metadata descriptions affect retrieval performance for different topic types and genres?

3.1 Scenario

The scenario is that of a user turning to Amazon Books and LibraryThing to search for books they want to read, buy or add to their personal catalogue. Both services host large collaborative book catalogues that may be used to locate books of interest.

On LibraryThing, users can catalogue the books they read, manually index them by assigning tags, and write reviews for others to read. Users can also post messages on a discussion forum asking for help in finding new, fun, interesting, or relevant books to read. The forums allow users to tap into the collective bibliographic knowledge of hundreds of thousands of book enthusiasts. On Amazon, users can read and write book reviews and browse to similar books based on links such as "customers who bought this book also bought... ".

Users can search online book collections with different intentions. They can search for specific books of which they know all the relevant details with the intention to obtain them (buy, download, print). In other cases, they search for a specific book of which they do not know those details, with the intention of identifying that book and find certain information about it. Another possibility is that they are not looking for a specific book, but hope to discover one or more books meeting some criteria. These criteria can be related to subject, author,

genre, edition, work, series or some other aspect, but also more serendipitously, such as books that merely look interesting or fun to read.

Although book metadata can often be used for browsing, this task assumes a user issues a query to a retrieval system, which returns a (ranked) list of book records as results. This query can be a number of keywords, but also one or more book records as positive or negative examples. We assume the user inspects the results list starting from the top and works her way down until she has either satisfied her information need or gives up. The retrieval system is expected to order results by relevance to the user's information need.

3.2 Task description

The SB task is to reply to a user's request that has been posted on the LibraryThing forums (see Section 3.5) by returning a list of recommended books. The books must be selected from a corpus that consists a collection of book metadata extracted from Amazon Books and LibraryThing, extended with associated records from library catalogues of the Library of Congress and the British Library (see the next section). The collection includes both curated and social metadata. User requests vary from asking for books on a particular genre, looking for books on a particular topic or period or books by a given author. The level of detail also varies, from a brief statement to detailed descriptions of what the user is looking for. Some requests include examples of the kinds of books that are sought by the user, asking for similar books. Other requests list examples of known books that are related to the topic but are specifically of no interest. The challenge is to develop a retrieval method that can cope with such diverse requests. Participants of the SB task are provided with a set of book search requests and are asked to submit the results returned by their systems as ranked lists.

3.3 Submissions

We want to evaluate the book ranking of retrieval systems, specifically the top ranks. We adopt the submission format of TREC, with a separate line for each retrieval result, consisting of six columns:

1. topic_id: the topic number, which is based on the LibraryThing forum thread number.
2. Q0: the query number. Unused, so should always be Q0.
3. isbn: the ISBN of the book, which corresponds to the file name of the book description.
4. rank: the rank at which the document is retrieved.
5. rsv: retrieval status value, in the form of a score. For evaluation, results are ordered by descending score.
6. run_id: a code to identifying the participating group and the run.

Participants are allowed to submit up to six runs, of which at least one should use only the *title* field of the topic statements (the topic format is described in Section 3.5). For the other five runs, participants could use any field in the topic statement.

3.4 Data

To study the relative value of social and controlled metadata for book search, we need a large collection of book records that contains controlled subject headings and classification codes as well as social descriptions such as tags and reviews, for a set of books that is representative of what readers are searching for. We use the Amazon/LibraryThing corpus crawled by the university of Duisburg-Essen for the INEX Interactive Track [1].

The collection consists of 2.8 million book records from Amazon, extended with social metadata from LibraryThing. This set represents the books available through Amazon. These records contain title information as well as a Dewey Decimal Classification (DDC) code and category and subject information supplied by Amazon. From a sample of Amazon records we noticed the subject descriptors to be noisy, with many inappropriately assigned descriptors that seem unrelated to the books to which they have been assigned.

Each book is identified by ISBN. Since different editions of the same work have different ISBNs, there can be multiple records for a single intellectual work. The corpus consists of a collection of 2.8 million records from Amazon Books and LibraryThing.com. See https://inex.mmci.uni-saarland.de/data/nd-agreements.jsp for information on how to get access to this collection. Each book record is an XML file with fields like ¡isbn¿, ¡title¿, ¡author¿, ¡publisher¿, ¡dimensions¿, ¡numberofpage¿ and ¡publicationdate¿. Curated metadata comes in the form of a Dewey Decimal Classification in the ¡dewey¿ field, Amazon subject headings are stored in the ¡subject¿ field, and Amazon category labels can be found in the ¡browseNode¿ fields. The social metadata from Amazon and LibraryThing is stored in the ¡tag¿, ¡rating¿, and ¡review¿ fields. The full list of fields is shown in Table 2.

How many of the book records have curated metadata? There is a DDC code for 61% of the descriptions and 57% of the collection has at least one subject heading.The classification codes and subject headings cover the majority of records in the collection.

More than 1.2 million descriptions (43%) have at least one review and 82% of the collection has at least one LibraryThing tag.

The distribution of books over the Amazon subject categories shows that *Literature*, *History*, *Professional and Technical* and *Religion* are some of the largest categories (see Table 3). There are also administrative categories related to sales, edition (paperback, hardcover) and others, but we show only the genre-related categories. If we look at the distribution over DDC codes (showing only the main classes in Table 4), we see a somewhat different distribution. *Literature* is still the largest class, but is followed by *Social sciences*, *Arts and recreation*, *Technology*, then *History* and *Religion*. Note that a book has only one DDC

Table 2. A list of all element names in the book descriptions

tag name			
book	similarproducts	title	imagecategory
dimensions	tags	edition	name
reviews	isbn	dewey	role
editorialreviews	ean	creator	blurber
images	binding	review	dedication
creators	label	rating	epigraph
blurbers	listprice	authorid	firstwordsitem
dedications	manufacturer	totalvotes	lastwordsitem
epigraphs	numberofpages	helpfulvotes	quotation
firstwords	publisher	date	seriesitem
lastwords	height	summary	award
quotations	width	editorialreview	browseNode
series	length	content	character
awards	weight	source	place
browseNodes	readinglevel	image	subject
characters	releasedate	imageCategories	similarproduct
places	publicationdate	url	tag
subjects	studio	data	

Table 3. Amazon category distribution (in percentages)

Category	%	**Category**	%
Non-fiction	20	Science	7
Literature and fiction	20	Fiction	7
Children	14	Literature	7
History	13	Christianity	7
Reference	11	Health, Mind and Body	6
Professional and Technical.	11	Arts and Photography	5
Religion and Spirituality	10	Business and Investing	5
Social science	10	Biography and Memoirs	5

Table 4. Distribution over DDC codes (in percentages)

DDC main class	%
Computer science, information and general works	4
Philosophy and psychology	4
Religion	8
Social sciences	16
Language	2
Science (including mathematics)	5
Technology and applied Science	13
Arts and recreation	13
Literature	25
History, geography, and biography	11

code—it can only have one physical location on a library shelf—but can have multiple Amazon categories, which could explain the difference in distribution. Note also that all but 296 books in the collection have at least one Amazon category, while only 61% of the records have DDC codes.

3.5 Information needs

LibraryThing users discuss their books in the discussion forums. Many of the topic threads are started with a request from a member for interesting, fun new books to read. They describe what they are looking for, give examples of what they like and do not like, indicate which books they already know and ask other members for recommendations. Other members often reply with links to works catalogued on LibraryThing, which have direct links to the corresponding records on Amazon. These requests for recommendation are natural expressions of information needs for a large collection of online book records. We aim to evaluate the SB task using a selection of these forum topics.

The books suggested by members in replies to the initial message are collected in a list on the side of the topic thread (see Figure 1). A technique called *touchstone* can be used by members to easily identify books they mention in the topic thread, giving other readers of the thread direct access to a book record on LibraryThing, with associated ISBNs and links to Amazon. We use these suggested books as initial relevance judgements for evaluation. Some of these touchstones identify an incorrect book, and suggested books may not always be what the topic creator asked for, but merely be mentioned as a negative example or for some other reason. From this it is clear that the collected list of suggested books can contain false positives and is probably incomplete as not all relevant books will be suggested (false negatives), so may not be appropriate for reliable evaluation. We discuss this in more detail in Section 3.7. We first describe how we created a large set of topics, then analyse what type of topics we ended up with and how suitable they for this task.

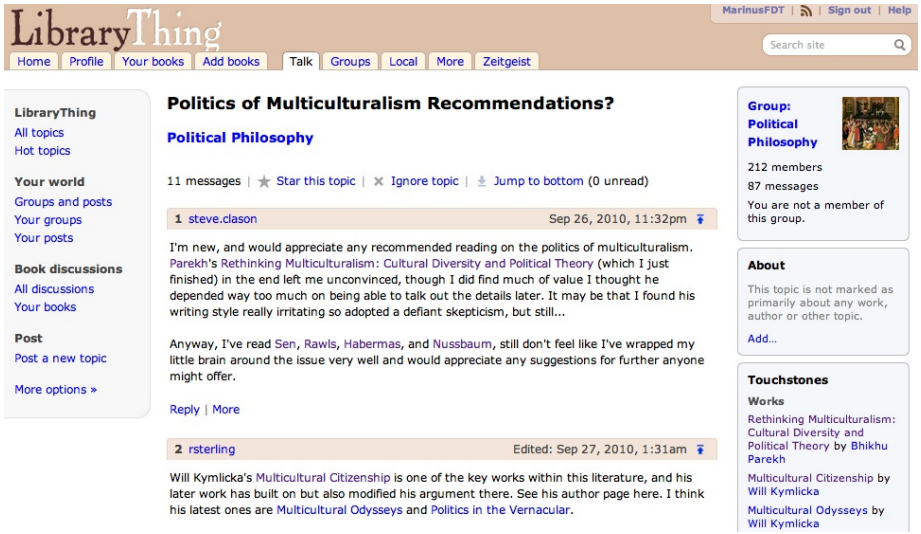


Fig. 1. A topic thread in LibraryThing, with suggested books listed on the right hand side.

Topic analysis We crawled 18,427 topic threads from 1,560 discussion groups. From these, we extracted 943 topics where the initial message contains a request for book suggestions. Each topic has a title and is associated with a group on the discussion forums. For instance, topic 99309 in Figure 1 has title *Politics of Multiculturalism Recommendations?* and was posted in the group *Political Philosophy*. Not all titles are good descriptions of the information need expressed in the initial message. To identify which of these 943 topics have good descriptive titles, we used the titles as queries and retrieved records from the Amazon/LibraryThing collection and evaluated them using the suggested books collected through the touchstones. We selected all topics for which at least 50% of the suggested books were returned in the top 1000 results and manually labelled them with information about topic type, genre and specificity and extracted positive and negatives example books and authors mentioned in the initial message. Some topics had very vague requests or relied on external source to derive the information need (such as recommendations of books listed on certain web page), leaving 211 topics in the official test topic set from 122 different discussion groups.

To illustrate how we marked up the topics, we show topic 99309 from Figure 1 as an example:

```
<topic id="99309">
  <title>Politics of Multiculturalism</title>
  <group>Political Philosophy</group>
  <narrative>I'm new, and would appreciate any recommended reading on the
    politics of multiculturalism. <author>Parekh</author>'s
    <work id="164382"> Rethinking Multiculturalism: Cultural Diversity and
    Political Theory</work> (which I just finished) in the end left me un-
```

```
    convinced, though I did find much of value I thought he depended way
    too much on being able to talk out the details later. It may be that I
    found his writing style really irritating so adopted a defiant skepti-
    cism, but still... Anyway, I've read <author>Sen</author>, <author>
    Rawls</author>, <author>Habermas</author>, and <author>Nussbaum
    </author>, still don't feel like I've wrapped my little brain around
    the issue very well and would appreciate any suggestions for further
    anyone might offer.
  </narrative>
  <type>subject</type>
  <genre>politics</genre>
  <specificity>narrow</specificity>
  <similar>
    <work id="164382">
      <isbn>0333608828</isbn>
      <isbn>0674004361</isbn>
      <isbn>1403944539</isbn>
      <isbn>0674009959</isbn>
    </work>
    <author>Parekh</author>
    <author>Sen</author>
    <author>Rawls</author>
    <author>Habermas</author>
    <author>Nussbaum</author>
  </similar>
  <dissimilar><dissimilar>
</topic>
```

The distribution over topic type is shown on the left side of Table 5. The majority of topics have subject-related book requests. For instance, the topic in Figure 1 is a subject-related request, asking for books about politics and multiculturalism. Most requests (64%) are subject-related, followed by author-related (15%), then series (5%), genre (4%), edition and known-item (both 3%). Some topics can be classified with 2 types, such as subject and genre. For instance, in one topic thread, the topic creator asks for biographies of people with eating disorders. In this case, the subject is *people with eating disorders* and the genre is *biography*. The topic set covers a broad range of topic types, but for work- and language-related topics the numbers are too small to be representative. We will conduct a more extensive study of the topics to see if this distribution is representative or whether our selection method has introduced some bias.

Next, we classified topics by genre, roughly based on the main classes of the LCC and DDC (see right side of Table 5), using separate classes for philosophy and religion (similar to DDC, while LCC combines them in one main class). The two most requested genres are literature (42%, mainly prose and some poetry), and history (28%). We only show the 12 most frequent classes. There are more main classes represented by the topics, such as law, psychology and genealogy, but they only represent one or two topics each. If we compare this distribution with the Amazon category and DDC distributions in Tables 3 and 4, we see

Table 5. Distribution of topic types and genres

Type	Freq.	Genre	Freq.
subject	134	literature	89
author	32	history	60
series	10	biography	24
genre	8	military	16
edition	7	religion	16
known-item	7	technology	14
subject & genre	7	science	11
work	2	education	8
genre & work	1	politics	4
subject & author	1	philosophy	4
language	1	medicine	3
author & genre	1	geography	3

that military books are more popular among LibraryThing forum users than is represented by the Amazon book corpus, while social science is less popular. Literature, history, religion, technology are large class in both the book corpus and the topic set. The topic set is a reasonable reflection of the genre distribution of the books in the Amazon/LibraryThing collection.

Furthermore, we added labels for specificity. The specificity of a topic is somewhat subjective and we based it on a rough estimation of the number of relevant books. It is difficult to come up with a clear threshold between broad and narrow, and equally hard to estimate how many books would be relevant. Broad topics have requests such as recommendations within a particular genre ("please recommend good science fiction books."), for which thousands of books could be considered relevant. The topic in Figure 1 is an example of a narrow topic. There are 177 topics labelled as narrow (84%), 34 topics as broad (16%). We also labelled books mentioned in the initial message as either positive or negative examples of what the user is looking for. There are 58 topics with positive examples (27%), 9 topics with negative examples (4%). These topics could be used as query-by-example topics, or maybe even for recommendation. The examples add further detail to the expressed information need and increase the realism of the topic set.

We think this topic set is representative of book information needs and expect it to be suitable for evaluating book retrieval techniques. We note that the titles and messages of the topic threads may be different from what these users would submit as queries to a book search system such as Amazon, LibraryThing, the Library of Congress or the British Library. Our topic selection method is an attempt to identify topics where the topic title describes the information need. In the first year of the task, we ask the participants to generate queries from the title and initial message of each topic. In the future, we could approach the topic creators on LibraryThing and ask them to supply queries or set up

Table 6. Statistics on the number of recommended books for the 211 topics from the LT discussion groups

# rel./topic	# topics	min.	max.	median	mean	std. dev.
All	211	1	79	7	11.3	12.5
Fiction	89	1	79	10	16.0	15.8
Non-fiction	132	1	44	6	8.3	8.3
Subject	142	1	68	6	9.6	10.0
Author	34	1	79	10	15.9	17.6
Genre	16	1	68	7	13.3	16.4

a crowdsourcing task where participants provide queries while searching the Amazon/LibraryThing collection for relevant books.

Touchstone Recommendations as Judgements We use the recommended books for a topic as relevance judgements for evaluation. Each book in the Touchstone list is considered relevant. How many books are recommended to LT members requesting recommendations in the discussion groups? Are other members compiling exhaustive lists of possibly interesting books or do they only suggest a small number of the best available books? Statistics on the number of books recommended for the 211 topics are given in Table 6.

The number of relevant books per topic ranges between 1 and 79 with an mean of 11.3. The median is somewhat lower (7), indicating that most of the topics have a small number of recommended books. The topics requesting fiction books have more relevant books (16 on average) than the topics requesting non-fiction (8.3 on average). Perhaps this is because there is both more fiction in the collection and more fiction related topics in the topic set. The latter point suggests that fiction is more popular among LT members, such that requests for books get more responses.

The breakdown over topic types *Subject*, *Author* and *Genre* shows that subject related topics have fewer suggested books than author and genre related topics. This is probably related to the distinction between fiction and non-fiction. Most of the *Subject* topics are also *Non-fiction* topics, which have fewer recommended books than *Fiction* books.

ISBNs and intellectual works

Each record in the collection corresponds to an ISBN, and each ISBN corresponds to a particular intellectual work. However, an intellectual work can have different editions, each with their own ISBN. The ISBN-to-work relation is a many-to-one relation. In many cases, we assume the user is not interested in all the different editions, but in different intellectual works. For evaluation we collapse multiple ISBN to a single work. The highest ranked ISBN is evaluated and all lower ranked ISBNs ignored. Although some of the topics on LibraryThing are requests to recommend a particular edition of a work—in which case the distinction between different ISBNs for the same work are important—we leave

them out of the relevance assessment phase for this year to make evaluation easier.

However, one problem remains. Mapping ISBNs of different editions to a single work is not trivial. Different editions may have different titles and even have different authors (some editions have a foreword by another author, or a translator, while others have not), so detecting which ISBNs actually represent the same work is a challenge. We solve this problem by using mappings made by the collective work of LibraryThing members. LT members can indicate that two books with different ISBNs are actually different manifestations of the same intellectual work. Each intellectual work on LibraryThing has a unique work ID, and the mappings from ISBNs to work IDs is made available by LibraryThing.[8]

However, the mappings are not complete and might contain errors. Furthermore, the mappings form a many-to-many relationship, as two people with the same edition of a book might independently create a new book page, each with a unique work ID. It takes time for members to discover such cases and merge the two work IDs, which means that at time, some ISBNs map to multiple work IDs. LibraryThing can detect such cases but, to avoid making mistakes, leaves it to members to merge them. The fraction of works with multiple ISBNs is small so we expect this problem to have a negligible impact on evaluation.

3.6 Crowdsourcing Judgements on Relevance and Recommendation

Members recommend books they have read or that they know about. This may be only a fraction of all the books that meet the criteria of the request. The list of recommended books in a topic thread may therefore be an incomplete list of appropriate books. Retrieval systems can retrieve many relevant books that are not recommended in the thread. On the other hand, LT members might leave out certain relevant books on purpose because they consider these books inferior to the books they do suggest.

To investigate this issue we ran an experiment on Amazon Mechanical Turk, where we asked workers to judge the relevance and make recommendations for books based on the descriptions from the Amazon/LT collection. For the PI task last year we found that relevance judgements for digitised book pages from AMT give reliable system rankings [5]. We expect that judging the relevance of an Amazon record given a narrative from the LibraryThing discussion forum has a lower cognitive load for workers, and with appropriate quality-control measures built-in, we expect AMT judgements on book metadata to be useful for reliable evaluation as well. An alternative or complement is to ask task participants to make judgements.

We pooled the top 10 results of all official runs for 24 topics and had each book judged by 3 workers. We explicitly asked workers to first judge the book on topical relevance and with a separate question asked them to indicate whether they would also recommend it as one the best books on the requested topic.

Topic selection

[8] See: http://www.librarything.com/feeds/thingISBN.xml.gz

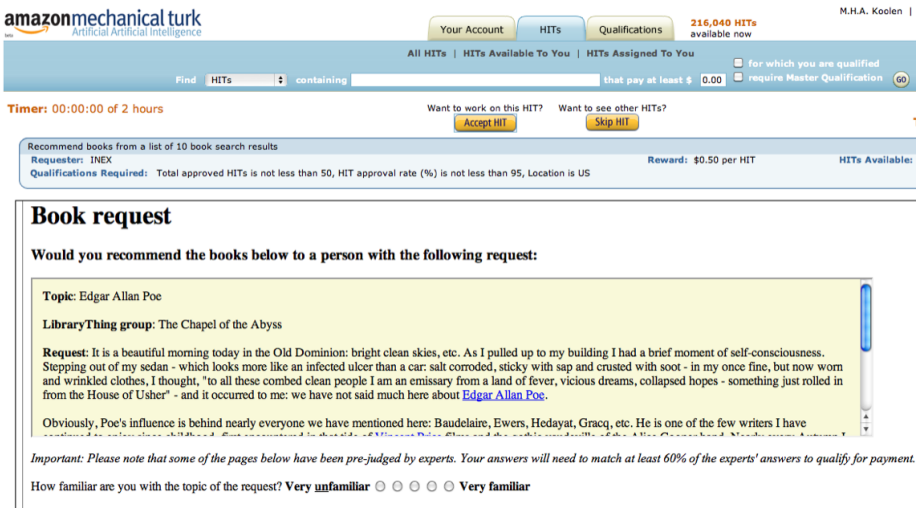


Fig. 2. Snapshot of the AMT book request.

For the Mechanical Turk judgements, we 24 topics from the set of 211, 12 fiction and 12 non-fiction. We selected the following 12 fiction topics: 17299, 25621, 26143, 28197, 30061, 31874, 40769, 74433, 84865, 94888, 92178 and 106721. We selected the following 12 non-fiction topics: 3963, 12134, 14359, 51583, 65140, 83439, 95533, 98106, 100674, 101766, 107464 and 110593.

Pooling
We pooled the top 10 results per topic of all 22 submitted runs. If the resulting pool was smaller than 100 books, we continued the round-robin pooling until each pool contained at least 100 books.

Generating HITs
Each HIT contains 10 books, with at least one book that was recommended in the topic thread on the LibraryThing discussion group for validation. In total, 269 HITs were generated, and each HIT was assigned to 3 workers, who got paid \$0.50 per HIT. With a 10% fee charged by Amazon per HIT, the total cost was $269 * 3 * \$0.50 * 1.1 = \$443.85$.

HIT design
The design of the HIT is illustrated in Figures 2, 3, 4 and 5. The HIT starts with short instructions explaining what the task is and what the goal of the task is, after which the request is shown (see Figure 2). After the request, worker get a list of 10 book questionnaires, with each questionnaire containing frame with official metadata (Figure 3), user-generated metadata (Figure 4) and a list of questions (Figure 5). The official metadata consists of the title information, publisher information and the Amazon categories, subject headings and classification information. The user-generated metadata consists of user reviews and ratings from Amazon and user tags from LibraryThing.

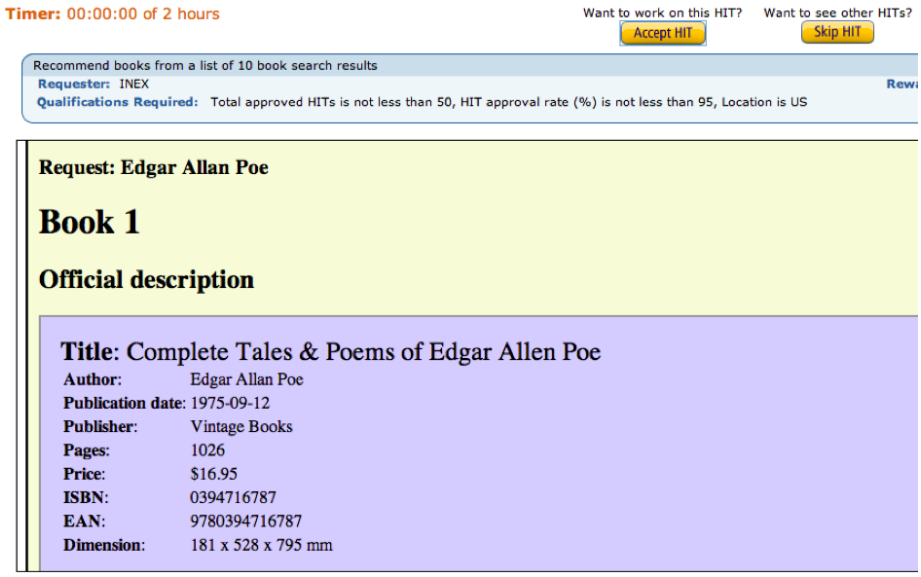


Fig. 3. Snapshot of the AMT design for the official description.

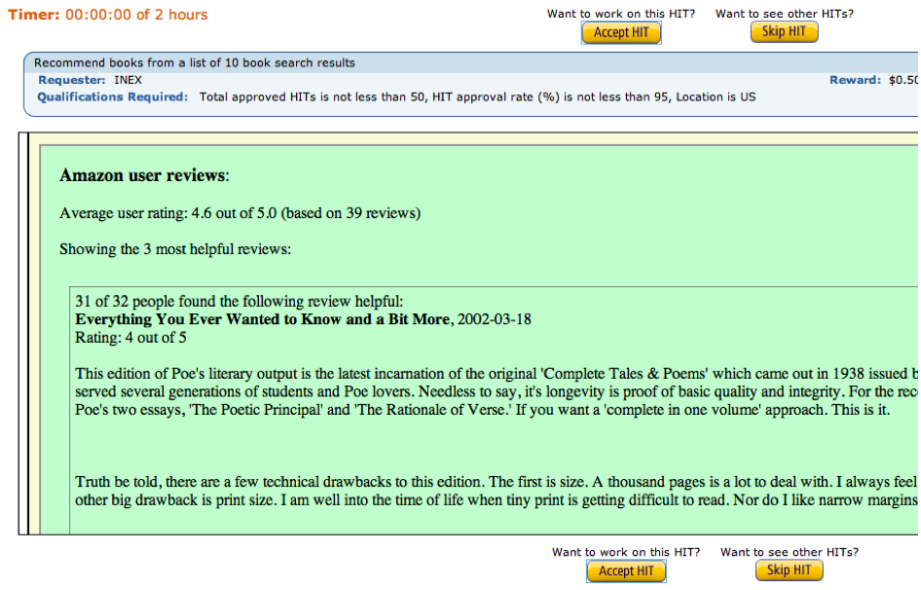


Fig. 4. Snapshot of the AMT design for the user-generated description.

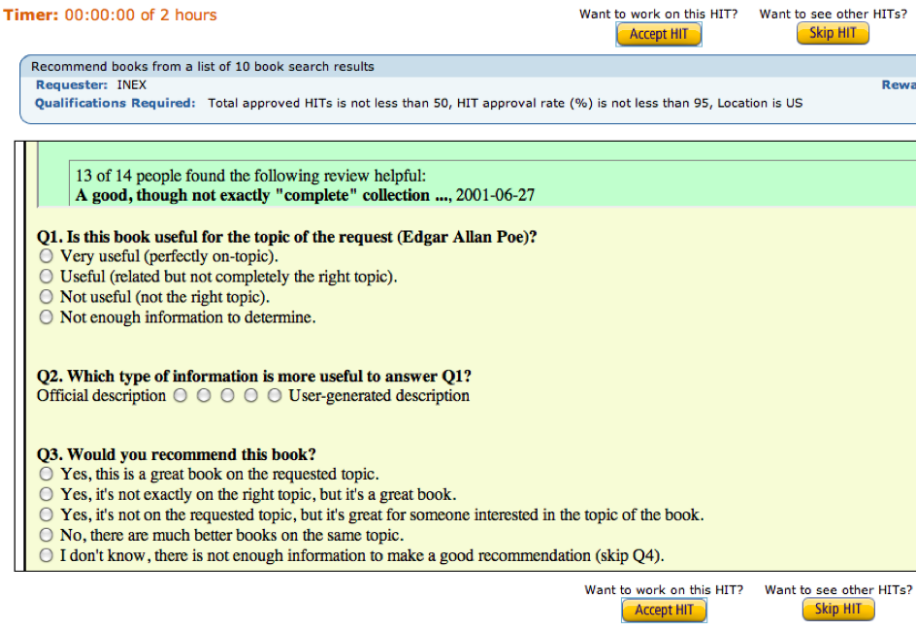


Fig. 5. Snapshot of the AMT questionnaire design.

The questionnaire has 5 questions:

- **Q1. Is this book useful for the topic of the request?** Here workers can choose between
  - *perfectly on-topic*,
  - *related but not completely the right topic*,
  - *not the right topic* and
  - *not enough information*.
- **Q2. Which type of information is more useful to answer Q1?** Here workers have to indicate whether the official or user-generated metadata is more useful to determine relevance.
- **Q3. Would you recommend this book?** Here workers can choose between
  - *great book on the requested topic*,
  - *not exactly on the right topic, but it's a great book*,
  - *not on the requested topic, but it's great for someone interested in the topic of the book*,
  - *there are much better books on the same topic,* and
  - *not enough information to make a good recommendation*.
- **Q4. Which type of information is more useful to answer Q3?** Here workers have to indicate whether the official or user-generated metadata is more useful to base their recommendation on.

Table 7. Statistics on the number of recommended books for the 211 topics from the LT discussion groups

# rel./topic	# topics	min.	max.	median	mean	std. dev.
LT all	211	1	79	7	11.3	12.5
LT Fiction	89	1	79	10	16.0	15.8
LT Non-fiction	132	1	44	6	8.3	8.3
LT (24 AMT topics)	24	2	79	7	15.7	19.3
AMT all	24	4	56	25	25.0	12.7
AMT fiction	12	4	30	25	22.8	10.8
AMT non-fiction	12	4	56	29	27.3	13.7

– **Q5. Please type the most useful tag (in your opinion) from the LibraryThing tags in the User-generated description.** Here workers had to pick one of the LibraryThing user tags as the most useful, or tick the box *or tick here if there are no tags for this book* when the user-generated metadata has no tags.

There was also an optional comments field per book.

Agreement
What is the agreement among workers? We compute the pairwise agreement on relevance among workers per HIT in three different ways. The most strict agreement distinguishes between the four possible answers: 1) *Perfectly on-topic*, 2) *related but not perfect*, 3) *not the right topic* and 4) *not enough information*. In this case agreement is 0.54. If we consider only answer 1 as relevant and merge answers 2 and 3 (related means not relevant), agreement is 0.63. If we also take answer 4 to mean non-relevant (merging 2, 3 and 4, giving binary judgements), agreement is 0.68.

Recall that each HIT has at least one book that is recommended on the LT discussion thread. The average agreement between workers and forum members is 0.52. That is, on average, each worker considered 52% of the books recommended on LT as *perfectly on-topic*.

We turn the AMT relevance data from multiple workers into binary relevance judgements per book by taking the majority vote judgement. We only consider the *perfectly on-topic* category as relevant and map the other categories to non-relevant. For most books we have 3 votes, which always leads to a majority. Some books occur in multiple HITs because they are added as known relevant books from the LT forums. If there are fewer recommended books in the LT forum than there are HITs, some books have to be included in multiple HITs. Books with judgements from an even number of workers could have tied votes. In these cases we use the fact that the book was recommended on the LT topic thread as the deciding vote and label the book as relevant.

How does the relevance distribution of the AMT judgements compare to the relevance judgements from the LT discussion groups? We compare the AMT relevance judgements with the recommendations from LT in Table 7. The fiction

Table 8. Evaluation results for the official submissions using the LT relevance judgements of all 211 topics

Run	nDCG@10	P@10	MRR	MAP
p4-inex2011SB.xml_social.fb.10.50	**0.3101**	**0.2071**	**0.4811**	**0.2283**
p54-run4.all-topic-fields.reviews-split.combSUM	0.2991	0.1991	0.4731	0.1945
p4-inex2011SB.xml_social	0.2913	0.1910	0.4661	0.2115
p54-run2.all-topic-fields.all-doc-fields	0.2843	0.1910	0.4567	0.2035
p62.recommandation	0.2710	0.1900	0.4250	0.1770
p62.sdm-reviews-combine	0.2618	0.1749	0.4361	0.1755
p18.UPF_QE_group_BTT02	0.1531	0.0995	0.2478	0.1223
p18.UPF_QE_genregroup_BTT02	0.1327	0.0934	0.2283	0.1001

topics have more LT recommendations than the non-fiction, but fewer relevant books according to the AMT workers. This might be a sign that, without having read the book, judging the relevance of fiction books is harder than that of non-fiction books. For fiction there is often more to the utility of a book (whether it is interesting and/or fun) than the subject and genre information provided by book metadata. Or perhaps the relevance of fiction books is not harder to judge, but fiction is less readily considered relevant. For non-fiction information needs, the subject of a book may be one of the main aspects on which the relevance of the book is based. For fiction information needs, the subject of a book might play no role in determine its relevance. Another explanation might that the judgements pools based on the official runs might be better for non-fiction topics than for fiction topics.

3.7 Evaluation

For some topics, relevance may be both trivial and complex. Consider a topic where a user asks for good historical fiction books. The suggestions from the LT members will depend on their ideas of what are good historical fiction books. From the metadata alone it is hard to make this judgement. Should all historical fiction books be considered relevant, or only the ones suggested by the LT members? Or should relevance be graded?

For now, we will use a one-dimensional relevance scale, but like to explore alternatives in the future. One way would be to distinguish between books that a user considers as interesting options to read next and the actual book or books she decides to obtain and read. This roughly corresponds to the distinction between the library objective of helping to *find or locate* relevant items and the objective of helping to *choose* which of the relevant items to access [7].

We first show the results for the 211 topics and associated relevance judgements from the LT forums in Table 8. The best SB run (nDCG@10=0.3101) was submitted by the University of Amsterdam (p4-inex2011SB.xml_social.fb.10.50), which uses pseudo relevance feedback on an index with only reviews and tags in addition with the basic title information.

Table 9. Evaluation results for the official submissions using the AMT relevance judgements

Run	nDCG@10	P@10	MRR	MAP
p62.baseline-sdm	**0.6092**	**0.5875**	0.7794	0.3896
p4-inex2011SB.xml_amazon	0.6055	0.5792	0.7940	0.3500
p62.baseline-tags-browsenode	0.6012	0.5708	0.7779	**0.3996**
p4-inex2011SB.xml_full	0.6011	0.5708	0.7798	0.3818
p54-run2.all-topic-fields.all-doc-fields	0.5415	0.4625	**0.8535**	0.3223
p54-run3.title.reviews-split.combSUM	0.5207	0.4708	0.7779	0.2515
p18.UPF_base_BTT02	0.4718	0.4750	0.6276	0.3269
p18.UPF_QE_group_BTT02	0.4546	0.4417	0.6128	0.3061

Table 10. Evaluation results for the official submissions using the LT relevance judgements for the 24 topics used in AMT

Run	nDCG@10	P@10	MRR	MAP
p4-inex2011SB.xml_social.fb.10.50	**0.3039**	**0.2120**	**0.5339**	0.1994
p54-run2.all-topic-fields.all-doc-fields	0.2977	0.1940	0.5225	**0.2113**
p4-inex2011SB.xml_social	0.2868	0.1980	0.5062	0.1873
p54-run4.all-topic-fields.reviews-split.combSUM	0.2601	0.1940	0.4758	0.1515
p62.recommandation	0.2309	0.1720	0.4126	0.1415
p62.sdm-reviews-combine	0.2080	0.1500	0.4048	0.1352
p18.UPF_QE_group_BTT02	0.1073	0.0720	0.2133	0.0850
p18.UPF_QE_genregroup_BTT02	0.0984	0.0660	0.1956	0.0743

Next we look at the results for the 24 topics select for the AMT experiment and associated relevance judgements in Table 9. The best SB run (nDCG@10=0.6092) was submitted by the University of Avignon (p62-baseline-sdm. The most striking difference with the LT forum judgements is that here the scores for all runs are much higher. There are at least three possible explanations for this. First, the AMT judgements are based on the top 10 results of all runs, meaning all top 10 results of each run is judged, whereas many top ranked documents are not covered by the LT forum judgements. Second, the AMT judgements are explicitly based on relevance, whereas the LT forum judgements are probably more like recommendations, where users only suggest the best books on a topic and often only books they know about or have read.

A more important point is that the two evaluations are based on different topic sets. The LT forum evaluation is based on 211 topics, while the AMT evaluation is based on a subset of 24 topics.

We can see the impact of the last explanation by using the LT forum judgements only on the subset of 24 topics selected for the AMT experiment. The results for this are shown in Table 8. It seems that the topic set has little impact, as the results for the subset of 24 topics are very similar to the results for the 211 topics. This is a first indication that the LT forum test collection is

robust with respect to topic selection. It also suggests that the LT forum and AMT judgements reflect different tasks. The latter is the more traditional topical relevance task, while the former is closer to recommendation. We are still in the process of analysing the rest of the AMT data to establish to what extent the LT forum suggestions reflects relevance and recommendation tasks.

3.8 Discussion

Relevance or recommendation?
Readers may not only based their judgement on the topical relevance—is this book a historical fiction book—but also on their personal taste. Reading a book is often not just about relevant content, but about interesting, fun or engaging content. Relevance in book search might require different dimensions of graded judgements. The topical dimension (how topically relevant is this book?) is separate from the interestingness dimension (how interesting/engaging is this book?) Many topic creators ask for recommendations, and want others to explain their suggestions, so that they can better gauge how a book fits their taste.

Judging metadata or book content
In a realistic scenario, a user judges the relevance or interestingness of the book metadata, not of the content of the book. The decision to read a book comes before the judgement of the content. This points at an important problem with the suggested books collected through the touchstones. Members often suggest books they have actually read, and therefore base their suggestion on the actual content of the book. Such a relevance judgement—from someone other than the topic creator—is very different in nature from the judgement that the topic creator can make about books she has not read. Considering the suggested books as relevant brushes over this difference. We will further analyse the relevance and recommendation judgements from AMT to find out to what extent the LT forum suggestions reflect traditional topical relevance judgements and to what extent they reflect recommendation.

Extending the Collection
The Amazon/LibraryThing collection has a limited amount of professional metadata. Only 61% of the books have a DDC code and the Amazon subjects are noisy with many seemingly unrelated subject headings assign to books.

To make sure there is enough high-quality metadata from traditional library catalogues, we plan to extend the data set next year with another collection of library catalogue records from the Library of Congress and the British Library. These records contain formal metadata such as classification codes (mainly DDC and LCC) and rich subject headings based on the Library of Congress Subject Headings (LCSH).[9] Both the LoC records and the BL records are in MARCXML[10] format. We obtained MARCXML records for 1.76 million books in

[9] For more information see: http://www.loc.gov/aba/cataloging/subject/
[10] MARCXML is an XML version of the well-known MARC format. See: http://www.loc.gov/standards/marcxml/

the collection. There are 1,248,816 records from the Library of Congress and 1,158,070 records in MARC format from the British Library. Combined, there are 2,406,886 records covering 1,823,998 of the ISBNs in the Amazon/LibraryThing collection (66%). Although there is no single library catalogue that covers all books available on Amazon, we think these combined library catalogues can improve both the quality and quantity of professional book metadata.

4 The Prove It (PI) Task

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

Participants could submit up to 10 runs. Each run could contain, for each of the 83 topics (see Section 4.2), a maximum of 1,000 book pages estimated relevant to the given aspect, ordered by decreasing value of relevance.

A total of 18 runs were submitted by 2 groups (6 runs by UMass Amhers (ID=50) and 12 runs by Oslo University College (ID=100)), see Table 1.

4.1 The Digitized Book Corpus

The track builds on a collection of 50,239 out-of-copyright books[11], digitized by Microsoft. The corpus is made up of books of different genre, including history books, biographies, literary studies, religious texts and teachings, reference works, encyclopedias, essays, proceedings, novels, and poetry. 50,099 of the books also come with an associated MAchine-Readable Cataloging (MARC) record, which contains publication (author, title, etc.) and classification information. Each book in the corpus is identified by a 16 character long bookID – the name of the directory that contains the book's OCR file, e.g., A1CD363253B0F403.

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

```
<document>
 <page pageNumber="1" label="PT_CHAPTER" [coords] key="0" id="0">
  <region regionType="Text" [coords] key="0" id="0">
   <section label="SEC_BODY" key="408" id="0">
    <line [coords] key="0" id="0">
```

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

```
     <word [coords] key="0" id="0" val="Moby"/>
     <word [coords] key="1" id="1" val="Dick"/>
    </line>
    <line [...]><word [...] val="Melville"/>[...]</line>[...]
   </section>   [...]
  </region>     [...]
 </page>        [...]
</document>
```

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

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

4.2 Topics

We use the same topic set as last year [6], consisting of 83 topics. Last year, relevance judgements were collected for 21 topics from two sources. In the first phase, INEX participants judged pages using the relevance assessment system developed at Microsoft Research Cambridge.[12] In the second phase, relevance judgements were collected from Amazon Mechanical Turk.

4.3 Collected Relevance Assessments

This year, we have judgements from INEX participants for an extra 9 topics. We will use these a ground truth to bootstrap the Mechanical Turk experiments.

4.4 Evaluation Measures and Results

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

[12] URL: http://www.booksearch.org.uk

5 The Structure Extraction (SE) Task

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

In 2011, the task was run for the second time as a competition of the International Conference on Document Analysis and Recognition (ICDAR). Full details are presented in the corresponding specific competition description [4]. This year, the main novelty was the fact that the ground truth data built in 2009 and 2010 was made available online [13]. Participants were hence able to build and fine tune their systems using training data.

Participation

Following the call for participation issued in January 2011, 11 organizations registered. As in previous competitions, several participants expressed interest but renounced due to time constraints. Of the 11 organizations that signed up, 5 dropped out, that is, they neither submitted runs, nor participated in the ground truth annotation process. The list of active participants is given in Table 11. Interestingly, half of them are newcomers (Nankai University, NII Tokyo and University of Innsbrück).

Organization	Submitted runs	Ground truthing
Microsoft Development Center (Serbia)	1	y
Nankai University (PRC)	4	y
NII Tokyo (Japan)	0	y
University of Caen (France)	3	y
University of Innsbrück (Austria)	0	y
Xerox Research Centre Europe (France)	2	y

Table 11. Active participants of the Structure Extraction task.

Results

As in previous years [3], the 2011 task permitted to gather manual annotations in a collaborative fashion. The efforts of the 2011 round gave way to the gathering and addition of 513 new annotated book ToCs to the previous 527.

RunID	Participant	Title-based [3]	Link-based [2]
MDCS	MDCS	40.75%	65.1%
Nankai-run1	Nankai U.	33.06%	63.2%
Nankai-run4	Nankai U.	33.06%	63.2%
Nankai-run2	Nankai U.	32.46%	59.8%
Nankai-run3	Nankai U.	32.43%	59.8%
XRCE-run1	XRCE	20.38%	57.6%
XRCE-run2	XRCE	18.07%	58.1%
GREYC-run2	University of Caen	8.99%	50.7%
GREYC-run1	University of Caen	8.03%	50.7%
GREYC-run3	University of Caen	3.30%	24.4%

Table 12. Summary of performance scores for the Structure Extraction competition 2011 (F-measures).

A summary of the performance of all the submitted runs is given in Table 12.

The Structure Extraction task was launched in 2008 to compare automatic techniques for extracting structure information from digitized books. While the construction of hyperlinked ToCs was originally thought to be a first step on the way to the structuring of digitized books, it turns out to be a much tougher nut to crack than initially expected.

Future work aims to investigate into the usability of the extracted ToCs. In particular we wish to use qualitative measures in addition to the current precision/recall evaluation. The vast effort that this requires suggests that this can hardly be done without crowdsourcing. We shall naturally do this by building on the experience of the Book Search tasks described earlier in this paper.

6 The Active Reading Task (ART)

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

[13] http://users.info.unicaen.fr/~doucet/StructureExtraction/training/

The ART is based on the evaluation experience of EBONI [8], and adopts its evaluation framework with the aim to guide participants in organising and running user studies whose results could then be compared.

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

6.1 Task Setup

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

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

7 Conclusions and plans

This was the first year for The Social Search for Best Books (SB) task, but the amount of activity and the results promise a bright future for this task. We are in the process of analysing the data from the Amazon Mechanical Turk experiment. The topic set and relevance and recommendation judgements should give us enough data to answer questions about the relative value of professional controlled metadata and user-generated content for book search, for subject search topics as well as more recommendation oriented topics.

This year the Prove It task continued unchanged with respect to last year. The number of participants for the PI task was low and relevance judgements based on the top-k pools still need to be collected.

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

The ART was offered as last year. The task has so far only attracted 2 groups, none of whom submitted any results at the time of writing.

Bibliography

[1] Thomas Beckers, Norbert Fuhr, Nils Pharo, Ragnar Nordlie, and Khairun Nisa Fachry. Overview and results of the inex 2009 interactive track. In Mounia Lalmas, Joemon M. Jose, Andreas Rauber, Fabrizio Sebastiani, and Ingo Frommholz, editors, *ECDL*, volume 6273 of *Lecture Notes in Computer Science*, pages 409–412. Springer, 2010.

[2] Hervé Déjean and Jean-Luc Meunier. Reflections on the inex structure extraction competition. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS '10, pages 301–308, New York, NY, USA, 2010. ACM.

[3] Antoine Doucet, Gabriella Kazai, Bodin Dresevic, Aleksandar Uzelac, B.Radakovic, and Nikola Todic. Setting up a competition framework for the evaluation of structure extraction from ocr-ed books. *International Journal of Document Analysis and Recognition (IJDAR), Special Issue on Performance Evaluation of Document Analysis and Recognition Algorithms.*, 14(1):45–52, 2011.

[4] Antoine Doucet, Gabriella Kazai, and Jean-Luc Meunier. ICDAR 2011 Book Structure Extraction Competition. In *Proceedings of the Eleventh International Conference on Document Analysis and Recognition (ICDAR'2011)*, pages 1501–1505, Beijing, China, September 2011.

[5] Gabriella Kazai, Jaap Kamps, Marijn Koolen, and Natasa Milic-Frayling. Crowdsourcing for book search evaluation: Impact of hit design on comparative system ranking. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 205–214. ACM Press, New York NY, 2011.

[6] Gabriella Kazai, Marijn Koolen, Jaap Kamps, Antoine Doucet, and Monica Landoni. Overview of the INEX 2010 book track: Scaling up the evaluation using crowdsourcing. In Shlomo Geva, Jaap Kamps, Ralf Schenkel, and Andrew Trotman, editors, *Comparative Evaluation of Focused Retrieval : 9th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2010)*, volume 6932 of *LNCS*, pages 101–120. Springer, 2011.

[7] Elaine Svenonius. *The Intellectual Foundation of Information Organization.* MIT Press, 2000.

[8] Ruth Wilson, Monica Landoni, and Forbes Gibb. The web experiments in electronic textbook design. *Journal of Documentation*, 59(4):454–477, 2003.

University of Amsterdam at INEX 2011: Book and Data Centric Tracks

Frans Adriaans[1,2] Jaap Kamps[1,3] and Marijn Koolen[1]

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

Abstract. In this paper we describe our participation in INEX 2011 in the Book Track and the Data Centric Track. For the Book Track we focus on the impact of different document representations of book metadata for book search, using either professional metadata, user-generated content or both. We evaluate the retrieval results against ground truths derived from the recommendations in the LibraryThing discussion groups and from relevance judgements obtained from Amazon Mechanical Turk. Our findings show that standard retrieval models perform better on user-generated metadata than on professional metadata. For the Data Centric Track we focus on the selection of a restricted set of facets and facet values that would optimally guide the user toward relevant information in the Internet Movie Database (IMDb). We explore different methods for effective result summarization by means of weighted aggregation. These weighted aggregations are used to achieve maximal coverage of search results, while at the same time penalizing overlap between sets of documents that are summarized by different facet values. We expect that weighted result aggregation combined with redundancy avoidance results in a compact summary of available relevant information.

1 Introduction

Our aims for the Book Track were to look at the relative value of user tags and reviews and traditional book metadata for ranking book search results. The Social Search for Best Books task is newly introduced this year and uses a large catalogue of book descriptions from Amazon and LibraryThing. The descriptions are a mix of traditional metadata provided by professional cataloguers and indexers and user-generated content in the form of ratings, reviews and tags.

Because both the task and collection are new, we keep our approach simple and mainly focus on a comparison of different document representations. We made separate indexes for representations containing a) only title information, b) all the professional metadata, c) the user-generated metadata, d) the metadata from Amazon, e) the data from LibraryThing and f) all metadata. With these indexes we compare standard language model retrieval systems and evaluate them using the relevance judgements from the LibraryThing discussion forums and from Amazon Mechanical Turk. We break down the results to look

at performance on different topic types and genres to find out which metadata is effective for particular categories of topics.

For the Data Centric Track we focus on the selection of a restricted set of facets and facet values that would optimally guide the user toward relevant information. We aim to improve faceted search by addressing two issues: *weighted result aggregation* and *redundancy avoidance*.

The traditional approach to faceted search is to simply count the number of documents that is associated with each facet value. Those facet values that have the highest number of counts are returned to the user. In addition to implementing this simple approach, we explore the aggregation of results using weighted document counts. The underlying intuition is that facet values with the *most* documents are not necessarily the *most relevant* values [1]. That is, buying a dvd by the director who directed the *most* movies does not necessarily meet the search demands of a user. It may be more suitable to return directors who made a large number of *important* (and/or popular) movies. More sophisticated result aggregations, acknowledging the importance of an entity, may thus provide better hints for further faceted navigation than simple document counts. We therefore explore different methods for effective result summarization by means of weighted aggregation.

Another problem in faceted search concerns the avoidance of overlapping facets [2]. That is, facets whose values describe highly similar set of documents should be avoided. We therefore aim at penalizing overlap between sets of documents that are summarized by different facet values. We expect that weighted result aggregation combined with redundancy avoidance results in a compact summary of the available relevant information.

We describe our experiments and results for the Book Track in Section 2 and for the Data Centric Track in Section 3. In Section 4, we discuss our findings and draw preliminary conclusions.

2 Book Track

In the INEX 2011 Book Track we participated in the Social Search for Best Books task. Our aim was to investigate the relative importance of professional and user-generated metadata. The document collection consists of 2.8 million book description, with each description combining information from Amazon and LibraryThing. The Amazon data has both traditional book metadata such as title information, subject headings and classification numbers, and user-generated metadata as well as user ratings and reviews. The data from LibraryThing consists mainly of user tags.

Professional cataloguers and indexers aim to keep metadata mostly objective. Although subject analysis to determine headings and classification codes is somewhat subjective, the process follows a formal procedure and makes use of controlled vocabularies. Readers looking for interesting or fun books to read may not only want objective metadata to determine what book to read or buy next, but also opinionated information such as reviews and ratings. Moreover, subject

headings and classification codes might give a very limited view of what a book is about. LibraryThing users tag books with whatever keywords they want, including personal tags like *unread* or *living room bookcase*, but also highly specific, descriptive tags such *WWII pacific theatre* or *natives in Oklahoma*.

We want to investigate to what extent professional and user-generated metadata provide effective indexing terms for book retrieval.

2.1 Experimental Setup

We used Indri [3] for indexing, removed stopwords and stemmed terms using the Krovetz stemmer. We made 5 separate indexes:

Full : the whole description is indexed.
Amazon : only the elements derived from the Amazon data are indexed.
LT : only the elements derived from the LibraryThing data are indexed.
Title : only the title information fields (title, author, publisher, publication date, dimensions, weight, number of pages) are indexed.
Official : only the traditional metadata fields from Amazon are indexed, including the title information (see Title index) and classification and subject heading information.
Social : only the user-generated content such as reviews, tags and ratings are indexed.

In the *Full*, *LT* and *Social* indexes, the field information from `<tag>` elements is also indexed in a separate column to be able to give more weight to terms occurring in `<tag>` elements.

The topics are taken from the LibraryThing discussion groups and contain a *title* field which contains the title of a topic thread, a *group* field which contains the discussion group name and a *narrative* field which contains the first message from the topic thread.

In our experiments we only used the *title* fields as queries and default settings for Indri (Dirichlet smoothing with $\mu = 2500$). We submitted the following six runs:

xml_amazon : a standard LM run on the Amazon index.
xml_full : a standard LM run on the Full index.
xml_full.fb.10.50 : a run on the Full index with pseudo relevance feedback using 50 terms from the top 10 results.
xml_lt : a standard LM run on the LT index.
xml_social : a standard LM run on the Social index.
xml_social.fb.10.50 : a run on the Social index with pseudo relevance feedback using 50 terms from the top 10 results.

Additionally we created the folowing runs:

xml_amazon.fb.10.50 : a standard LM run on the Amazon index.
xml_lt.fb.10.50 : a standard LM run on the LT index.
xml_official : a standard LM run on the Official index.
xml_title : a standard LM run on the Title index.

Table 1: Evaluation results for the Book Track runs using the LT recommendation Qrels. Runs marked with * are official submissions.

Run	nDCG@10	P@10	MRR	MAP
xml_amazon.fb.10.50	0.2665	0.1730	0.4171	0.1901
*xml_amazon	0.2411	0.1536	0.3939	0.1722
*xml_full.fb.10.50	0.2853	0.1858	0.4453	0.2051
*xml_full	0.2523	0.1649	0.4062	0.1825
xml_lt.fb.10.50	0.1837	0.1237	0.2940	0.1391
*xml_lt	0.1592	0.1052	0.2695	0.1199
xml_prof	0.0720	0.0502	0.1301	0.0567
*xml_social.fb.10.50	**0.3101**	**0.2071**	**0.4811**	**0.2283**
*xml_social	0.2913	0.1910	0.4661	0.2115
xml_title	0.0617	0.0403	0.1146	0.0563

Table 2: Evaluation results for the Book Track runs using the AMT Qrels. Runs marked with * are official submissions.

Run	nDCG@10	P@10	MRR	MAP
xml_amazon.fb.10.50	0.5954	0.5583	0.7868	0.3600
*xml_amazon	**0.6055**	**0.5792**	0.7940	0.3500
*xml_full.fb.10.50	0.5929	0.5500	**0.8075**	**0.3898**
*xml_full	0.6011	0.5708	0.7798	0.3818
xml_lt.fb.10.50	0.4281	0.3792	0.7157	0.2368
*xml_lt	0.3949	0.3583	0.6495	0.2199
xml_prof	0.1625	0.1375	0.3668	0.0923
*xml_social.fb.10.50	0.5425	0.5042	0.7210	0.3261
*xml_social	0.5464	0.5167	0.7031	0.3486
xml_title	0.2003	0.1875	0.3902	0.1070

2.2 Results

The Social Search for Best Books task has two sets of relevance judgements. One based on the lists of books that were recommended on the LT discussion groups, and one based on document pools of the top 10 results of all official runs, judged by Mechanical Turk workers. For the latter set of judgements, a subset of 24 topics was selected from the larger set of 211 topics from the LT forums.

We first look at the evaluation results based on the Qrels derived from the LT discussion groups in Table 1. The runs on the Social index outperform the other runs on all measures. The indexes with no user-generated metadata–Official and Title—lead to low scoring runs. Feedback is effective on the four indexes Amazon, Full, LT and Social.

Next we look at the results based on the Mechanical Turk judgements over the subset of 24 topics in Table 2. Here we see a different pattern. With the top 10 results judged on relevance, all scores are higher than with the LT judgements. This is probably due in part to the larger number of judged documents, but perhaps also to the difference in the tasks. The Mechanical Turk workers were

Table 3: Evaluation results for the Book Track runs using the LT recommendation Qrels for the 24 topics selected for the AMT experiment. Runs marked with * are official submissions.

Run	nDCG@10	P@10	MRR	MAP
xml_amazon.fb.10.50	0.2103	0.1625	0.3791	0.1445
xml_amazon	0.1941	0.1583	0.3583	0.1310
xml_full.fb.10.50	0.2155	0.1708	0.3962	0.1471
xml_full	0.1998	0.1625	0.3550	0.1258
xml_lt.fb.10.50	0.1190	0.0833	0.3119	0.0783
xml_lt	0.1149	0.0708	0.3046	0.0694
xml_prof	0.0649	0.0500	0.1408	0.0373
xml_social.fb.10.50	**0.3112**	**0.2333**	**0.5396**	**0.1998**
xml_social	0.2875	0.2083	0.5010	0.1824
xml_title	0.0264	0.0167	0.0632	0.0321

asked to judged the topical relevance of books—is the book on the same topic as the request from the LT forum—whereas the LT forum members were asked by the requester to recommend books from a possibly long list of topically relevant books. Another interesting observation is that feedback is not effective for the AMT evaluation, whereas it was effective for the LT evaluation.

Perhaps another reason is that the two evaluations use different topic sets. To investigate the impact of the topic set, we filtered the LT judgements on the 24 topics selected for AMT, such that the LT and AMT judgements are more directly comparable. The results are shown in Table 3. The pattern is similar to that of the LT judgements over the 211 topics, indicating that the impact of the topic set is small. The runs on the Social index outperform the others, with the Amazon and Full runs scoring better than the LT runs, which in turn perform better than the Official and Title runs. Feedback is again effective for all reported measures. In other words, the observed difference between the LT and AMT evaluations is not caused by difference in topics but probably caused by the difference in the tasks.

2.3 Analysis

The topics of the SB Track are labelled with topic type and genre. There are 8 different type labels: *subject* (134 topics), *author* (32), *genre* (17), *series* (10), *known-item* (7), *edition* (7), *work* (3) and *language* (2).

We break down the evaluation results over topic types and take a closer look at the *subject*, *author* and *genre* types.

The other types have either very small numbers of topics (*work* and *language*), or are hard to evaluate with the current relevance judgements. For instance, the *edition* topics ask for a recommended edition of a particular work. In the relevance judgements the multiple editions of a work are all mapped to a single work ID in LibraryThing. Some books have many more editions than others, which would create in imbalance in the relevance judgements for most topics.

Table 4: Evaluation results using the LT recommendation Qrels across different topic genres and types. Runs marked with * are official submissions.

	nDCG@10				
Run	**Fiction**	**Non-fiction**	**Subject**	**Author**	**Genre**
xml_amazon.fb.10.50	0.2739	0.2608	0.2203	0.4193	0.0888
*xml_amazon	0.2444	0.2386	0.1988	0.3630	0.0679
*xml_full.fb.10.50	0.2978	0.2765	0.2374	0.4215	0.1163
*xml_full	0.2565	0.2491	0.2093	0.3700	0.0795
xml_lt.fb.10.50	0.1901	0.1888	0.1597	0.2439	0.0850
*xml_lt	0.1535	0.1708	0.1411	0.2093	0.0762
xml_prof	0.0858	0.0597	0.0426	0.1634	0.0225
*xml_social.fb.10.50	**0.3469**	**0.2896**	**0.2644**	**0.4645**	0.1466
*xml_social	0.3157	0.2783	0.2575	0.4006	**0.1556**
xml_title	0.0552	0.0631	0.0375	0.1009	0.0000

The genre labels can be grouped into fiction, with genre label *Literature* (89 topics) and non-fiction, with genre labels such as *history* (60 topics), *biography* (24), *military* (16), *religion* (16), *technology* (14) and *science* (11).

The evaluation results are shown in Table 4.

For most runs there is no big difference in performance between *fiction* and *non-fiction* topics, with slightly better performance on the *fiction* topics. For the two runs on the Social index the difference is bigger. Perhaps this is due to a larger amount of social metadata for fiction books. The standard run on the LT index (xml_lt) performs better on the non-fiction topics, suggesting the tags for non-fiction are more useful than for fiction books.

Among the topic types we see the same pattern across all measures and all runs. The *author* topic are easier than the *subject* topics, which are again easier than the *genre* topics. We think this is a direct reflection of the clarity and specificity of the information needs and queries. For author related topics, the name of the author is a very clear and specific retrieval cue. Subject are somewhat broader and less clearly defined, making it harder to retrieve exactly the right set of books. For genre-related topics it is even more difficult. Genres are broad and even less clearly defined. For many genres there are literally (tens of) thousands of books and library catalogues rarely go so far in classifying and indexing specific genres. This is also reflected by the very low scores of the Official and Title index runs for *genre* topics.

3 Data Centric Track

For the Data Centric Track we participated in the Ad Hoc Task and the Faceted Search Task. Our particular focus was on the Faceted Seach Task where we aim to discover for each query a restricted set of facets and facet values that best describe relevant information in the results list. Our general approach is to use weighted result aggregations to achieve maximal coverage of relevant documents in IMDb. At the same time we aim to penalize overlap between sets of documents

that are summarized by different facet values. We expect that weighted result aggregation combined with redundancy avoidance results in a compact summary of the available relevant information. Below we describe our setup and provide details about the different runs that we submitted to INEX. At the time of writing no results are available for the Faceted Search Task. We are therefore unable to provide an analysis of the performance of our different Faceted Search runs at this point in time.

3.1 Experimental Setup

In all ad hoc runs (Ad Hoc and Faceted Search) we use Indri [3] with Krovetz stemming and default smoothing (Dirichlet with $\mu = 2500$) to create an index. All XML leaf elements in the IMDb collection are indexed as fields. The XML document structure was not used for indexing. Documents were retrieved using title fields only. The maximum number of retrieved documents was set to 1000 (Ad Hoc Task) and 2000 (Faceted Search Task). We submitted one run for the Ad Hoc Search Task and three runs for the Faceted Search Task.

Ad Hoc Task Since the Ad Hoc Search Task was not the focus of our participation, only one run (*UAms2011adhoc*) was generated for the Ad Hoc topic set, using the settings described above.

Faceted Search Task Three runs were generated for the Faceted Search Task (*UAms2011indri-c-cnt*, *UAms2011indri-cNO-scr2*, *UAms2011lucene-cNO-lth*). In each run, a hierarchy of recommended facet values is constructed for each topic. Every path through the hierarchy represents an accumulated set of conditions on the retrieved documents. The search results become more refined at every step, and the refinement ultimately narrows down a set of potentially interesting documents. Below we describe our approach to faceted search in more detail.

3.2 Step 1: Ad hoc run on IMDb collection

Two ad hoc result files were used: the *2011-dc-lucene.trec* file provided by the INEX organization, and an ad hoc run that was created on the fly using Indri. The maximum number of results was set to 2000.

3.3 Step 2: Facet selection

The candidate set consists of all numerical and categorical fields in the IMDb collection. (Free-text fields were not allowed as candidate facets by the organization.) The goal was to select useful facets (and values) from the set of candidate facets.

Result aggregation We explored two different methods of weighted result aggregation. The first method is aggregation using document lengths rather than document counts. Since popular movies in IMDb have larger entries (which we measure by file size), we reasoned that summing over document lengths may help in getting popular facet values (associated with popular movies) at the top of the ranked set of facet values. The second method is aggregation using retrieval scores. That is, we sum the retrieval scores of each document taken from the ad hoc run. The idea is that higher-ranked documents in the results file display those facet values that are most likely to be of interest to the user, given the user's query. The difference between the two weighted aggregation methods is that document length is a static ('global') measure of document importance, whereas retrieval scores are dynamic ('local'), resulting in different degrees of importance for different topics. We compare both methods to traditional non-weighted aggregation of search results. The result aggregations form the basis of facet selection, which is described below.

Coverage The idea behind our approach to facet selection is the simple intuition that facets which provide compact summaries of the available data would allow for fast navigation through the collection. This intuition was implemented as *facet coverage*: the number of documents that are summarized by a facet's top n values[1]. Two types of coverage were implemented. The first version, *coverage*, simply sums up the (weighted) document counts that are associated with the facet's top n values. A potential pitfall of this approach, however, is that this method favors redundancy. That is, the sets of documents that are associated with different facet values may have a high degree of overlap. For example, the keywords 'murder' and 'homicide' may point to almost identical sets of documents. Since we want to give the user compact overviews of different, non-overlapping sets of documents that may be of interest to the searcher, we implemented a second version: *coverageNO* ('coverage, no overlap'). Rather than summing up document counts, *coverageNO* counts the number of unique documents that are summarized by the facet's top n values. This way redundancy in facet values is penalized.

Coverage-based facet selection is applied recursively. Starting with the complete set of ad hoc results (corresponding to the root node of the facet hierarchy), the facet with the highest coverage is chosen. The set of results is then narrowed down to the set of documents that are covered by this facet. With this new set, a second facet is chosen with the highest coverage within the new set. This selection process continues until a specified number of facets has been selected.[2] We apply facet selection to movie candidate facets and person candidate facets independently, since these facets describe different types of documents (i.e., you cannot drill-down into person files after you've narrowed down the results using a movie facet). An example of a ranked set of movie facets for the query 'Vietnam' is given in Table 5.

[1] In our runs, we explored $n = 5$ and $n = 10$.
[2] We set the maximum number of selected facets to 5.

Table 5: Facets ranked by coverage (based on document counts).

Rank	**Coverage**	**Facet**	**Top-5 values**
1	945	genre	Drama (306) Documentary (207) War (199) Action (157) Comedy (76)
2	850	keyword	vietnam (286) vietnam-war (220) independent-film (162) vietnam-veteran (110) 1960s (72)
3	477	language	English (400) Vietnamese (42) French (16) Spanish (10) German (9)
4	437	country	USA (345) UK (30) Canada (27) France (19) Vietnam (16)
5	397	color	Color (291) Color - (Technicolor) (45) Black and White (40) Color - (Eastmancolor) (11) Color - (Metrocolor) (10)

3.4 Step 3: Path construction

The selected set of facets with corresponding top n ranked values form the basis of the facet hierarchy. Paths in the hierarchy are generated by selecting a value for the first facet, then a value for the second facet, etc. The paths respect the rankings of the values along the path. That is, paths through high-ranked facet values are listed at the top of the hierarchy, followed by paths through lower-ranked facet values. In order to restrict the number of paths in the hierarchy (not all logically possible paths are considered relevant) we return only paths that we think are useful recommendations for the user, using a formal criterium. In our current implementation, only paths are included which lead to a set of documents of a specified size.[3] Paths that lead to fewer documents (e.g., < 10 documents) are ignored because they are too specific. Conversely, paths that lead to a larger number of documents (e.g., > 20 documents) are considered too general, and the system will attempt to branch into a deeper, more specific level.

We generate trees for 'movies' and 'persons' independently and join them in the order of the highest number of paths. (For most queries there were more movie paths than person paths.) An example of our approach to constructing paths through facet values is shown below. We display the tree corresponding to the 'Vietnam' query, using the facets from Table 5:

```
<topic tid="2011205">
<fv f="/movie/overview/genres/genre" v="Drama">
  <fv f="/movie/overview/keywords/keyword" v="vietnam">
    <fv f="/movie/additional_details/languages/language" v="Vietnamese">
      <fv f="/movie/additional_details/countries/country" v="USA">
        <fv f="/movie/additional_details/colors/color" v="Color"/>
          </fv>
        </fv>
      </fv>
  <fv f="/movie/overview/keywords/keyword" v="vietnam-war">
    <fv f="/movie/additional_details/languages/language" v="English">
      <fv f="/movie/additional_details/countries/country" v="USA">
        <fv f="/movie/additional_details/colors/color" v="Color - (Technicolor)"/>
          </fv>
        </fv>
    <fv f="/movie/additional_details/languages/language" v="Vietnamese">
      <fv f="/movie/additional_details/countries/country" v="USA"/>
        </fv>
      </fv>
  <fv f="/movie/overview/keywords/keyword" v="vietnam-veteran">
    <fv f="/movie/additional_details/languages/language" v="English">
      <fv f="/movie/additional_details/countries/country" v="USA">
        <fv f="/movie/additional_details/colors/color" v="Color - (Technicolor)"/>
```

[3] As an inclusion criterium, we keep all paths that lead to a set of 10-20 documents.

```
            </fv>
          </fv>
        </fv>
      </fv>
<fv f="/movie/overview/genres/genre" v="Documentary">
  <fv f="/movie/overview/keywords/keyword" v="vietnam">
    <fv f="/movie/additional_details/languages/language" v="English">
      <fv f="/movie/additional_details/countries/country" v="USA">
        <fv f="/movie/additional_details/colors/color" v="Black and White"/>
          </fv>
        </fv>
      </fv>
  <fv f="/movie/overview/keywords/keyword" v="vietnam-war">
    <fv f="/movie/additional_details/languages/language" v="English">
      <fv f="/movie/additional_details/countries/country" v="USA">
        <fv f="/movie/additional_details/colors/color" v="Black and White"/>
          </fv>
        </fv>
      </fv>
  <fv f="/movie/overview/keywords/keyword" v="vietnam-veteran">
    <fv f="/movie/additional_details/languages/language" v="English">
      <fv f="/movie/additional_details/countries/country" v="USA">
        <fv f="/movie/additional_details/colors/color" v="Color"/>
          </fv>
        </fv>
      </fv>
  <fv f="/movie/overview/keywords/keyword" v="1960s">
    <fv f="/movie/additional_details/languages/language" v="English">
      <fv f="/movie/additional_details/countries/country" v="USA">
        <fv f="/movie/additional_details/colors/color" v="Color"/>
          </fv>
        </fv>
...
```

3.5 The Faceted Seach runs

With the methodology described above, a total of 32 runs was generated by varying the parameters listed in Table 6. From this set the following three runs were selected for submission:

UAms2011indri-c-cnt This is our baseline run which implements the standard approach of selecting those facet values that summarize the largest number of documents.

UAms2011indri-cNO-scr2 : This run uses weighted result aggregation (using retrieval scores, in contrast to the unranked aggregation in the baseline run). In addition, this run penalizes overlap between document sets that correspond to different facet values.

Table 6: Experimental parameters (which resulted in 2x4x2x1x2x1x1 = 32 runs)

Parameter	Values
Ad hoc input	*lucene, indri*
Document weights	*count, length, score, score*^{2}
Selection method	*coverage, coverageNO*
Number of facets	5
Number of values	5, 10
Min. number of path results	10
Max. number of path results	20

UAms2011lucene-cNO-lth : The third run uses the Lucene run that was provided by the INEX organizers. The run uses weighted result aggregation based on document lengths (file sizes, as opposed to retrieval scores).

3.6 Results and discussion

Our run for the Ad Hoc Task ranked 1st (based on MAP scores; MAP = 0.3969). The success of our Ad Hoc run indicates that indexing the complete XML structure of IMDb is not necessary for effective document retrieval. It appears, at least for the Ad Hoc case, that it suffices to index leaf elements. Results of the Faceted Search Task are unknown at this time.

4 Conclusion

In this paper we discussed our participation in the INEX 2011 Book and Data Centric Tracks.

In the Book Track we participated in the Social Search for Best Books task and focused on comparing different document representations based on professional metadata and user-generated metadata. Our main finding is that standard language models perform better on representations of user-generated metadata than on representations of professional metadata.

In our result analysis we differentiated between topics requesting fiction and non-fiction books and between subject-related topics, author-related topics and genre-related topics. Although the patterns are similar across topic types and genres, we found that social metadata is more effective for fiction topics than for non-fiction topics, and that regardless of document representation, all systems perform better on author-related topics than on subject related topics and worst on genre-related topics. We expect this is related to the specificity and clarity of these topic types. Author-related topics are highly specific and target a clearly defined set of books. Subject-related topics are broader and less clearly defined, but can still be specific. Genre-related topics are very broad—many genres have tens of thousands of books—and are also more vague information needs that are closer to exploratory search.

In future work we will look closer at the relative value of various types of metadata and directly compare individual types of metadata such as reviews,

tags and subject headings. We will also look at the different search scenarios underlying the relevance judgements and topic categories, such as subject search, recommendation and exploratory search.

In the Data Centric Track we participated in the Ad Hoc and Faceted Search Task. While our Ad Hoc approach worked fairly well (as demonstrated by the high MAP), the results of the Faceted Search Task are not yet available. Our expectation is that weighted result aggregation will improve faceted search, since it acknowledges either the global or local importance of different documents in the results list. In addition, we expect that redundancy avoidance will lead to a more compact representation of the results list.

Acknowledgments Frans Adriaans was supported by the Netherlands Organization for Scientific Research (NWO) grants # 612.066.513 and 446.010.027. Jaap Kamps was supported by NWO under grants # 612.066.513, 639.072.601, and 640.005.001. Marijn Koolen was supported by NWO under grant # 639.072.601.

Bibliography

[1] O. Ben-Yitzhak, N. Golbandi, N. Har'El, and R. Lempel. Beyond basic faceted search. In *WSDM'08*, 2008.

[2] C. Li, N. Yan, S. B. Roy, L. Lisham, and G. Das. Facetedpedia: Dynamic generation of query-dependent faceted interfaces for wikipedia. In *Proceedings of WWW 2010*, 2010.

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

RSLIS at INEX 2011: Social Book Search Track

Toine Bogers[1], Kirstine Wilfred Christensen[2], and Birger Larsen[1]

[1] Royal School of Library and Information Science
Birketinget 6, 2300 Copenhagen, Denmark
{tb,blar}@iva.dk
[2] DBC, Tempovej 7, 2750 Ballerup, Denmark
{kwc}@dbc.dk

Abstract. In this paper, we describe our participation in the INEX 2011 Social Book Search track. We investigate the contribution of different types of document metadata, both social and controlled, and examine the effectiveness of re-ranking retrieval results using social features. We find that the best results are obtained using all available document fields and topic representations.

Keywords: XML retrieval, social tagging, controlled metadata, book recommendation

1 Introduction

In this paper, we describe our participation in the INEX 2011 Social Book Search track. Our goals for the Social Book Search task were (1) to investigate the contribution of different types of document metadata, both social and controlled; and (2) to examine the effectiveness of using social features to re-rank the initial content-based search results.

The structure of this paper is as follows. We start in Section 2 by describing our methodology: pre-processing the data, which document and topic fields we used for retrieval, and our evaluation. In Section 3, we describe the results of our content-based retrieval runs. Section 4 describes our use of social features to re-rank the content-based search results. Section 5 describes which runs we submitted to INEX, with the results of those runs presented in Section 6. We discuss our results and conclude in Section 7.

2 Methodology

2.1 Data and Preprocessing

In our experiments we used the Amazon/LibraryThing collection provided by the organizers of the INEX 2011 Social Book Search track. This collection contains XML representations of 2.8 million books, with the book representation data crawled from both Amazon.com and LibraryThing.

A manual inspection of the collection revealed the presence of several XML fields that are unlikely to contribute to the successful retrieval of relevant books. Examples include XML fields like <image>, <listprice>, and <binding>. While it is certainly not impossible that a user would be interested only in books in a certain price range or in certain bindings, we did not expect this to be likely in this track's particular retrieval scenario of recommending books based on a topical request. We therefore manually identified 22 such fields and removed them from the book representations.

In addition, we converted the original XML schema into a simplified version. After these pre-processing steps, we were left with the following 19 content-bearing XML fields in our collection: <isbn>, <title>, <publisher>, <editorial>[3], <creator>[4], <series>, <award>, <character>, <place>, <blurber>, <epigraph>, <firstwords>, <lastwords>, <quotation>, <dewey>, <subject>, <browseNode>, <review>, and <tag>.

One of the original fields (<dewey>) contains the numeric code representing the Dewey Decimal System category that was assigned to a book. We replaced these numeric Dewey codes by their proper textual descriptions using the 2003 list of Dewey category descriptions[5] to enrich the controlled metadata assigned to each book. For example, the XML element <dewey>519</dewey> was replaced by the element <dewey>Probabilities & applied mathematics</dewey>.

2.2 Field categories and Indexing

The 19 remaining XML fields in our collection's book representations fall into different categories. Some fields, such as <dewey> and <subject>, are examples of *controlled metadata* produced by LIS professionals, whereas other fields contains *user-generated metadata*, such as <review> and <tag>. Yet other fields contain 'regular' book metadata, such as <title> and <publisher>. Fields such as <quotation> and <firstwords> represent a book's content more directly.

To examine the influence of these different types of fields, we divided the document fields into five different categories, each corresponding to an index. In addition, we combined all five groups of relevant fields for an index containing all fields. This resulted in the following six indexes:

All fields For our first index all-doc-fields we simply indexed all of the available XML fields (see the previous section for a complete list).

Metadata In our metadata index, we include all metadata fields that are immutably tied to the book itself and supplied by the publisher: <title>, <publisher>, <editorial>, <creator>, <series>, <award>, <character>, and <place>.

[3] Our <editorial> fields contain a concatenation of the original <source> and <content> fields for each editorial review.

[4] For our <creator> field, we disregard the different roles the creators could have in the original XML schema and simply treat all roles the same.

[5] Available at http://www.library.illinois.edu/ugl/about/dewey.html

Content For lack of access to the actual full-text books, we grouped together all XML fields in the content index that contain some part of the book text: blurbs, epigraphs, the first and last words, and quotations. This corresponded to indexing the fields <blurber>, <epigraph>, <firstwords>, <lastwords>, and <quotation>.

Controlled metadata In our controlled-metadata index, we include the three controlled metadata fields curated by library professionals: <browseNode>, <dewey>, and <subject>.

Tags We split the social metadata contained in the document collection into two different types: tags and reviews. For the tags index, we used the tag field, expanding the tag count listed in the original XML. For example, the original XML element <tag count="3">fantasy</tag> would be expanded as <tag>fantasy fantasy fantasy</tag>. This ensures that the most popular tags have a bigger influence on the final query-document matching.

Reviews The user reviews from the <review> fields were indexed in two different ways: (1) all user reviews belonging to a single book were combined in a single document representation for that book, and (2) each book review was indexed and retrieved separately. The former book-centric review index reviews is used in Section 3; the latter review-centric index reviews-split is used in our social re-ranking approach described in Section 4.

We used the Indri 5.0 retrieval toolkit[6] for indexing and retrieval. We performed stopword filtering on all of our indexes using the SMART stopword list, and preliminary experiments showed that using the Krovetz stemmer resulted in the best performance. Topic representations were processed in the same manner.

2.3 Topics

As part of the INEX 2011 Social Book Search track two set of topics were released with requests for book recommendations based on textual description of the user's information need: a training set and a test set. Both topic sets were extracted from the LibraryThing forum. The training set consisted of 43 topics and also contained relevance judgments, which were crawled from the LibraryThing forum messages. These relevance judgments were provided to the participants unfiltered and possibly incomplete. Despite these known limitations, we used the training set to optimize our retrieval algorithms in the different runs. The results we report in Sections 3 and 4 were obtained using this training set.

The test set containing 211 topics is the topic set used to rank and compare the different participants' systems at INEX. The results listed in Section 6 were obtained on this test set.

Each topic in the two sets are represented by several different fields, with some fields only occurring in the test set. In our experiments with the training and the test set, we restricted ourselves to automatic runs using the following three fields (partly based on a manual inspection of their usefulness for retrieval):

[6] Available at http://www.lemurproject.org/

Title The <title> field contains the title of the forum topic and typically provide a concise description of the information need. Runs that only use the topic title are referred to as title.

Group The LibraryThing forum is divided into different groups covering different topics. Runs that only use the <group> field (i.e., the name of the LibraryThing group as query) are referred to as group.

Narrative The first message of each forum topic, typically posted by the topic creator, describes the information need in more detail. This often contains a description of the information need, some background information, and possibly a list of books the topic creator has already read or is not looking for. The <narrative> field typically contains the richest description of the topic and runs using only this field are referred to as narrative.

All topic fields In addition to runs using these three fields individually, we also performed runs with all three fields combined (all-topic-fields).

The test and training sets contained several other fields that we did not experiment with due to temporal constraints, such as <similar> and <dissimilar>. However, we list some of our ideas in Section **??**.

2.4 Experimental setup

In all our retrieval experiments, we used the language modeling approach with Jelinek-Mercer (JM) smoothing as implemented in the Indri 5.0 toolkit. We preferred JM smoothing over Dirichlet smoothing, because previous work has shown that for longer, more verbose queries JM smoothing performs better than Dirichlet smoothing [1], which matches the richer topic descriptions provided in the training and test sets.

For the best possible performance, we optimized the λ parameter, which controls the influence of the collection language model. We varied λ in steps of 0.1, from 0.0 to 1.0 using the training set of topics. For each topic we retrieve up 1000 documents and we used NDCG as our evaluation metric [2].

3 Content-based Retrieval

For our first round of experiments focused on a standard content-based retrieval approach where we compared the different index and the different topic representations. We had six different indexes (all-doc-fields, metadata, content, controlled-metadata, tags, and reviews) and four different sets of topic representations (title, group, narrative, and all-topic-fields). We examined each of these pairwise combinations for a total of 24 different content-based retrieval runs. Table 1 shows the best NDCG results for each run on the training set with the optimal λ values.

We can see several interesting results in Table 1. First, we see that the best overall content-based run used all topic fields for the training topics, retrieved against the index containing all document fields (all-doc-fields). In fact, for three out of four topic sets, using all-doc-fields provides the best performance. The

Table 1. Results of the 24 different content-based retrieval runs on the training set using NDCG as evaluation metric. Best-performing runs for each topic representation are printed in bold. The boxed run is the best overall.

Document fields	Topic fields			
	title	narrative	group	all-topic-fields
metadata	0.2756	0.2660	0.0531	0.3373
content	0.0083	0.0091	0.0007	0.0096
controlled-metadata	0.0663	0.0481	0.0235	0.0887
tags	0.2848	0.2106	0.0691	0.3334
reviews	**0.3020**	0.2996	0.0773	0.3748
all-doc-fields	0.2644	**0.3445**	**0.0900**	**0.4436**

book-centric reviews index is close second with strong performance on all four topic sets. Finally, we observe that the content and controlled-metadata indexes result in the worst retrieval performance across all four topic sets.

When we compare the different topic sets, we see that the all-topic-fields set consistently produces the best performance, followed by the title and narrative topic sets. The group topic set generally produced the worst-performing runs.

4 Social Re-ranking

The inclusion of user-generated metadata in the Amazon/LibraryThing collection gives the track participants the opportunity to examine the effectiveness of using social features to re-rank or improve the initial content-based search results. One such a source of social data are the tags assigned by LibraryThing users to the books in the collection. The results in the previous section showed that even when treating these as a simple content-based representation of the collection using our tags index, we can achieve relatively good performance.

In this section, we turn our attention to the book reviews entered by Amazon's large user base. We mentioned in Section 2.1 that we indexed the user reviews from the <review> fields in two different ways: (1) all user reviews belonging to a single book were combined in a single document representation for that book (reviews), and (2) each book review was indexed and retrieved separately (reviews-split). The results of the content-based runs in the previous section showed that a book-centric approach to indexing reviews provided good performance.

Review-centric retrieval However, all user reviews are not equal. Some reviewers provide more accurate, in-depth reviews than others, and in some cases reviews may be even be misleading or deceptive. This problem of spam reviews on online shopping websites such as Amazon.com is well-documented [3]. This suggests that indexing and retrieving reviews individually and then aggregating the individually retrieved reviews could be beneficial by matching the best, most topical reviews against our topics.

Our review-centric retrieval approach works as follows. First, we index all reviews separately in our reviews-split index. We then retrieve the top 1000 individual reviews for each topic (i.e., this is likely to be a mixed of different reviews for different books). This can result in several reviews covering the same book occurring in our result list, which then need to be aggregated into a single relevance score for each separate book. This problem is similar to the problem of *results fusion* in IR, where the results of *different* retrieval algorithms on the *same* collection are combined. This suggest the applicability of standard methods for results fusion as introduced by [4]. Of the six methods they investigated, we have selected the following three for aggregating the review-centric retrieval results.

- The CombMAX method takes the maximum relevance score of a document from among the different runs. In our case, this means that for each book in our results list, we take the score of the highest-retrieved individual review to be the relevance score for that book.
- The CombSUM method fuses runs by taking the sum of the relevance scores for each document separately. In our case, this means that for each book in our results list, we take the sum of the relevance scores for all reviews referring to that particular book.
- The CombMNZ method does the same as the CombSUM method, but boost thes sum of relevance scores by the number of runs that actually retrieved the document. In our case, this means that for each book in our results list, we take the sum of the relevance scores for all reviews referring to that particular book, and multiply that by the number of reviews that were retrieved for that book.

Helpfulness of reviews One of the more popular aspects of user reviewing process on Amazon.com is that reviews can be marked as helpful or not helpful by other Amazon users. By using this information, we could ensure that the most helpful reviews have a better chance of being retrieved. We can use this information to improve the retrieval results by assigning higher weights to the most helpful reviews and thereby boosting the books associated with those reviews. The assumption behind this is that helpful reviews will be more accurate and on-topic than unhelpful reviews.

We estimate the helpfulness of a review by dividing the number of votes for helpfulness by the total number of votes for that review. For example, a review that 3 out of 5 people voted as being helpful would have a helpfulness score of 0.6. For each retrieved review i we then obtain a new relevance score $score_{weighted}(i)$ by multiplying that review's original relevance score $score_{org}(i)$ with its helpfulness score as follows:

$$score_{weighted}(i) = score_{org}(i) \times \frac{helpful\ vote\ count}{total\ vote\ count} \tag{1}$$

This will results in the most helpful reviews having a bigger influence on the final rankings and the less helpful reviews having a smaller influence. We combine this weighting method with the three fusion methods CombMAX, CombSUM, and

CombMNZ to arrive at a weighted fusion approach.

Book ratings In addition, users can also assign individual ratings from zero to five stars to the book they are reviewing, suggesting an additional method of taking into account the quality of the books to be retrieved. We used these ratings to influence the relevance scores of the retrieved books. For each retrieved review i we obtain a new relevance score $score_{weighted}(i)$ by multiplying that review's original relevance score $score_{org}(i)$ with its normalized rating r as follows:

$$score_{weighted}(i) = score_{org}(i) \times \frac{r}{5} \tag{2}$$

This will results in the positive reviews having a bigger influence on the final rankings and the negative reviews having a smaller influence. An open question here is whether positive reviews are indeed a better source of book recommendations than negative reviews. We combine this weighting method with the three fusion methods CombMAX, CombSUM, and CombMNZ to arrive at a weighted fusion approach.

Table 2 shows the results of the different social ranking runs for the optimal λ values. The results of the runs using the book-centric reviews index are also included for convenience.

Table 2. Results of the 9 different social ranking runs with the reviews-split index on the training set using NDCG as evaluation metric. The results of the runs using the book-centric reviews index are also included for convenience. Best-performing runs for each topic representation are printed in bold. The boxed run is the best overall using the reviews-split index.

Runs	**Topic fields**			
	title	narrative	group	all-topic-fields
CombMAX	0.3117	**0.3222**	0.0892	0.3457
CombSUM	**0.3377**	0.3185	**0.0982**	**0.3640**
CombMNZ	0.3350	0.3193	**0.0982**	0.3462
CombMAX - Helpfulness	0.2603	0.2842	0.0722	0.3124
CombSUM - Helpfulness	0.2993	0.2957	0.0703	0.3204
CombMNZ - Helpfulness	0.3083	0.2983	0.0756	0.3203
CombMAX - Ratings	0.2882	0.2907	0.0804	0.3306
CombSUM - Ratings	0.3199	0.3091	0.0891	0.3332
CombMNZ - Ratings	0.3230	0.3080	0.0901	0.3320
reviews	0.3020	0.2996	0.0773	0.3748

What do the results of the social ranking approaches tell us? The best overall social ranking approach is the unweighted CombSUM method using all available topic fields, with a NDCG score of 0.3640. Looking at the unweighted fusion methods, we see that our results confirm the work of, among others [4] and [5],

as the CombSUM and CombMNZ fusion methods tend to perform better than CombMAX. For the weighted fusion approaches where the weights are derived from information about review helpfulness and book ratings we see the same patterns for these three methods: CombSUM and CombMNZ outperform CombMAX.

Overall, however, the unweighted fusion methods outperform the two weighted fusion methods. This is *not* in line with previous research [6,7], where the optimal combination of weighted runs tends to outperform the unweighted variants. This suggests that our weighting methods using helpfulness and ratings are not optimal. Apparently, reviews that are helpful for users are not necessarily helpful for a retrieval algorithm. Analogously, increasing the influence of positive reviews over negative reviews is not the ideal approach either. We do observe however that using weights based on book ratings have a slight edge over weights derived from review helpfulness.

Finally, if we compare the book-centric and review-centric approaches, we see a mixed picture: while the best result using the reviews-split index is not as good as the best result using the reviews index, this is only true for one of the four topic sets. For the other topic sets where the retrieval algorithm has less text to work with the review-centric approach actually comes out on top.

5 Submitted runs

We selected four automatic runs for submission to INEX[7] based on the results of our content-based and social retrieval runs. Two of these submitted runs were content-based runs, the other two were social ranking-based runs.

Run 1 title.all-doc-fields This run used the titles of the test topics[8] and ran this against the index containing all available document fields, because this index provided the best content-based results.

Run 2 all-topic-fields.all-doc-fields This run used all three topic fields combined and ran this against the index containing all available document fields. We submitted this run because this combination provided the best overall results on the training set.

Run 3 title.reviews-split.CombSUM This run used the titles of the test topics and ran this against the review-centric reviews-split index, using the unweighted CombSUM fusion method.

Run 4 all-topic-fields.reviews-split.CombSUM This run used all three topic fields combined and ran this against the review-centric reviews-split index, using the unweighted CombSUM fusion method.

[7] Our participant ID was 54.

[8] While our experiments showed that using only the title topic set did not provide the best results, submitting at least one run using only the title topic set was required by the track organizers.

6 Results

The runs submitted to the INEX Social Book Search track was examined using three different types of evaluations. In all three evaluations the results were calculated using NDCG@10, P@10, MRR and MAP, with NDCG@10 being the main metric. The first evaluation was using the 211 test set topics where the relevance judgments derived from the books recommended on the LibraryThing discussion threads of the 211 topics. Table 3 shows the results of this evaluation.

Table 3. Results of the four submitted runs on the test set, evaluated using all 211 topics with relevance judgments extracted from the LibraryThing forum topics. The best run scores are printed in bold.

Runs	NDCG@10	P@10	MRR	MAP
title.all-doc-fields	0.1129	0.0801	0.1982	0.0868
all-topic-fields.all-doc-fields	0.2843	0.1910	0.4567	**0.2035**
title.reviews-split.CombSUM	0.2643	0.1858	0.4195	0.1661
all-topic-fields.reviews-split.CombSUM	**0.2991**	**0.1991**	**0.4731**	0.1945

We see that, surprisingly, the best-performing runs on all 211 topics was run 4 with an NCDG@10 of 0.2991. Run 4 used all available topic fields and the unweighted CombSUM fusion method on the review-centric reviews-split index. Run 2, with all available document and topic fields was a close second.

For the first type of evaluation the book recommendations came from LibraryThing users who actually read the book(s) they recommend. The second type of evaluation conducted by the track participants enlisted Amazon Mechanical Turk workers for judging the relevance of the book recommendations for 24 of the 211 test topics. These 24 topics were divided so that they covered 12 fiction and 12 non-fiction book requests. The judgments were based on pools of the top 10 results of all official runs submitted to the track, evaluated using all 211 topics. Table 4 shows the results of this second type of evaluation.

Table 4. Results of the four submitted runs on the test set, evaluated using 24 selected topics with relevance judgments from Amazon Mechanical Turk. The best run scores are printed in bold.

Runs	NDCG@10	P@10	MRR	MAP
title.all-doc-fields	0.4508	0.4333	0.6600	0.2517
all-topic-fields.all-doc-fields	**0.5415**	0.4625	**0.8535**	**0.3223**
title.reviews-split.CombSUM	0.5207	**0.4708**	0.7779	0.2515
all-topic-fields.reviews-split.CombSUM	0.5009	0.4292	0.8049	0.2331

We see that consistent with the results on the training set the best-performing run on the 24 selected topics was run 2 with an NCDG@10 of 0.5415. Run 2 used all available topic and document fields. Runs 3 and 4 were a close second and third.

The third type of evaluation used the same 24 Amazon Mechanical Turk topics from the the second evaluation, but with the original LibraryThing relevance judgments. Table 5 shows the results of this third type of evaluation.

Table 5. Results of the four submitted runs on the test set, evaluated using 24 selected Amazon Mechanical Turk topics with relevance judgments extracted from the LibraryThing forum topics. The best run scores are printed in bold.

Runs	NDCG@10	P@10	MRR	MAP
title.all-doc-fields	0.0907	0.0680	0.1941	0.0607
all-topic-fields.all-doc-fields	**0.2977**	**0.1940**	**0.5225**	**0.2113**
title.reviews-split.CombSUM	0.2134	0.1720	0.3654	0.1261
all-topic-fields.reviews-split.CombSUM	0.2601	**0.1940**	0.4758	0.1515

We see that, again consistent with the results on the training set, the best-performing run on the 24 selected topics with LibraryThing judgments was run 2 with an NCDG@10 of 0.2977. Run 2 used all available topic and document fields. Run 4 was a close second and third.

We also see that for the same 24 topics, evaluation scores are much lower than for the second type of evaluation. This is probably due to the fact that the Amazon Mechanical Turk judgements are more directly focused on topical relevance. The LibraryThing judgments are more likely to reflect recommendations by people who have actually read the books they recommend. This is a more restrictive criterion, which results in lower evaluation scores.

7 Discussion & Conclusions

Both in the the training set and the test set good results were achieved by combining all topic and document fields. This shows support for the principle of polyrepresentation [8] which states that combining cognitively and structurally different representations of the information needs and documents will increase the likelihood of finding relevant documents. However, using only the split reviews as index gave in four cases in the test set even better results, which speaks against the principle of polyrepresentation.

We also examined the usefulness of user-generated metadata for book retrieval. Using tags and reviews in separate indexes showed good promise, demonstrating the value of user-generated metadata for book retrieval. In contrast, the effort that is put into curating controlled metadata was not reflected its retrieval performance. A possible explanation could be that user-generated data is much

richer, describing the same book from different angles, whereas controlled metadata only reflects the angle of the library professional who assigned them.

We also experimented with a review-centric approach, where all reviews were indexed separately and fused together at a later stage. This approach yielded good results, both on the training and the test set. We attempted to boost the performance of this approach even further by using review helpfulness and book ratings as weights, but this only decreased performance. At first glance, this is surprising since a helpful review can be expected to be well-written and well-informed. The quality of a book as captured by the rating could also be expected to have an influence on the review usefulness for retrieval, as could have been expected. Our current weighting scheme was not able to adequately capture these features though.

Our overall recommendation would therefore be to always use all available document fields and topic representations for book retrieval.

7.1 Future work

Future work would include exploring additional social re-ranking methods. As we are dealing with a book recommendation task, it would be a logical next step to explore techniques from the field of recommender systems, such as collaborative filtering (CF) algorithms. One example could be to use book ratings to calculate the neighborhood of most similar items for each retrieved book and use this re-rank the result list. The lists of (dis)similar items in the topic representations could also be used for this.

References

1. Zhai, C., Lafferty, J.: A Study of Smoothing Methods for Language Models Applied to Information Retrieval. ACM Transactions on Information Systems **22**(2) (2004) 179–214
2. Järvelin, K., Kekäläinen, J.: Cumulated Gain-based Evaluation of IR Techniques. ACM Transactions on Information Systems **20**(4) (2002) 422–446
3. Jindal, N., Liu, B.: Review Spam Detection. In: WWW '07: Proceedings of the 16th International Conference on World Wide Web, New York, NY, USA, ACM (2007) 1189–1190
4. Fox, E.A., Shaw, J.A.: Combination of Multiple Searches. In: TREC-2 Working Notes. (1994) 243–252
5. Lee, J.H.: Analyses of Multiple Evidence Combination. SIGIR Forum **31**(SI) (1997) 267–276
6. Renda, M.E., Straccia, U.: Web Metasearch: Rank vs. Score-based Rank Aggregation Methods. In: SAC '03: Proceedings of the 2003 ACM Symposium on Applied Computing, New York, NY, USA, ACM (2003) 841–846
7. Bogers, T., Van den Bosch, A.: Fusing Recommendations for Social Bookmarking Websites. International Journal of Electronic Commerce **15**(3) (Spring 2011) 33–75
8. Ingwersen, P.: Cognitive Perspectives of Information Retrieval Interaction: Elements of a Cognitive IR Theory. Journal of Documentation **52**(1) (1996) 3–50

Social Recommendation and External Resources for Book Search

Romain Deveaud[1], Eric SanJuan[1] and Patrice Bellot[2]

[1] LIA - University of Avignon
339, chemin des Meinajaries, F-84000 Avignon Cedex 9
{romain.deveaud,eric.sanjuan}@univ-avignon.fr

[2] LSIS - Aix-Marseille University
Domaine universitaire de Saint Jérôme, F-13397 Marseille Cedex 20
patrice.bellot@lsis.org

Abstract. In this paper we describe our participation in the INEX 2011 Book Track and present our contributions. This year a brand new collection of documents issued from Amazon was introduced. It is composed of Amazon entries for real books, and their associated user reviews, ratings and tags.
We tried a traditional approach for searching with two query expansion methods involving Wikipedia as an external source of information. We also took advantage of the social data with recommendation runs that use user ratings and reviews.

1 Introduction

Previous editions of the INEX Book Track focused on the retrieval of real out-of-copyright books [1]. These books were written almost a century ago and the collection consisted of the OCR content of over 50,000 books. It was a hard track because of vocabulary and writing style mismatches between the topics and the books themselves. Information Retrieval systems had difficulties to found relevant information, and assessors had difficulties judging the documents.

This year, for the books search task, the document collection changed and is now composed of the Amazon pages of real books. IR systems must now search through editorial data and user reviews and ratings for each book, instead of searching through the whole content of the book. The topics were extracted from the LibraryThing[1] forums and represent real requests from real users.

This year we experimented with query expansion approaches and recommendation methods. Like we already did last year, we used a Language Modeling approach to retrieval. We started by using Wikipedia as an external source of information, since many books have their dedicated Wikipedia article [2]. A Wikipedia article is associated to each topic, and the most informative words of the articles are selected as expansion terms. For our recommendation runs, we used the reviews and the ratings attributed to books by Amazon users. We

[1] http://www.librarything.com/

computed a "social relevance" probability for each book, considering the amount of reviews and the ratings. This probability was then interpolated with scores obtained by Maximum Likelihood Estimates computed on whole Amazon pages, or only on reviews and titles, depending on the run.

The rest of the paper is organized as follows. The following Section gives an insight into the document collection whereas Section 3 describes the our retrieval framework. Finally, we describe our runs in Section 4.

2 The Amazon Collection

The document used for this year's Book Track is composed of Amazon pages of existing books. These pages consist of editorial information such as ISBN number, title, number of pages etc... However, in this collection the most important content resides in social data. Indeed Amazon is social-oriented, and user can comment and rate products they purchased or they own. Reviews are identified by the `<review>` fields and are unique for a single user: Amazon does not allow a forum-like discussion. They can also assign tags of their creation to a product. These tags are useful for refining the search of other users in the way that they are not fixed: they reflect the trends for a specific product. In the XML documents, they can be found in the `<tag>` fields. Apart from this user classification, Amazon provides its own category labels that are contained in the `<browseNode>` fields.

Table 1. Some facts about the Amazon collection.

Number of pages (i.e. books)	2,781,400
Number of reviews	15,785,133
Number of pages that contain a least a review	1,915,336

3 Retrieval Model

3.1 Sequential Dependence Model

This year we used a language modeling approach to retrieval [3]. We use Metzler and Croft's Markov Random Field (MRF) model [4] to integrate multi-word phrases in the query. Specifically, we use the Sequential Dependance Model (SDM), which is a special case of the MRF. In this model three features are considered: single term features (standard unigram language model features, f_T), exact phrase features (words appearing in sequence, f_O) and unordered window features (require words to be close together, but not necessarily in an exact sequence order, f_U).

Finally, documents are ranked according to the following scoring function:

$$\begin{aligned} score_{SDM}(Q,D) = \lambda_T \sum_{q \in Q} f_T(q,D) \\ + \lambda_O \sum_{i=1}^{|Q|-1} f_O(q_i, q_{i+1}, D) \\ + \lambda_U \sum_{i=1}^{|Q|-1} f_U(q_i, q_{i+1}, D) \end{aligned} \quad (1)$$

where the features weights are set according to the author's recommendation ($\lambda_T = 0.85$, $\lambda_O = 0.1$, $\lambda_U = 0.05$).

3.2 Wikipedia as an External Resource

As previously done last year, we exploited Wikipedia in a Pseudo-Relevance Feedback fashion to expand the query with informative terms. We use the Wikipedia API to retrieve the best ranked article $\mathcal{W}$ for each topic. We then strip the HTML and tokenize the document into words. An entropy measure $H_{\mathcal{W}}(w)$ is then computed for each word w in $\mathcal{W}$:

$$H_{\mathcal{W}}(w) = -p(w|\mathcal{W}) \cdot \log p(w|\mathcal{W})$$

This measure gives a weight to each word in the page that reflects their relative informativeness. The top 20 words with best entropies are used as expansion words and are incorporated to the SDM model along with their weights:

$$\begin{aligned} score(Q,D) = score_{SDM}(Q,D) \\ + \lambda_{\mathcal{W}} \sum_{w \in \mathcal{W}} H_{\mathcal{W}}(w) \cdot f_T(w,D) \end{aligned} \quad (2)$$

where $\lambda_{\mathcal{W}} = 1$ in our experiments.

3.3 Wikipedia Thematic Graphs

In the previous methods we expand the query with words selected from pages directly related to the query. Here, we wanted to select broader, more general words that could stretch topic coverage. The main idea is to build a thematic graph between Wikipedia pages in order to generate a set of articles that (ideally) completely covers the topic.

For this purpose we use anchor texts and their associated hyperlinks in the first Wikipedia page associated to the query. We keep the term extraction process detailed in Section 3.2 for selecting a Wikipedia page highly relevant to the query. We extract informative words from this page using the exact same method as above. But we also extract all anchor texts in this page. The words selected with the entropy measure are considering as a set $T_{\mathcal{W}}$, as well as each anchor text.

We then compute and intersection between set $T_{\mathcal{W}}$ and each anchor text set. If the intersection is not null, we consider that the Wikipedia article that is linked with the anchor text is thematically relevant to the first retrieved Wikipedia article. We can iterate and construct a directed graph of Wikipedia articles linked together. Children node pages (or *sub-articles*) are weighted half that of their parents in order to minimize a potential *topic drift*. Informative words are then extracted from the sub-articles and incorporated to the query as previously described.

4 Runs

This year we submitted 6 runs for the Social Search for Best Books task only. We used Indri[2] for indexing and searching. We did not remove any stopword and used the standard Krovetz stemmer.

baseline-sdm This run is the implementation of the SDM model described in (1). We use it as a strong baseline.

baseline-tags-browsenode This is an attempt to produce an improved baseline that uses the Amazon classification as well as user tags. We search all single query terms in the specific XML fields (`<tag>` and `<browseNode>`). This part is then combined with the SDM model, which is weighted four times more than the "tag searching" part. We set these weights empirically after observations on the test topics. The Indri syntax for the query `schumann biography` would be:

```
#weight (
  0.2 #combine ( #1(schumann).tag #1(biography).tag
                 #1(schumann).browseNode #1(biography).browseNode )
  0.8 #weight ( 0.85 #combine( Schumann Biography )
                0.1  #combine( #1(schumann biography) )
                0.05 #combine( #uw8(schumann biography) ) )
        )
```

sdm-wiki This run is the implementation of the Wikipedia query expansion model described in Section 3.2. The Wikipedia API was queried on August, 2011.

sdm-wiki-anchors This run is the implementation of the Wikipedia thematic graph approach described in Section 3.3.

[2] `http://www.lemurproject.org`

sdm-reviews-combine This run uses the social information contained in the user reviews. First, a **baseline-sdm** is performed. Then for each document retrieved we extract the number of reviews and their ratings. A probability that the book is popular is then computed with a t-test. This "popularity score" is finally interpolated to the SDM score and documents are re-ranked.

recommendation This run is similar to the previous one except that we compute a query likelihood only on the `<title>` and on the `<content>` fields, instead of considering the whole document like the SDM does. Scores for the title and the reviews, and the popularity of the books are interpolated in a logistic regression fashion. The sum of these three scores gives a recommendation score for each book, based only on its title and on user opinions.

5 Conclusions

In this paper we presented our contributions for the INEX 2011 Book Track. We proposed two query expansion methods that exploit the online version of Wikipedia as an external source of expansion terms. The first one simply considers the most informative words of the best ranked article, whereas the second one focuses on building a limited thematic graph of Wikipedia articles in order to extract more expansion terms.

We also experimented with the use of the social information available in the Amazon collection. We submitted two runs that exploits the number of reviews and the user ratings to compute popularity scores that we interpolate with query likelihood probabilities.

References

1. Gabriella Kazai, Marijn Koolen, Antoine Doucet, and Monica Landoni. Overview of the INEX 2010 Book Track: At the Mercy of Crowdsourcing. In Shlomo Geva, Jaap Kamps, Ralf Schenkel, and Andrew Trotman, editors, *Comparative Evaluation of Focused Retrieval*, pages 98–117, Heidelberg, 2011. Springer.
2. Marijn Koolen, Gabriella Kazai, and Nick Craswell. Wikipedia pages as entry points for book search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 44–53, New York, NY, USA, 2009. ACM.
3. D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Inf. Process. Manage.*, 40:735–750, September 2004.
4. Donald Metzler and W. Bruce Croft. 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 '05, pages 472–479, New York, NY, USA, 2005. ACM.

The University of Massachusetts Amherst's participation in the INEX 2011 Prove It Track

Henry A. Feild, Marc-Allen Cartright, and James Allan

Center for Intelligent Information Retrieval
University of Massachusetts Amherst, Amherst MA 01003, USA
{hfeild, irmarc, allan}@cs.umass.edu

Abstract. We describe the process that led to the our participation in the Prove It task in INEX 2011. We submitted the results of six book page retrieval systems over a collection of 50,000 books. To generate book page scores, we use the sequential dependency model (a model that uses both unigrams and bigrams from a query) for two runs and an interpolation between language model scores at the passage level and sequential dependency model scores at the page level for the other four runs. In this report, we describe our observations of various retrieval models applied to the Prove It task.

Keywords: inex, prove it, book retrieval, sequential dependency model, passage retrieval

1 Introduction

In this report we describe our submissions to the 2011 INEX Prove It task, where the goal is to rank book pages that are supportive, refutative, or relevant with respect to a given fact. We did not participate in the optional sub task of classifying each result as confirming or refuting the topic; in our submissions we label all retrieved documents as *confirmed*. To determine what retrieval systems to submit, we investigated several models. In the following sections, we detail those models and give a summary of the results that led to our submissions.

2 Indexing and Retrieval Models

We only consider indexing pages. The index uses no other information about a page's corresponding book, chapter, or section. All tokens are stemmed using the Porter stemmer. It includes 6,164,793,369 token occurrences indexed from 16,971,566 pages from 50,232 books, built using a specialized version of the Galago retrieval system.[1]

We explored a number of models for page and passage retrieval, including relevance modeling, sequential dependence modeling, passage modeling, removing stop words, and mixtures thereof. We describe each below.

[1] http://galagosearch.org/

Query likelihood language modeling (QL). This model scores each page by its likelihood of generating the query [4]. The model also smooths with a background model of the collection; for this, we use Dirichlet smoothing with the default smoothing parameter: $\mu = 1500$.

Relevance modeling (RM). A form of pseudo relevance feedback, relevance modeling creates a language model from the top k pages retrieved for a query, expands the query with some number of the most likely terms from the model, and performs a second retrieval [2]. We investigate relevance modeling because, as with all pseudo relevance feedback methods, it allows the vocabulary of the original query to be expanded, hopefully capturing related terms. There are three parameters to set: the number of feedback pages to use (set to 10), the number of feedback term to use (also set to 10), and the weight to give the original query model and the relevance model for the second retrieval (set to 0.5). These are the default settings within Galago.

Sequential dependence modeling (SDM). This model interpolates between document scores for three language models: unigram, bigram, and proximity of adjacent query term pairs [3]. Because of its use of bigrams, SDM captures portions of phrases that unigram models miss. The weight of each sub language model are parameters that can be set, and we use the defaults suggested by Metzler and Croft [3]: 0.85, 0.10, 0.05 for the unigram, bigram, and proximity models respectively. In addition, each laguage model uses Dirichlet smoothing, and we experiment with $\mu = 1500$ (the Galago default) and $\mu = 363$ (the average number of terms per page).

Passage modeling (PM). This model first scores passages using QL with Dirichlet smoothing (setting μ to the length of the passage), selects the highest passage score per page, and then interpolates between that score and the corresponding page's SDM score. In our implementation, we first retrieve the top 1,000 pages (*Pages*) and the top 10,000 passages (*Pass*)[2] as two separate retrieval list, then the lists are interpolated. If a passage is present in *Pass*, but the corresponding page is not in *Pages*, the page score is set to the minimum page score in *Pages*. Likewise, if a page is retrieved in *Pages* but no passages from that page are present in *Pass*, the lowest passage score in *Pass* is used as a proxy. The parameters of the PM model include the passage length l and the interpolation factor, λ, where the maximum passage score is weighted by λ and the page score is weighted by $1 - \lambda$.

Stop word removal (Stop). When stopping is used, query terms found in a list of 119 stop words[3] are removed.

We considered several combinations of the above models, all using stemming. These include: LM, RM, SDM, SDM+RM, PM, and each of these with and without stop words removed.

[2] We allow multiple passages per document to appear on this list; filtering the highest scoring passage per page is performed on this 10,000 passage subset.

[3] http://www.textfixer.com/resources/common-english-words.txt

Field	Example 1	Example 2	Example 3
ID	2010000	2010012	2010015
Fact	In the battle of New Orleans on the 8th of January 1815, 2000 British troops were killed, wounded or imprisoned, while only 13 American troops were lost, 7 killed and 6 wounded.	The main function of telescope is to make distant objects look near.	Victor Emanuel enters Rome as king of united Italy.
Info need	All sections of books that detail the losses suffered either at the British or the American side are relevant. I am not interested in how the battle was fought, but just want to find out about the losses at the end of the battle.	Most of the book is relevant to Astronomy as its a handbook on astronomy.	Italy will celebrate next year its re-unification and I needed to check the facts and their dates. Italy had two other capitals, Turing and Florence, before it was possible to get Rome back from the Vatican State.
Query	New Orleans battle 1815 troops lost killed	Telescope	Rome capital
Subject	battle of New Orleans 1815	telescope	Rome becomes capital of united Italy
Task	My task is to find out the scale of losses on both the British and American side in the battle of New Orleans in 1815	We need to write a primer on Astronomy.	Find out the date when Rome became capital of reunited Italy.

Table 1. The INEX Prove It topic fields.

3 Training data

Of the 83 total topics available for the Prove It task, 21 have judgments to evaluate submissions from last year's workshop. We use these as our training set; all of the observations and results discussed in this report use only this training data.

Inevitably, new systems pull up unjudged book pages in the top ten ranks. To handle these cases, we developed a judgment system with which lab members, including the authors, annotate pages as being *supportive*, *refutative*, or *relevant* in the case that a page is on topic, but is not distinctly and completely supportive or refutative. The system displays all fields of a topic, making the annotator as informed as possible. The fields are listed in the first column of Table 1 along with three examples of the field contents. The *info need* field usually describes what should be considered relevant, and the accessors are asked to abide by this. Some topics are tricky to judge, as in the case of Example 3 in Table 1 (Topic 2010015), where the broad focus is clearly on Italy, but the specific information being sought is inconsistent across the fields. In cases such as these, annotators interpret the information need and ensure that all pages are judged relative to that interpretation of the topic.

We manually added 535 relevance judgments to the training set. This covers many of the unjudged documents the systems retrieved in their top 10 lists for each topic.

System	NDCG@10 Stopped	Unstopped
$LM_{\mu=1500}$	0.811	0.811
RM	0.751	0.701
$SDM+RM_{\mu=1500}$	0.755	0.751
$SDM_{\mu=1500}$	0.834	0.854
$SDM_{\mu=363}$	0.828	0.854
$PM_{l=100,\lambda=0.25}$	0.856	0.859
$PM_{l=50,\lambda=0.25}$	0.863	0.873

Table 2. The results of several systems over the 21 training topics .

4 Results

In this section we discuss the performance of the models listed in Section 2. We evaluate over all 21 training topics. Each model considers only the *fact* field of each topic; when using the *query* field, our best models only outperformed the better systems from last year's track by a small margin [1]. The substantial difference in performance between using the the two fields stems from the poor representation of the information need in most topics' *query* field. Consider Example 2 in Table 1: the query "telescope" does not adequately describe the information need, which is the assertion that the primary function of a telescope is to magnify distant objects.

Table 2 reports the normalized discounted cumulative gain at rank 10 (NDCG-@10) of the systems with and without stopwords removed. We binarize the graded relevance judgements such that the *supportive*, *refutative*, and *relevant* labels are conflated. The relevance models do not perform as well as the others, though this is partially due to not having enough judgments. Even if the unjudged documents are assumed relevant, SDM outperforms RM in the unstopped case, and RM only marginally improves over SDM in the stopped case. Setting μ to the average page length is not helpful for SDM, however, we entered $SDM_{\mu=363}$ as a submission because the choice of μ is more principled than the Galago default.

The PM models outperform the others, with a passage size of 50 terms taking the lead. To understand why we choose $\lambda = 0.25$,[4] see Figure 1 (this only shows the variation with stop words removed). For both 50 and 100 term passages, it is clear that a value of λ in the $[0.20, 0.30]$ range, and specifically 0.25, is optimal. This places much of the final page score on SDM, but still gives a substantial amount of weight to the maximum LM passage score.

SDM captures pieces of phrases in a fact, and these seem to be important given the results. PM adds the notion of tight proximity—a high passage score ideally applies to passages that are topical hot spots. By setting $\lambda = 0.25$, the model ranks higher pages that seem relevant overall and also contain topical hot spots. The page's content is more important than the passage content, however. Said differently, a page with many medium scoring passages can be ranked higher than a page with one very high scoring passage. A manual assessment

[4] Our submissions' names suggest we used $\lambda = 0.025$, however this was a typo.

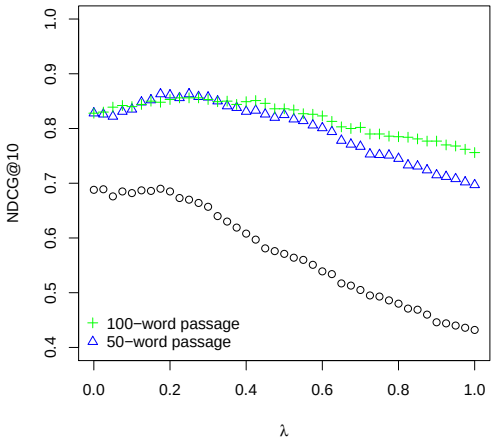


Fig. 1. A sweep over the λ parameter for the passage model. Smaller λ values mean more weight is given to the SDM score of the page, while higher values mean more weight is given to the highest scoring passage (using QLM). All queries were sopped.

of the retrieved documents supports this observation—pages that only have one high scoring maximum passage are often non-relevant. The passage may make reference to an aspect of the topic, but provides no in depth information. Perhaps an artifact of the format of books, relevant book sections tend to discuss topics over several paragraphs and even pages.

5 Summary

We considered several systems to retrieve supportive and refutative book pages for a given fact as part of the 2011 INEX Prove It task. We found that sequential dependence modeling (SDM) and passage-page interpolation (PM) perform best.

References

1. Geva, S., Kamps, J., Schenkel, R., Trotman, A. (eds.): Pre-proceedings of the INEX workshop (2010)
2. Lavrenko, V., Croft, W.: Relevance based language models. In: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 120–127. ACM (2001)
3. Metzler, D., Croft, W.B.: 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. pp. 472–479. SIGIR '05, ACM, New York, NY, USA (2005)
4. Ponte, J., Croft, W.: A language modeling approach to information retrieval. In: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. pp. 275–281. ACM (1998)

TOC Structure Extraction from OCR-ed Books

Caihua Liu[1,†], Jiajun chen[2,†], Xiaofeng Zhang[2,†], Jie Liu[1,⋆], and Yalou Huang[2]

[1] College of Information Technical Science, Nankai University, Tianjin, China 300071
[2] College of Software, Nankai University, Tianjin, China 300071
{liucaihua,chenjiajun,zhangxiaofeng}@mail.nankai.edu.cn
{jliu,ylhuang}@nankai.edu.cn

Abstract. This paper addresses the task of extracting the table of contents (TOC) from OCR-ed books. Since the OCR process misses a lot of layout and structural information, it is incapable of enabling navigation experience. A TOC is needed to provide a convenient and quick way to locate the content of interest. In this paper, we propose a hybrid method to extract TOC, which is composed of rule-based method and SVM-based method. The rule-based method mainly focuses on discovering the TOC from the books *with* TOC pages while the SVM-based method is empolyed to handle with the books *without* TOC pages. Experimental results indicate that the proposed methods obtain comparable performance against the other participants of the ICDAR 2011 Book structure extraction competition.

Keywords: table of contents, book structure extraction, xml extraction

1 Introduction

Nowadays many libararies focus on converting the whole libraries by digitizing books on an industrial scale and this project is referred as '*digital libararies*'. One of the most important tasks in digital libararies is extracting the TOC. A table of contents (TOC) is a list of TOC entries each of which consists of three elements: title, page number and level of the title. The intention of extracting TOC is to provide a convenient and quick way to locate content of interest.

To extract the TOC of the books, we are faced with several challenges. First, the books are various in forms, since the books come from different fields and there are kinds of layout formats. A large variety of books increases the difficutlty of utilizing a uniform method to well extract the TOC. Taking the poems for example, some poems contain TOC pages while some not. The alignment of poems may be left-aligned, middle-aligned and right-aligned. Second, due to the limitation of OCR technologies, there are a certain number of mistakes. OCR mistakes also cause trouble in extracting TOC, especially when some keywords such as 'chapters', 'sections', etc., are mistakenly recognized.

[†] The first three authors make equal contributions
[⋆] Corresponding author

Many methods have been proposed to extract TOC, most of which are published in INEX[1] workshop. Since 80% of these books contain table of contents, MDCS [8] and Noopsis [1] took the books with table of content into consideration. While the University of Caen [5] utilized a four pages window to detect the large whitespace, which is considered as the beginning or ending of chapters. XRCE [3, 6, 7] segmented TOC pages into TOC entries and used the references to obtain page numbers. XRCE also proposed a method trailling whitespace.

In this paper, we propose a hybrid method to extract TOC, since there are two types of data. 80% of the books contain TOC pages and the remaining do not. This two situations are considered via rule-based method and SVM-based method respectively. For books containing TOC pages, some rules are designed to extract these TOC entries. The rules designed are compatible with the patterns of most books, which is also demonstrated in the experiments. For books without TOC pages, a SVM model is trained to judge whether one paragraph is a title or not. A set of features is devised for representing each paragraph in the book. These features also do not depend on knowledge of TOC pages. Using these features and the machine learning method we can extract TOC entries, whether the book has an TOC page or not. To better organize these TOC entries, the level and the page number of each TOC entry locating are also extracted, besides these TOC entries themselves.

The paper is organized as follows. In section 2, we give a description of the previous works about the extraction of TOC. An introduction about the books and the format of these data is presented in section 3. The main idea of our works to extract TOC entries is shown in section 4. In section 5, we locate the target page for each TOC entry. We assign levels for TOC entries in section 6. Finally, experiments and a short conclusion are displayed.

2 Related works

In the application of digital libraries, there are four main technologies, information collecting, organizing, retrieving and security. Organizing data with XML is the normal scheme, especially when we are faced with large scales of data. A information retrieval workshop named INEX has been organized to retrieve information from XML data. In 2008, BSE(Book Structure extraction) was added to INEX, whose purpose is to evaluate the performance of automatic TOC structure extraction from OCR-ed books.

MDCS [8] and Noopsis[1] focus on books containing TOC pages. Except for locating the TOC entries, they make no use of the rest of the books. MDCS emploies three steps to extract TOC entries. Firstly, they recognize TOC pages. Secondly, they assgin a physical page number for every page. Finally, they extract the TOC entries via a supervised method relying on pattern occurrences detected. MDCS's method depends on the TOC pages and it can not work for books without TOC pages. University of Caen's [5] method did not rely on the

[1] http://www.inex.otago.ac.nz/

content page, the key hypothesis of which is that the large whitespace is the beginning of one chapter and the ending of another chapter. So they utilized a four pages window to detect the whitespace. However, it works well on high-level title but the lower title can not be recognized well. Xerox Research Center Europe [3, 6, 7] used four methods to extract the TOC entries. First two are based on TOC pages and index pages. The third method is similar to MDCS, which also defines some patterns while the last method trails the whitespace like University of Caen does. The method proposed by us is more effecient than others, since we directly extract TOC entries by analyzing the TOC pages for books with TOC pages. Due to the diversity of books without TOC pages, it's difficult to find a uniform rule or pattern to extract the TOC entries. To these issues, we perform an automatic learning method for extracting TOC entries.

3 The architecture of the hybrid extracting method

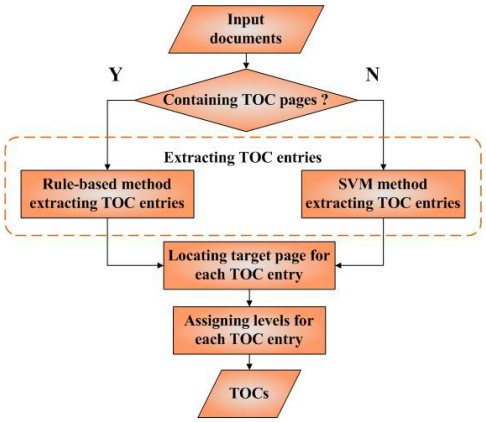


Fig. 1. The flow chart of our hybrid extracting system.

In this section, we will give a discription of the architecture of our method. Since 80% of the books contain TOC pages while the remaining do not, we consider these two situations respectively. One more crucial reason is that the TOC pages contain a lot of hidden structural contents. The well using of the TOC pages can help improve the extracting performance. In addition, for books with TOC pages, directly extracting TOC from the TOC pages performs better than extracting TOC from main text.

As previously stated, each TOC entry contains three parts: title, page number and level of the title. As shown in Figure 1, the extracting process is conducted with the following steps. (1) A judgement is conducted to seperate the books into two parts via whether containing TOC pages. (2) Extracting each TOC

entry and obtaining the title, page number. Since there are two types of data, we extract the TOC from them respectively. For books containing TOC pages, a rule-based method is proposed to extract these TOC entries from the TOC pages. For books without TOC pages, a SVM method is introduced to achieve the purpose of extracing TOC. (3) Locating the target page for each TOC entry. (4) Assigning levels for these extracted TOC entries. A simple relationship between these TOC entries can be obtained by this step.

After these steps, each TOC entry has been extracted. we also obtain the target pages and the organizational structure of these TOC entries. Then the TOC is outputed as the predefined format. Until now, all of the works to extract TOC has been accomplished. And the specific methods to conduct this three steps is stated in the following sections.

4 Extracting TOC entries

In this section, we focus on the two methods to extract TOC entries. The following two sections will give a specific introduction of these two methods respectively.

4.1 Extracting with TOC pages

We extract the TOC from the original TOC page in the book with TOC pages. This task is divided into two steps: locating the TOC pages of the book and extracting TOC entries from the TOC pages.

Locating TOC Pages If a book contains TOC pages, we extract the TOC entries from the TOC pages directly. Naturally, the TOC pages start with key words such like '*Contents*' and '*Index*', and the TOC page contains many lines ending with numbers. We can use these features to locate the beginning of contents pages. Since most books have headers, the pages are ascertained to be TOC pages, if there are key words like '*Contents*' and '*Index*' appearing at the beginning of the page, or if there is a considerable number of lines ending with numbers. Naturally, TOC pages usually appear in the front part of a book, as a result, we only need to consider the first half of the book to accelerate the process.

Extracting Contents Entries TOC entries in books vary greatly in different books. As is shown in Figure 2, some entries like Figure 2(a) occupy only one line, while some entries Figure 2(b) occupy several lines. Some entries Figure 2(c) are divided into multi-lines, of which the first line is text, the second line is logical page number and the third line is introduction of the section.

To extract each TOC entry, we need to obtain the beginning and the ending of each TOC entry. The following rules are conducted to identify the beginning of TOC entries.

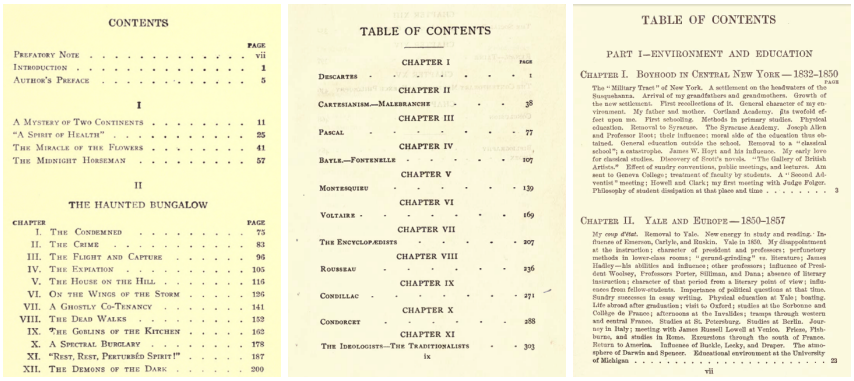


Fig. 2. A variety of Contents structure. (a) Each TOC entry occupy one line; (b) Each TOC entry occupy serveral lines; (c) Each TOC entry occupy multi-lines

1. if current line starts with key words like '*Chapter*', '*Part*', '*Volume*' or '*Book*' etc.
2. if current line starts with numbers or Roman numerals.
3. if last line ends with numbers or Roman numerals.

The start of a new TOC entry is the end of last TOC entry, so we can easily construct the following rules to identity the end of TOC entries.

1. if next line starts with key words like '*Chapter*', '*Part*', '*Volume*' or '*Book*' etc.
2. if next line starts with numbers or Roman numerals.
3. if current line ends with numbers or Roman numerals.

The above rules can handle most of the situations, however, some TOC entries such as multi-lines can not be well extracted using only those rules. To these issues, a new rule is added. If the last line does not have key words like '*Chapter*' and Roman numerals etc. that obviously separate contents items and the formats of current line and last line are very different, delete line information collected before.

The new rule treats current line as the start of a new TOC entry, and delete stored lines formerly should be treated as part of current TOC entry. The difficulty of this rule lies in the quantitative description of differences between two lines. Our approach consider only the relative font sizes.

$$relative\_font\_size = \frac{currentlinefontsize - averagefontsize}{maxiumfontsize - avergefontsize} \tag{1}$$

If the '*relative_font_size*' of the two lines are very different and greater than some pre-set thresholds, these two lines can not be treated as one TOC entries

4.2 Extracting without TOC pages

For the books without TOC pages, we use SVM to extract TOC entries. It comes into the following steps: (1) extracting the features of each paragraph and labeling them(2) training the RBF-SVM to classify every paragraph.

Features Through observing data set, some obvious features can be employed to identitfy a TOC entry, as shown in table 1. Though we get the eight features, it

Table 1. Features designed for books without TOC pages.

Feature ID	Discription
1	Proportion of Capital Letters
2	Font Size
3	Left End position of a Paragraph
4	Right End Position of a Paragraph
5	Space between Paragraph
6	Line Number of a Paragraph
7	Average Number of words in Each Line of a Paragraph
8	The y-coordinate of a Paragraph Start

happens that some common paragraphs have one or more features. For example, the page header is much more similar to the TOC entry, so this will confuse the classifier. Commonly the page header always has the same content with the title, while it has a lot of duplications. So we use post-process method to delete the duplication and make the first page header that has the same content with others as the title. Another example, the title page (this page only contains a title, and in the next page it also starts with this title in the top) has also some of the features, what's more the effect of the features is more obvious than the title at times. So we must do some efforts to solve the problem. According to the title page, we set a threshold to judge whether a page is title page. It is means that if most of the paragraphs in the page are recognized as title, we think it is a title page.

Recongnizing the TOC entries We use SVM[2] to identify whether one paragraph is TOC entry. Since many normal paragraphs are predicted as positive, a further analysis is made on these data. There are four situations to consider: the title we expect, page header, the misrecognized paragraph and the spot or handwritten note in the book which can be OCRed. In order to get a much higher performance, a post-process is conducted. And the following principles are devised to delete some of the positive ones.

1. If there are less capital letters in a paragraph than a threhold.

[2] http://www.cs.cornell.edu/people/tj/svm_light.html

2. If a paragraph only contains a letter, and it is not Rome number as well.
3. If a paragraph is similar to others,then keep the first one and delete the others.
4. If there are more than two positive paragraphs.

5 Locating target page

After extracting TOC entries, we need to ascertain where the entries actually locate for navigation purpose. The page numbers shown in TOC are the logical numbers. While the physical number shows the actual page number in the whole document. Hence the matching of physical page number and logical page number is expected to help users navigate over the whole document.

To match the physical and the logical page numbers, the logical numbers for every page are needed to be extracted first. Commonly, the logical number appears in the headers or footers. However, logical pages extracted in this way are not prefect enough, as some pages may indeed do not have logical page numbers or maybe an OCR error makes the logical page numbers not recognized correctly, so we need to deal with those omissions and errors. So a remedy is conducted to obtain the complete page numbers. First, fill the vacancies of pages without page numbers using the following method. If the physical page i and j $(j > i)$ have logical page $L(i)$ and $L(j)$ respectively, and logical page numbers of pages between i and j are absent, at the same time, if $L(j) - L(i) = j - i$, fill the vacancies of logical page numbers for those page between i and j.

Whereas, there are still some pages with physical numbers which can not find logical number. For these physical pages we set them to 0. And then the logical page numbers of the extracted contents items are replaced to physical page numbers. For these physical pages labeled '0', We first find the the maximum logical page number smaller then current logical page number, and match with physical page number. Then from this maximum logical page number and forward we use text match to find the first page that contains the text of current TOC entry, and use the physical page number of this page as the physical page of current TOC entry.

6 Ranking levels for TOC entries

The final step is to rank those extracted TOC entries. Via the analysis of the data, we find that most of the content entries contain the key term '*Book*','*Volume*','*Part*', '*Chapter*' and so on. While information like arabic numerals and roman numerals can also be utilized to assign the level for content entries. So we pre-define five levels to arrange the levels of every contents entry.

1. First level: containing key words '*Part*', '*Volume*' and '*Book*' etc.
2. Second level: containing keywords '*Chapter*' and '*Chap*' etc.
3. Third level: containing keywords '*Section*' and '*Sect*' etc.

4. Forth level: containing Arabic numerals and Roman numerals or keywords like '*(a)*' and '*a*'.
5. To be ascertained level: other TOC entries that do not have above mentioned features while its level depends on their neighbors' contents, for example, previous rank.

A specific statistics on randomly selecting 100 books from the ICDAR 2011 dataset is shown in Figure 3. 70% of TOC entries contain keywords 'chapter' etc. and books with keyword 'Section' only occupy a small proportion of 100 books. More obvious is that 90% of the books correspond to our definition of levels. We first scan the whole content entries and assign levels for every entry

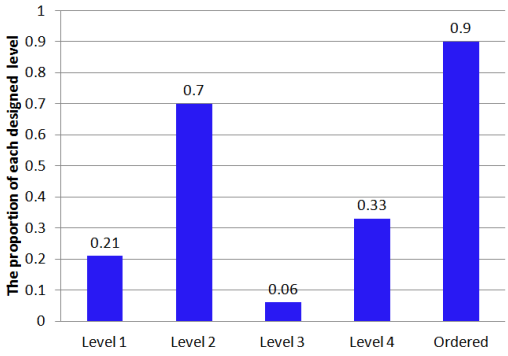


Fig. 3. The percent of each level and 'Ordered' means that the level of TOC corresponds to the level we defined.

by the rule pre-defined above. Most of these entries are all assigned levels, only these entries without any characteristics left. A statistics has been conducted by us and it demonstrates that these left entries have a higher probability of the same level as the previous one, therefore, the levels for these left entries are assigned via this idea.

7 Experiments

In order to measure the performance of our method, we conduct experiment on two datasets. One is the 1000 books provided by the Book structure extraction competition, while the other is ICDAR 2009 competition dataset. The pdf and DjVuXML format of the books are both provieded in these two datasets.

To give a full evaluation of the performance, three evaluate criterions are considered on five aspects. The evaluation measures are: precision, recall and F-measure. And the 5 aspects are: (1) Titles, which evaluates whether the titles we obtained are sufficiently similar to the titles in the ground truth; (2) Links,

which is correctly recognized if the TOC entries recognized by our method link to the same physical page in the ground truth. (3) Levels, which means whether established level of the title is at the same depth in the ground truth. (4) Complete except depth, which represents the title and the link are both right. (5) Complete entries, which is considered right only when all of these three items are right.

Table 2. The performance of our method on five evaluation aspects conducting on ICDAR 2011 dataset.

Items	Precision	Recall	F-Measure
Titles	47.99%	45.70%	45.20%
Links	44.03%	41.44%	41.43%
Level	37.91%	36.84%	36.08%
Entries disregarding depth	44.03%	41.44%	41.43%
Complete entries	34.80%	33.28%	33.06%

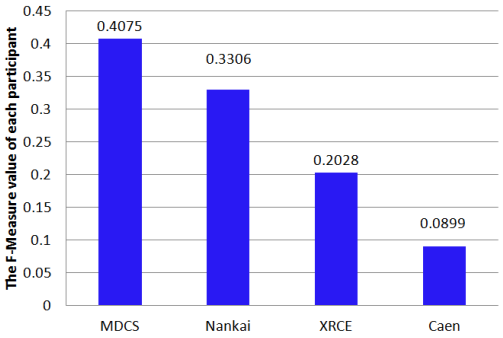


Fig. 4. The public results of ICDAR 2011 book structure extraction competition. It is conducted on the whole dataset and we rank second.

7.1 Experiments on the whole dataset

The experiments listed in this section are the public experimental results published by the official organizing committee of ICDAR. The performance of our method on the five aspects are reported in table 2. The performance comparision between our method with other participants of ICDAR is presented in Figure 4. It can be seen that MDCS outperforms others, but it is a TOC based method

and it can not deal with books without table well. We rank second in the completion, however, we are capable of processing those books without contents. To address these issues, our method is comparable to others.

7.2 Experiments on books without table of contents

To evaluate the ability of processing books without TOC pages, we conduct experiments on 2009 ICDAR dataset. In this dataset, there are ninty three books without TOC pages. Since the number of normal context is much larger than the number of TOC entries, so we use all of the TOC entries and randomly select the same number of normal text.

The result of five fold-cross validation is shown in Table 3. All of these five evaluate aspects are considered to give a full description of our method. Figure 5 shows the comparation with other methods public published in ICDAR 2009. It can be seen that our method outperforms others'.

Table 3. The performance of our method on five evaluation aspects conducting on ICDAR 2009 dataset.

Items	Precision	Recall	F-Measure
Titles	14.85%	23.64%	14.38%
Links	10.88%	16.17%	10.71%
Level	11.78%	19.62%	11.40%
Entries disregarding depth	10.88%	16.17%	10.71%
Complete entries	8.47%	13.07%	8.43%

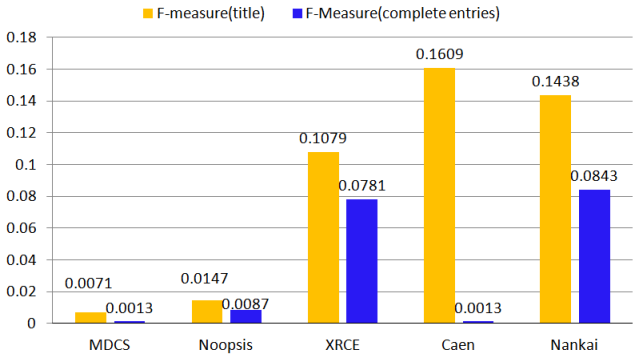


Fig. 5. Experiment on books without TOC pages and it is conducted on 93 books of ICDAR 2009 dataset.

8 Conclusion

This paper presents the task of extracting TOC entries for navigation purpose. However, extracting TOC entries from ocr-ed books is a challenging problem, since the OCR process misses a lot of layout and structural information. We proposed an effective method to solve this problem. For books containing contents page, a rule based method is conducted. For books without contents page, we utilize a machine learning method to classify the title. Besides, recognition of the title, the matching of physical and logical page is conducted to help users navigate. To get more specific information about the book, the partition of the level of title is also performed. The experiments show that our method considering these three aspects is usable and effective. A uniform model to effectively address the problem is expected as a future work.

References

1. Antoine Doucet, Gabriella Kazai, Bodin Dresevic, Aleksandar Uzelac, Bogdan Radakovic,Nikola Todic: Setting up a Competition Framework for the Evaluation of Structure Extraction from OCR-ed Books, in International Journal of Document Analysis and Recognition (IJDAR), special issue on "Performance Evaluation of Document Analysis and Recognition Algorithms", 22 pages, 2010.
2. Antoine Doucet, Gabriella Kazai, Bodin Dresevic, Aleksandar Uzelac, Bogdan Radakovic, Nikola Todic: ICDAR 2009 Book Structure Extraction Competition", in Proceedings of the Tenth International Conference on Document Analysis and Recognition (ICDAR'2009), Barcelona, Spain, July 26-29, p.1408-1412, 2009.
3. Herv Djean, Jean-Luc Meunier: Reflections on the INEX structure extraction competition, in Proceedings of the 9th IAPR International Workshop on Document Analysis Systems (DAS'2010), Boston, Massachusetts, p.301-308, 2010.
4. Herv Djean ,Jean-Luc Meunier: a useful level for facing noisy data, in Proceedings of the fourth workshop on Analytics for noisy unstructured text data (AND '10), Toronto, Canada, p.3-10, 2010.
5. Emmanuel Giguet, Nadine Lucas: The Book Structure Extraction Competition with the Resurgence Software at Caen University, in Advances in Focused Retrieval: 9th International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX 2010), Schloss Dagstuhl, Germany, p.170-178, 2010.
6. Herv Djean, Jean-Luc Meunier: XRCE Participation to the 2009 Book Structure Task, in Advances in Focused Retrieval: 9th International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX 2010), Schloss Dagstuhl, Germany, p.160-169, 2010.
7. Herv Djean, Jean-Luc Meunier: XRCE Participation to the Book Structure Task, inAdvances in Focused Retrieval: 8th International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX 2009), Schloss Dagstuhl, Germany, p.124-131, 2009.
8. Bodin Dresevic, Aleksandar Uzelac, Bogdan Radakovic, Nikola Todic: Book Layout Analysis: TOC Structure Extraction Engine, in Advances in Focused Retrieval: 8th International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX 2009), Schloss Dagstuhl, Germany, p.164-171, 2009.

OUC's participation in the 2011 INEX Book Track

Michael Preminger[1] and Ragnar Nordlie[1]

Oslo University College

Abstract. In this article we describe the Oslo University College's participation in the INEX 2011 Book track. In 2010, the OUC submitted retrieval results for the "prove it" task with traditional relevance detection combined with some rudimental detection of confirmation. In line with our belief that proving or refuting facts are different semantic aware actions of speech, we have this year attempted to incorporate some rudimentary semantic support based on the wordnet database.

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 refutal) of factual claims encountered during research.

On the surface, this may be seen as simple retrieval, but proving a fact is more than finding relevant documents. This type of retrieval requires from a passage to "make a statement about" rather than "be relevant to" a claim, which traditional retrieval is about. The questions we posed in 2010 were:

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

We also made the claim that "prove it" sorts within the (not very well-defined) category "semantic-aware retrieval", which, for the time being will be defined by us as retrieval that goes beyond simple string matching, and is aware of the meaning (semantics) of text.

Those question, being rhetorical in part, may be augmented by the questions

- How can one detect the meaning of texts (words, sentences and passages) and incorporate those in the retrieval process to attain semantic-aware retrieval

and consequently

- can one exploit technologies developed within the semantic web to improve semantic-aware retrieval

The latter is not directly addressed in this paper, but we claim that the techniques used here point in this direction.

2 The Prove-it task

2.1 Task definition and user scenario

The prove-it task is still at its infancy, and may be subject to some modifications in the future. Quoting the user scenario as formulated by the organizers

> 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. Users expect to be pointed directly at book pages that can help them to confirm or refute the claim of the topic. Users are assumed to view the ranked list of retrieved book pages starting from the top of the list and moving down, examining each result. No browsing is considered (only the returned book pages are viewed by users).

This user scenario is a natural point of departure as it is in the tradition of information retrieval and facilitates the development of the task by using existing knowledge. As a future strategy, it may be argued that this user scenario is gradually modified, as ranking in the context of proving is a highly complex process, and, in the context where Prove-it algorithms are most likely to be used, arguably superfluous.

2.2 Proving a claim vs. being relevant to a claim

As we wrote in **??**, we see proving a claim as being relevant to the claim and, in addition, possessing some characteristics as a document. The document characteristics are associated with the type of semantics (in the broad sense of the word) within which the text is written. Here there is a wide range of issues that need to be researched into: the language of the page, the type of words it contains, what kind of other claims there are in the document, and so on. These latter aspects are orthogonal to the statement or claim itself in the sense that they (at least ideally) apply equally to whatever claim being the subject of proving / confirming.

2.3 Ranking proving efficiency?

In this paper we are still following the two step strategy of first finding pages relevant to the claim, and from those pages trying to identify pages that are likely to prove the claim[1]. The first step is naturally done using current strategies for ranked retrieval. The second stage identifies *among relevant documents* those which prove / confirm the statement. Relevance ranking is not necessarily preserved in this process: if document A comprises a better string-wise match with the claim than does document B, document B can still be more efficient at proving the claim than document A. Not all elements that make a document relevant also make it a good prover

Another issue is the context in which prove-it is used. One example is the writing of a paper. A writer is (again, arguably) more likely to evaluate a greater number of sources for proof of a claim than he or she would in a context of pure fact finding or reference finding. Additionally, different contexts would arguably invite different proof emphases. All this advocates for use of other strategies of presenting proving results than ranked lists.

2.4 Semantic approaches to proof

A statement considered as a "proof" (or confirmation) may be characterized semantically by several indicators: - the phenomenon to be supported may be introduced or denoted by specific terms, for instance verbs indicating a definition: "is", "constitutes", "comprises" etc. - terms describing the phenomenon may belong to a specific semantic category - nouns describing the phenomenon may be on a certain level of specificity - verbs describing the phenomenon may denote a certain type of action or state Deciding which specificity level or which semantic categories will depend on the semantic content and the relationship between the terms of the original claim. Without recourse to the necessary semantic analysis, we assume that in general, terms indicating a proof / confirmation will be on a relatively high level of specificity. It will in some way constitute a treatment of one or more aspects of the claim at a certain level of detail, which we expect to be reflected in the terminology which is applied.

As an initial exploration of these potential indicators of proof, without access to semantic analysis of the claim statements, we are investigating whether terms, in our case nouns, found on a page indicated as a potential source of proof diverges in a significant way from other text in terms of level of specificity. We determine the level of noun specificity through their place in the Wordnet term hierarchies.

3 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

[1] We still see refutal as a totally different type of task and will not address it in this paper.

book (the context of the retrieved page) to be ABOUT the topic covering the fact. In this way the prove it task is different than e.g. the one referred to in **??** 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.

In line with the above, indexing should facilitate two main aspects at retrieval time: identifying relevant pages and finding which of these is likely to prove a claim. The first aspect is done by creating a simple index of all the words in the corpus, page by page. The pages are treated as separate documents regardless of common book belonging. The second aspect is catered for by calculating the average specificity of each page and tagging each page by one of a number of specificity tags.

Since this contribution is a direct further step from our 2010 contribution, we would also like to try and merge 2010's and this year's "proving detection". We do this by also creating a merged index where the pages are tagged by their average specificity as well as their "confirmatory word" occurrence rate.

3.1 Calculating specificity

At this stage of the research, the second aspect is catered for by statistically measuring the average specificity of words that occur in the page. We do this by calculating the specificity of each word and then averaging the measure of specificity of all the words in a page, as described below. To accomplish that, we have augmented the WordNet database (ref) by a Direct Acyclic Graph (DAG) of all the nouns, which lets us calculate a relative specificity of each word by its average position in this graph. Words closer to the common root of the graph (measured as a number of steps) are less specific, whereas words closer to the leaves are more specific. For each word in a trajectory, the specificity S is calculated as

$$S = \frac{P}{L},$$

where P is the position of the words in the trajectory (number of steps away from the root) and L is the length of the trajectory from root to leaf. Since this is a graph and not a tree, each word (a string of characters), even a leaf, may belong to more than one trajectory depending on the number of senses / synsets it participates in, and the number of parallel synsets it is a descendent of. Since we generally cannot know which sense of a word a certain occurrence stands for, we assign to each word (string of characters) the average of its specificities. Each page is then assigned the average of the specificities of its constituent words. Words not in the graph are assigned the "neutral" value of 0.5.

The pages are then categorized into predefined intervals of average specificity, where each interval has its own tag. These tags then facilitate weighting pages differently at retrieval time 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.

As in the 2010 contribution, 21 of 83 topics used for retrieval evaluation.

4 Runs and Results

The results presented here are an attempt at relating this year's results to the 2010 results. Taken directly from our 2010 contribution, figure 1(a) shows the results of weighting pages featuring 3 percents or more confirmatory words at retrieval time, weighted double, quintuple (5x) and decuple (10x) the baseline[2]. Our baseline was normal, non-weighted retrieval, as we would do for finding relevant pages. The analysis was done against a filtered version of the official qrel file, featuring only the pages assessed as confirming/refuting. Figure 1(b) shows the result when using double weight of the pages featuring 55% specificity as baseline, along with the results when superimposing on them the weighting of pages with more than 3% occurrence of confirmatory words

In absolute retrieval performance terms, the results in figure 1(b) feature a substantial improvement potential. They are still interesting. Here, superimposing the confirming words improves the baseline much more than in the 2010 results (figure 1(a)). Considering the use of a specificity measure as a stronger semantic characterization of a document (in the sense that it is less dependent on the occurrence of certain pre-chosen words), improving and adapting the semantic analysis is promising.

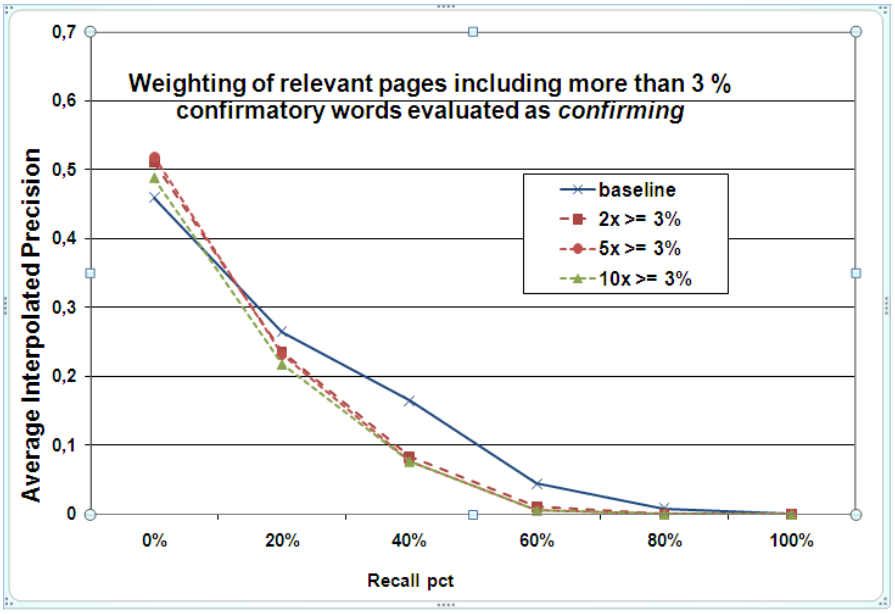
5 Discussion

5.1 Limitation of the current experiments and further research

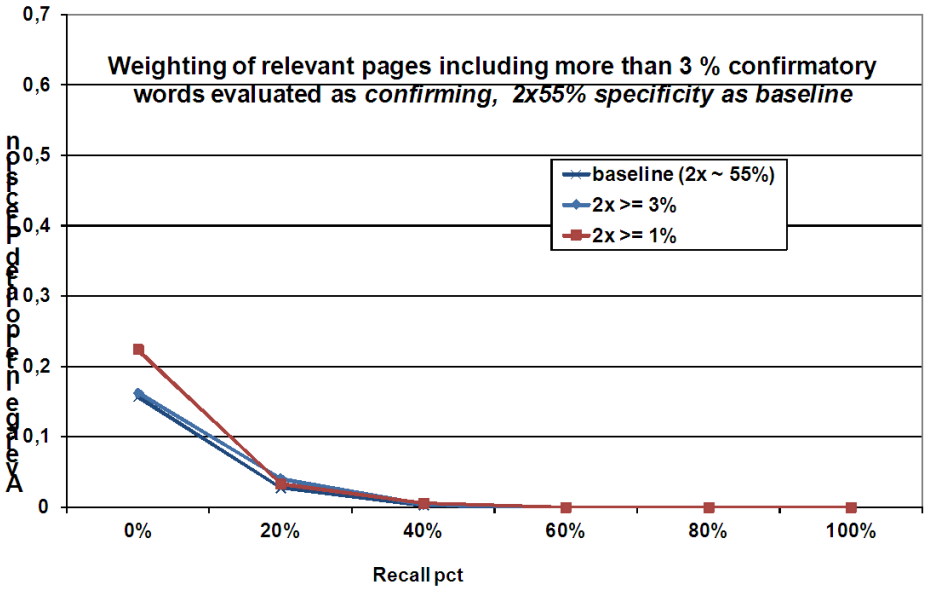
Exploring the semantics of a page in a basically statistic manner may be seen as a superposition of independent components. Counting occurrences of special words is one component on which we superimpose the detection of noun specificity. The treatment using wordnet is further progress from the 2010 experiments, but still rudimentary. Nouns is currently the only word-class we are treating, using only level of specificity. trying to detect classes nouns using the lateral structure of synsets may be another path to follow. It is also conceivable that treating of other word classes, primarily verbs might contribute to the treatment. Verbs are more complicated than nouns in wordnet (ref) and such treatment will be more demanding. Wordnet is not as detailed in treating adjectives and adverbs, so the possible contribution of treating these is questionable.

Utilizing digital books poses new challenges on information retrieval. The mere size of the book text poses both storage, performance and content related

[2] For these, as well as all other plots, We were using the indri combine / weight operation (a combination of weighted expression) with no changes to the default setting (regarding smoothing, a.s.o),



(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 confirming (proving) pages. Baseline marked by solid lines in both subfigures.

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.

This article represents work in progress. We explore techniques gradually in an increasing degree of complexity, trying to adapt and calibrate them.

Even though such activities may be developed and refined using techniques from e.g. Question Answering[2], we suspect that employing semantics-aware retrieval [3,4], which is closely connected to the development of the Semantic Web [5] would be a more viable (and powerful) path to follow.

One obstacle particular to this research is the test collection. Modern ontologies code facts that are closely connected to the modern world. For example the Yago ontology, that codes general facts automatically extracted from Wikipedia, may be complicated to apply to an out-of-copyright book collection emerging from academic specialized environments. But this is certainly a path to follow.

6 Conclusion

This article is a further step in 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 semantics the detection of confirmation (proving) of statement. We also discuss the task of proving statement, raising the question whether it is classifiable as a semantics-aware retrieval task.

References

1. Kazai, G., Koolen, M., Landoni, M.: Summary of the book track. In: INEX 2009
2. VOORHEES, E.M.: The trec question answering track. Natural Language Engineering **7** (2001) 361–378
3. Tim Finin, James Mayfield, A.J.R.S.C., Fink, C.: Information retrieval and the semantic web. In: Proc. 38th Int. Conf. on System Sciences, Digital Documents Track (The Semantic Web: The Goal of Web Intelligence)
4. Mayfield, J., Finin, T.: Information retrieval on the semantic web: Integrating inference and retrieval. In: SIGIR Workshop on the Semantic Web, Toronto
5. Berners-Lee, T., H.J., Lassila, O.: The semantic web. Scientific American (2001)

Overview of the INEX 2011 Data-Centric Track

Qiuyue Wang[1,2], Georgina Ramírez[3], Maarten Marx[4], Martin Theobald[5], Jaap Kamps[6]

[1] School of Information, Renmin University of China, P. R. China
[2] Key Lab of Data Engineering and Knowledge Engineering, MOE, P. R. China
qiuyuew@ruc.edu.cn
[3] Universitat Pompeu Fabra, Spain
georgina.ramirez@upf.edu
[4] University of Amsterdam, the Netherlands
maartenmarx@uva.nl
[5] Max-Planck-Institut für Informatik, Germany
martin.theobald@mpi-inf.mpg.de
[6] University of Amsterdam, the Netherlands
kamps@uva.nl

Abstract. This paper presents an overview of the INEX 2011 Data-Centric Track. Having the *ad hoc search task* running its second year, we introduced a new task, *faceted search task*, which goal is to provide the infrastructure to investigate and evaluate different techniques and strategies of recommending facet-values to aid the user to navigate through a large set of query results and quickly identify the results of interest. The same IMDB collection as last year was used for both tasks. A total of 9 active participants contributed a total of 60 topics for both tasks and submitted 35 ad hoc search runs and 13 faceted search runs. A total of 38 ad hoc search topics were assessed, which include 18 subtopics for 13 faceted search topics. We discuss the setup for both tasks and the results obtained by their participants.

1 Introduction

As the de facto standard for data exchange on the web, XML is widely used in all kinds of applications. XML data used in different applications can be categorized into two broad classes: one is document-centric XML, where the structure is simple and long text fields predominate, e.g. electronic articles, books and so on, and the other is data-centric XML, where the structure is very rich and carries important information about objects and their relationships, e.g. e-Commerce data or data published from databases. The INEX 2011 Data Centric Track is investigating retrieval techniques and related issues over a strongly structured collection of XML documents, the IMDB data collection. With richly structured XML data, we may ask how well such structural information could be utilized to improve the effectiveness of search systems.

The INEX 2011 Data-Centric Track features two tasks: the *ad hoc search task* and the *faceted search task*. The *ad hoc search task* consists of informational requests to be answered by the entities contained in the IMDB collection (movies, actors, directors, etc.); the *faceted search task* asks for a restricted list of facet-values that

will optimally guide the searcher towards relevant information in a ranked list of results, which is especially useful when searchers' information needs are vague or complex.

There were 49 institutes or groups interested in participating in the track, from which 8 (Kasetsart University, Benemérita Universidad Autónoma de Puebla, University of Amsterdam, IRIT, University of Konstanz, Chemnitz University of Technology, Max-Planck Institute for Informatics, Universitat Pompeu Fabra) submitted 45 valid ad hoc search topics and 15 faceted search topics. A total of 9 participants (Kasetsart University, Benemérita Universidad Autónoma de Puebla, University of Amsterdam, IRIT, University of Konstanz, Chemnitz University of Technology, Max-Planck Institute for Informatics, Universitat Pompeu Fabra, Renmin University of China, Peking University) submitted 35 ad hoc search runs and 13 faceted search runs. 38 ad hoc topics were assessed, which included 18 subtopics for 13 faceted search topics.

2 Data Collection

The track uses the cleaned IMDB data collection used in INEX 2010 Data-Centric Track [1]. It was generated from the plain text files published on the IMDB web site on April 10, 2010. There are two kinds of objects in the 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. Each XML document contains information about one object, i.e. a movie or person, with structures conforming to the movie.dtd or person.dtd [1]. In total, the IMDB data collection contains 4,418,081 XML documents, including 1,594,513 movies, 1,872,471 actors, 129,137 directors who did not act in any movie, 178,117 producers who did not direct nor act in any movie, and 643,843 other people involved in movies who did not produce nor direct nor act in any movie.

3 Ad-Hoc Search Task

The task is to return a ranked list of results, i.e. objects, or equivalently documents in the IMDB collection, estimated relevant to the user's information need.

3.1 Topics

Each participating group was asked to create a set of candidate topics, representative of a range of real user needs. Each group had to submit a total of 3 topics, one for each of the categories below:

- **Known-item**: Topics that ask for a particular object (movie or person). Example: "I am searching for the version of the movie 'Titanic' in which the two major characters are called Jack and Rose respectively". For these topics the relevant

answer is a single (or a few) document(s). We will ask participants to submit the file name(s) of the relevant document(s).

- **List**: Topics that ask for a list of objects (movies or persons). For example: "Find movies about drugs that are based on a true story", "Find movies about the era of ancient Rome".
- **Informational**: Topics that ask for information about any topic/movie/person contained in the collection. For example: "Find information about the making of The Lord of the Rings and similar movies", "I want to know more about Ingmar Bergman and the movies she played in".

All the data fields in the IMDB collection can be categorized into three types: categorical (e.g. genre, keyword, director), numerical (e.g. rating, release_date, year), and free-text (e.g. title, plot, trivia, quote). All submitted topics had to involve, at least, one free-text field. The list of all the fields along with their types is given in Appendix 1. We asked participants to submit challenging topics, i.e. topics that could not be easily solved by a current search engine or DB system. Both Content Only (CO) and Content And Structure (CAS) variants of the information need were requested. TopX provided by Martin Theobald was used to facilitate topic development.

After cleaning some duplicates and incorrectly-formed topics, there were a total of 25 valid topics (11 list, 7 known-item, 7 informational). An example of topic is shown in Fig. 1.

*<topic id="**2011105**" guid="**20**">*
*<task>**AdHoc**</task>*
*<type>**Known-Item**</type>*
*<title>**king kong jack black**</title>*
*<castitle>**//movie[about(.//title, king kong) and about(.//actor, jack black)]**</castitle>*
*<description>**I am searching for the version of the movie "King Kong" with the actor Jack Black.**</description>*
*<narrative>**Cause i've heard that this is the best King Kong movie, I am searching for the version of the movie "King Kong", with the actor Jack Black.**</narrative>*
</topic>

Fig. 1. INEX 2011 Data Centric Track Ad Hoc Search Topic 2011105

3.2 Submission Format

Each participant could submit up to 3 runs. Each run could contain a maximum of 1000 results per topic, ordered by decreasing value of relevance. The results of one run had to be contained in one submission file (i.e. up to 3 files could be submitted in total). For relevance assessment and evaluation of the results we required submission files to be in the familiar TREC format:

<qid> Q0 <file> <rank> <rsv> <run_id>

Here:

- The first column is the topic number.
- The second column is the query number within that topic. This is currently unused and should always be Q0.

- The third column is the file name (without .xml) from which a result is retrieved.
- The fourth column is the rank of the result.
- The fifth column shows the score (integer or floating point) that generated the ranking. This score MUST be in descending (non-increasing) order and is important to include so that we can handle tied scores (for a given run) in a uniform fashion (the evaluation routines rank documents from these scores, not from ranks).
- The sixth column is called the "run tag" and should be a unique identifier that identifies the group and the method that produced the run. The run tags must contain 12 or fewer letters and numbers, with NO punctuation, to facilitate labeling graphs with the tags.

An example submission is:

2011001 Q0 9996 1 0.9999 2011UniXRun1
2011001 Q0 9997 2 0.9998 2011UniXRun1
2011001 Q0 person_9989 3 0.9997 2011UniXRun1

Here are three results for topic “2011001”. The first result is the movie from the file 9996.xml. The second result is the movie from the file 9997.xml, and the third result is the person from the file person_9989.xml.

4 Faceted Search Task

Given a vague or broad query, the search system may return a large number of results. Faceted search is a way to help users navigate through the large set of query results to quickly identify the results of interest. It presents the user a list of facet-values to choose from along with the ranked list of results. By choosing from the suggested facet-values, the user can refine the query and thus narrow down the list of candidate results. Then, the system may present a new list of facet-values for the user to further refine the query. The interactive process continues until the user finds the items of interest. The key issue in faceted search is to recommend appropriate facet-values for the user to refine the query and thus quickly identify what he/she really wants in the large set of results. The task aims to investigate and evaluate different techniques and strategies of recommending facet-values to the user at each step in a search session.

4.1 Topics

Each participating group was asked to create a set of candidate topics representative of real user needs. Each topic consists of a general topic as well as a subtopic that refines the general topic by specifying a particular interest of it. The general topic had to result in more than 1000 results, while the subtopics had to be restrictive enough to be satisfied by 10 to 50 results.

Each group had to submit 4 topics: two from the set of general topics given by the organizers, and two proposed by the participants themselves. The given set of general

topics was: {"trained animals", "dogme", "food", "asian cinema", "art house", "silent movies", "second world war", "animation", "nouvelle vague", "wuxia"}.

After removing incorrectly-formed topics, we got a total of 15 general topics along with their 20 subtopics (2 subtopics for "Food", 3 subtopics for "Cannes" and 3 subtopics for "Vietnam"). An example of topic is shown in Fig. 2. The general topic is specified in the <general> field of the <topic> element, while the other fields of <topic>, e.g. <title> and <castitle>, are used to specify the subtopic, which is the searcher's real intention when submitting this general topic to the search system. The participants running the faceted search task could only view the 15 general topics, while the corresponding 20 subtopics were added to the set of topics for the ad hoc search task. The relevance results for these subtopics were used as the relevance results for their corresponding general topics. Thus, altogether we got 45 topics for the ad hoc search task and 15 topics for the faceted search task.

```
<topic id="2011202" guid="28">
  <task>Faceted</task>
  <general>animation</general>
  <title>animation fairy-tale</title>
  <castitle>//movie[about(.//genre, animation) and about(.//plot, fairy-tale)]</castitle>
  <description>I am searching for all animation movies based on a fairy-tale.</description>
  <narrative>I like fairy-tales and their animations remakes.</narrative>
</topic>
```

Fig. 2. INEX 2011 Data Centric Track Faceted Search Topic 2011202

4.2 Submission Format

Each participant had to submit up to 3 runs. A run consists of two files: one is the result file containing a ranked list of maximum 2000 results per topic in the ad hoc search task format, and the other is the recommended facet-value file, which can be a static facet-value file or a dynamic faceted search module.

(1) Facet-Value File. It contains a hierarchy of recommended facet-values for each topic, in which each node is a facet-value and all of its children constitute the newly recommended facet-value list as the searcher selects this facet-value to refine the query. The maximum number of children for each node is restricted to be 20. The submission format is in an XML format conforming to the following DTD.

```
<!ELEMENT run     (topic+)>
<!ATTLIST run     rid     ID      #REQUIRED>
<!ELEMENT topic   (fv+)>
<!ATTLIST topic   tid     ID      #REQUIRED>
<!ELEMENT fv      (fv*)>
<!ATTLIST fv      f       CDATA #REQUIRED
                  v       CDATA #REQUIRED>
```

Here:

- The root element is <run>, which has an ID type attribute, *rid*, representing the unique identifier of the run. It must be identical with that in the result file of the same run.

- The <run> contains one or more <topic>s. The ID type attribute, *tid*, in each <topic> gives the topic number.
- Each <topic> has a hierarchy of <fv>s. Each <fv> shows a facet-value pair, with *f* attribute being the facet and *v* attribute being the value. The facet is expressed as an XPath expression. The set of all the possible facets represented as XPath expressions in the IMDB data collection can be found in Appendix 1. We allow only categorical or numerical fields to be possible facets. Free-text fields are not considered. Each facet-value pair represents a facet-value condition to refine the query. For example, <fv f="/movie/overview/directors/director" v="Yimou Zhang"> represents the condition /movie/overview/directors/director="Yimou Zhang".
- The <fv>s can be nested to form a hierarchy of facet-values.

An example submission is:

```
<run rid="2011UniXRun1">
  <topic tid="2011001">
    <fv f="/movie/overview/directors/director" v="Yimou Zhang">
      <fv f="/movie/cast/actors/actor/name" v="Li Gong">
        <fv f="/movie/overview/releasedates/releasedate" v="2002"/>
        <fv f="/movie/overview/releasedates/releasedate" v="2003"/>
      </fv>
      <fv f="/movie/cast/actors/actor/name" v="Ziyi Zhang">
        <fv f="/movie/overview/releasedates/releasedate" v="2005"/>
      </fv>
    </fv>
    …
  </topic>
  <topic tid="2011002">
      ...
  </topic>
…
</run>
```

Here for the topic "2011001", the faceted search system first recommends the facet-value condition /movie/overview/directors/director="Yimou Zhang" among other facet-value conditions, which are on the same level of the hierarchy. If the user selects this condition to refine the query, the system will recommend a new list of facet-value conditions, which are /movie/cast/actors/actor/name="Li Gong" and /movie/cast/actors/actor/name="Ziyi Zhang", for the user to choose from to further refine the query. If the user then selects /movie/cast/actors/actor/name="Li Gong", the system will recommend /movie/overview/releasedates/releasedate="2002" and /movie/overview/releasedates/releasedate ="2003". Note that the facet-value conditions that are selected to refine the query form a path in the tree, e.g. /movie/overview/directors/director="Yimou Zhang" → /movie/cast/actors/actor/name = "Li Gong" → /movie/overview/releasedates/releasedate ="2003". It is required that no facet-value condition occurs twice on any path.

(2) Faceted Search Module. Instead of submitting a static hierarchy of facet-values, participants are given the freedom to dynamically generate lists of recommended facet-values and even change the ranking order of the candidate result list at each step in the search session. This is achieved by submitting a self-implemented dynamically linkable module, called Faceted Search Module (FSM). It implements the *FacetedSearchInterface* defined as the following:

```
public interface FacetedSearchInterface {
   public String[] openQuery(String topicID, String[] resultList);
   public String[] selectFV(String facet, String value, String[] selectedFV);
   public String[] refineQuery(String facet, String value, String[] selectedFV);
   public String[] expandFacet(String facet, String[] selectedFV);
   public void closeQuery(String topicID);
}

public class FacetedSearch implements FacetedSearchInterface {
    // to be implemented by the participant
}
```

The User Simulation System (USS) used in evaluation will interact with the FSM to simulate a faceted search session. The USS starts to evaluate a run by instantiating a FacetedSearch object. For each topic to be evaluated, the USS first invokes openQuery() method to initialize the object with the topic id and initial result list for this topic. The result list is actually the list of retrieved file names (without .xml) in the third column of the result file. The method would return a list of recommended facet-values for the initial result list. A facet-value is encoded into a String in the format "<facet>::<value>", for example, "/movie/overview/directors/director::Yimou Zhang".

After opening a query, the USS then simulates a user's behavior in a faceted search system based on some user model as described in Section 5. When the simulated user selects a facet-value to refine the query, the selectFV() method would be called to return a new list of recommended facet-values; and the refineQuery() method would be called to return a list of candidate results in the initial result list that satisfy all the selected facet-value conditions. The inputs to both methods are the currently selected facet and value, as well as a list of previously selected facet-values. A facet-vaue pair is encoded into a String in the format shown above.

If the user could not find a relevant facet-value to refine the query in the recommended list, he/she could probably expand the facet-value list by choosing a facet among all possible facets, examine all its possible values and then select one to refine the query. In such a case, the USS invokes the expandFacet() method with the name of the facet to be expanded as well as a list of previously selected facet-values as input and the list of all possible values of this facet as output. Observe that in the specification of FacetedSearchInterface, we do not restrict facet-value comparisons to be of equality, but can be of any other possible semantics since the interpretation of facet-value conditions is capsulated into the implementation of FacetedSearchInterface. Thus, given the same facet, different systems may give different sets of all possible values depending on if they will cluster and how they will cluster some values.

When the search session of a query ends, the closeQuery() method is invoked. The FacetedSearch object will be used as a persistent object over the entire evaluation of a run. That is, different topics in the same run will be evaluated using the same FacetedSearch object. But different runs may have different implementations of the FacetedSearch class.

5 Assessments and Evaluations

In total 35 ad hoc search runs and 13 faceted search runs were submitted by 9 active participants. Assessment was done using the same assessment tool as that used in INEX 2010 Data-Centric Track provided by Shlomo Geva. 38 ad hoc topics among 45 ones were assessed by those groups that submitted runs. Among the assessed topics, there are 9 list type topics, 6 known-item type topics, 5 informational type topics, and 18 subtopics for 13 faceted search topics.

Table 1 shows the mapping between the subtopics in ad hoc search task and the general topics in faceted search task. The relevance results of subtopics are treated as the intended results for their corresponding general topics. Note that some general topics, e.g. 2011205, 2011207 and 2011210, have more than one intention/subtopic. For these general topics, we take the subtopics that have the least number of relevance results. For example, compared with topic 2011120 and 2011142, topic 2011141 has the least number of relevance results, whose relevance results are then chosen as the relevance results for topic 2011205. The chosen subtopics are underlined in Table 1. Since the subtopics 2011121 and 2011139 were not assessed, we have no relevance results for topics 2011206 and 2011215 in the faceted search task.

Table 1. Mapping between the faceted search topics and subtopics in ad hoc task.

General Topics	Subtopics
2011201	2011111
2011202	2011114
2011203	2011118
2011204	2011119
2011205	2011120,2011141,2011142
2011206	2011121
2011207	2011112,2011140
2011208	2011129
2011209	2011130
2011210	2011135,2011144,2011145
2011211	2011143
2011212	2011136
2011213	2011137
2011214	2011138
2011215	2011139

The TREC MAP metric, as well as P@5, P@10, P@20 and so on, was used to measure the performance of all ad hoc runs at whole document retrieval.

For the faceted search task, since it is the first year, we used the following two types of evaluation approaches and metrics to gain better understanding to the problem.

- **NDCG of facet-values:** The relevance of the hierarchy of recommended facet-values is evaluated based on the relevance of the data covered by these facet-values, measured by NDCG. The details of this evaluation methodology are given in [2].

- **Interaction cost:** The effectiveness of a faceted search system is evaluated by measuring the interaction cost or the amount of efforts spent by a user in meeting his/her information needs. To avoid expensive user study and make the evaluation repeatable, we applied user simulation methodology like that used in [3, 4] to measure the costs.

We can use two metrics to measure the user's interaction cost. One is the number of results, facets or facet-values that the user examined before he/she encounters the first relevant result, which is similar to the Reciprocal Rank metric in traditional IR. Here we assume that the effort spent on examining each facet or facet-value is the same as that spent on examining each result. The other is the number of actions that the user performs in the search session. We only consider the click actions.

As in [3, 4], we assume that the user will end the search session when he/she encounters the first relevant result, and the user can recognize the relevant results from the list of results, and can distinguish the relevant facets or facet-values that match at least one relevant result from the list of facets or facet-values.

The user begins by examining the first page of the result list for the current query. It is assumed that each page displays at most 10 results. If the user finds relevant results on the first page, the user selects the first one and ends the session. If no relevant result is found, the user then examines the list of recommended facet-values. If there are relevant facet-values, the user then clicks on the first relevant facet-value in the list to refine the query, and the system returns the new lists of results and facet-values for the refined query. If none of the recommended facet-values is relevant, the user chooses the first relevant facet in the list of all possible facets to expand and select the first relevant value in this facet's value list to refine the query. If the user does not find any relevant facet to expand, the user begins to scan through the result list and stops at the first relevant result encountered. Fig. 3 shows the flowchart of the user interaction model and cost model used in the evaluation. Notation used in Fig. 3 is given in Table 2.

Table 2. Notation used in Fig. 3.

Symbol	Meaning
q	The current query
R_q	The result list of query q
FV_q	The list of recommended facet-values for query q
F_q	The list of all possible facets for query q
loc(x,y)	A function returns the position of item x in the list y
cost	The number of results, facet-values or facets examined by the user
actionCount	The number of click actions performed by the user

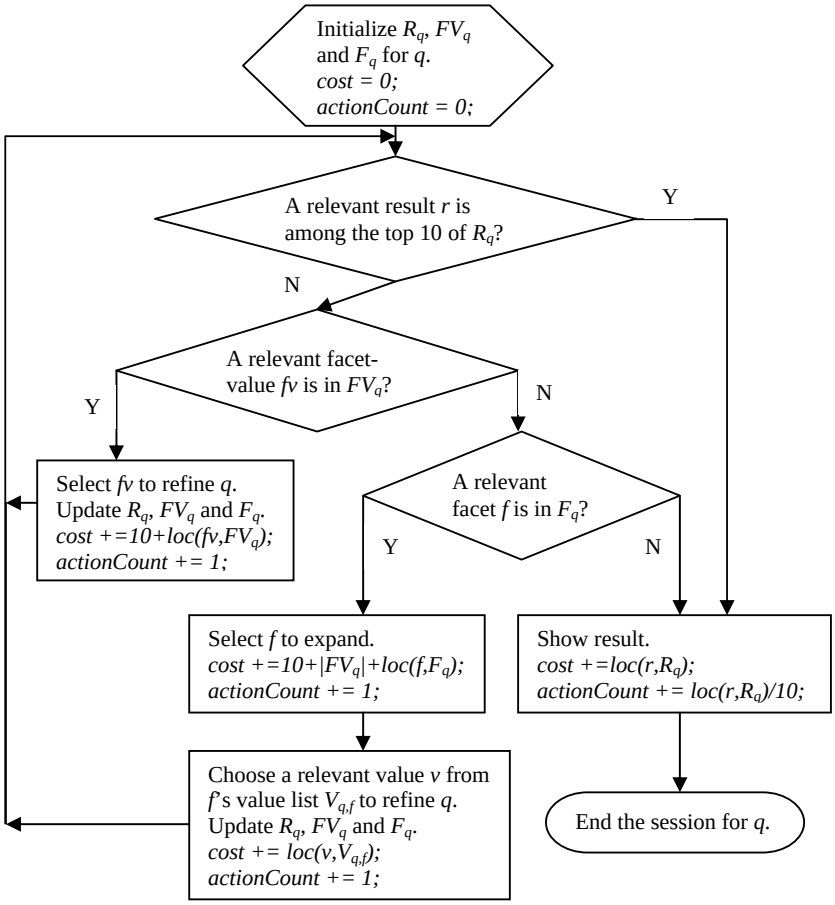


Fig. 3. Flowchart of the Simulated User Interaction Model with Faceted Search System

6 Results

6.1 Ad Hoc Search Results

As mentioned above, a total of 35 runs from 9 different institutes were submitted to the ad hoc search task. This section presents the evaluation results for these runs. Results were computed over the 38 topics assessed by the participants using the TREC evaluation tool. The topic set is a mixture of informational, known-item, list, and faceted (sub)topics. We use MAP as the main measure since it averages reasonably well over such a mix of topic types.

Table 3 shows an overview of the 10 best performing runs for this track. Over all topics, the best scoring run is from the University of Amsterdam with a MAP of 0.3969. Second best scoring team is Renmin University of China (0.3829). Third best scoring team is Kasetsart University (0.3479) with the highest score on mean reciprocal rank (1/rank). Fourth best team is Peking University (0.3113) and the highest precision at 10. Fifth best team is Universitat Pompeu Fabra, with a MAP of 0.2696 but the highest scores for precision at 20 and 30.

Table 3. Best performing runs (only showing one run per group) based on MAP over all ad hoc topics.

Run	map	1/rank	P@10	P@20	P@30
p4-UAms2011adhoc	**0.3969**	0.6991	0.4263	0.3921	0.3579
p2-ruc11AS2	0.3829	0.6441	0.4132	0.3842	0.3684
p16-kas16-MEXIR-2-EXT-NSW	0.3479	**0.6999**	0.4316	0.3645	0.3298
p77-PKUSIGMA01CLOUD	0.3113	0.5801	**0.4421**	0.4066	0.3851
p18-UPFbaseCO2i015	0.2696	0.5723	0.4342	**0.4171**	**0.3825**
p30-2011CUTxRun2	0.2099	0.6104	0.3684	0.3211	0.2965
p48-MPII-TOPX-2.0-co	0.1964	0.5698	0.3684	0.3395	0.3289
p47-FCC-BUAP-R1	0.1479	0.5120	0.3474	0.2763	0.2412
p12-IRIT_focus_mergeddtd_04	0.0801	0.2317	0.2026	0.1724	0.1702

Interpolated precision against recall is plotted in Fig. 4, showing quite solid performance for the better scoring runs.

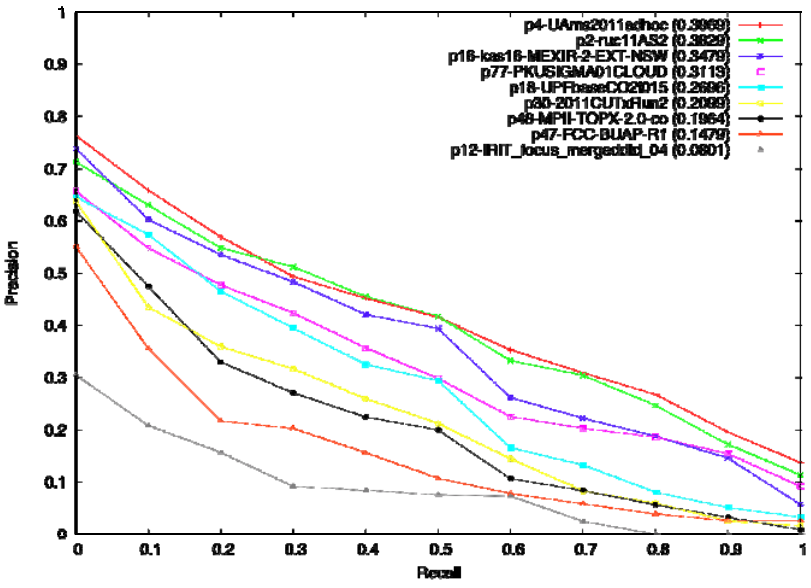


Fig. 4. Best run by each participating institute measured with MAP

Breakdown over Topic Types

In this section, we will analyze the effectiveness of the runs for each of the four topic types. Let us first analyze the topics and resulting judgments in more details. Table 4 lists the topics per topic type, and Table 5 lists statistics about the number of relevant entities.

Table 4. Breakdown over Topic Types

Topic Type	Topics created	Topics Judged	Topics with relevance
Informational	7	5	5
Known-Item	7	6	6
List	11	9	8
Faceted subtopics	20	18	18
All	45	38	37

Table 5. Relevance per Topic Type

Topic Type	Topics	Min	Max	Median	Mean	Std.	Total
Informational	5	6	327	40	125.8	150.4	629
Known-Item	6	1	416	2	71.3	168.9	428
List	8	5	299	32	98.6	118.1	789
Faceted subtopics	18	23	452	148	168.3	123.8	3,029
All	37	1	452	72	168.3	134.0	4,875

What we see in Table 4 is that we have 5 (informational) to 18 (faceted sub-) topics judged for each type. Given the small number of topics per type, one should be careful with drawing final conclusions based on the analysis, since the particular choice of topics may have had a considerable influence on the outcome.

While all topics have been judged “as is” without special instructions for each of the topic types, the statistics of the relevance judgments in Table 5 is confirming the differences between these topic types. The known-item topics have a median of 2 relevant documents, the list topics have a median of 32 relevant documents, and the informational topics have a median of 40. The faceted (sub)topics, which were based on a general seed topic, have even a median of 148 relevant documents. For all topic types the distribution over topics is skewed, and notable exceptions exist, e.g. a known-item topic with 416 relevant documents.

Table 6 shows the results over only the informational topics. We see that Kasetsart (0.3564), Chemnitz (0.3449), and BUAP (0.3219) now have the best scores, and that there are less differences in scores amongst the top 5 or 6 teams. Over all 34 submissions the system rank correlation (Kendall’s tau) with the ranking over all topics is moderate with 0.512.

Table 6. Best performing runs (only showing one run per group) based on MAP over the 5 informational ad hoc topics.

run	map	1/rank	P@10	P@20	P@30
p16-kas16-MEXIR-2-EXT-NSW	**0.3564**	0.8000	0.5000	0.4200	0.3600
p30-2011CUTxRun2	0.3449	0.7067	0.5000	**0.4700**	0.4333
p47-FCC-BUAP-R1	0.3219	**1.0000**	**0.5600**	0.4300	0.4133
p2-ruc11AMS	0.3189	0.6500	0.4200	0.4500	**0.4600**
p4-UAms2011adhoc	0.3079	0.6750	0.3800	0.3100	0.2600
p18-UPFbaseCO2i015	0.2576	0.6346	0.4600	0.4400	0.3800
p77-PKUSIGMA02CLOUD	0.2118	0.5015	0.4400	0.4200	0.3133
p48-MPII-TOPX-2.0-co	0.0900	0.3890	0.2600	0.1800	0.2000
p12-IRIT_focus_mergeddtd_04	0.0366	0.3022	0.2200	0.1100	0.0733

Table 7 shows the results over only the known-item topics, now evaluated by the mean reciprocal rank (1/rank). We observe that Amsterdam (0.9167), Renmin (also 0.9167), and MPI (0.7222). Hence the best teams over all topics score also well over the known-item topics. This is no surprise since the known-item topics tend to lead to relatively higher scores, and hence have a relatively large impact. Over all 34 submissions the system rank correlation based on MAP is 0.572.

Table 7. Best performing runs (only showing one run per group) based on 1/rank over the 6 known-item ad hoc topics.

run	map	1/rank	P@10	P@20	P@30
p4-UAms2011adhoc	**0.8112**	**0.9167**	**0.3167**	**0.2417**	**0.2167**
p2-ruc11AS2	0.7264	**0.9167**	**0.3167**	**0.2417**	**0.2167**
p48-MPII-TOPX-2.0-co	0.2916	0.7222	0.2333	0.1833	0.1778
p18-UPFbaseCO2i015	0.3752	0.7104	0.2500	0.2083	0.1944
p16-kas16-MEXIR-2-EXT-NSW	0.4745	0.6667	0.0833	0.0417	0.0278
p77-PKUSIGMA01CLOUD	0.5492	0.6389	**0.3167**	**0.2417**	**0.2167**
p30-2011CUTxRun2	0.3100	0.5730	0.2667	0.1750	0.1667
p47-FCC-BUAP-R1	0.2500	0.3333	0.0333	0.0167	0.0111
p12-IRIT_large_nodtd_06	0.0221	0.0487	0.0167	0.0333	0.0222

Table 8 shows the results over the list topics, now again evaluated by MAP. We see the best scores for Kasetsart (0.4251), Amsterdam (0.3454), and Peking University (0.3332). The run from Kasetsart outperforms all other runs on all measures for the list topics. Over all 34 submissions the system rank correlation is 0.672.

Table 8. Best performing runs (only showing one run per group) based on MAP over the 8 list ad hoc topics.

run	map	1/rank	P@10	P@20	P@30
p16-kas16-MEXIR-2-EXT-NSW	**0.4251**	**0.7778**	**0.4778**	**0.3833**	**0.3741**
p4-UAms2011adhoc	0.3454	0.6674	0.4222	0.3500	0.3222
p77-PKUSIGMA02CLOUD	0.3332	0.5432	0.3889	0.3667	0.3481
p2-ruc11AS2	0.3264	0.6488	0.4111	0.3333	0.2963
p48-MPII-TOPX-2.0-co	0.2578	0.4926	0.3000	0.3333	0.3259
p18-UPFbaseCO2i015	0.2242	0.5756	0.3556	0.3278	0.2741

p12-IRIT_focus_mergeddtd_04	0.1532	0.2542	0.2333	0.2111	0.2148
p30-2011CUTxRun3	0.0847	0.5027	0.1889	0.1611	0.1667
p47-FCC-BUAP-R1	0.0798	0.3902	0.2889	0.2500	0.2259

Table 9 shows the results over the faceted search subtopics (each topic covering only a single aspect). We see the best performance in the runs from Renmin (0.3258), Amsterdam (0.3093), and Peking University (0.3026), with Peking University having clearly the best precision scores. Given that 18 of the 37 topics are in this category, the ranking corresponds reasonably to the ranking over all topics. Over all 34 submissions the system rank correlation is high with 0.818.

Table 9. Best performing runs (only showing one run per group) based on MAP over the 18 facted ad hoc topics.

run	map	1/rank	P@10	P@20	P@30
p2-ruc11AS2	**0.3258**	0.5585	0.4722	0.4778	0.4722
p4-UAms2011adhoc	0.3093	0.6492	0.4778	0.4861	0.4500
p77-PKUSIGMA02CLOUD	0.3026	**0.7400**	**0.5722**	**0.5361**	**0.5315**
p16-kas16-MEXIR-2-EXT-NSW	0.2647	0.6443	0.5056	0.4472	0.4000
p18-UPFbaseCO2i015	0.2605	0.5072	0.5278	0.5250	0.5000
p30-2011CUTxRun2	0.2130	0.6941	0.4611	0.4083	0.3741
p48-MPII-TOPX-2.0-co	0.1635	0.6078	0.4778	0.4389	0.4167
p47-FCC-BUAP-R1	0.0995	0.4969	0.4222	0.3333	0.2778
p12-IRIT_focus_mergeddtd_04	0.0810	0.2754	0.2500	0.2278	0.2296

6.2 Faceted Search Results

In the faceted search task, 5 groups, University of Amsterdam (Jaap), Max-Plank Institute, University of Amsterdam (Maarten), Universitat Pompeu Fabra, and Renmin University of China, submitted 12 valid runs. All runs are in the format of static hierarchy of facet-values except that one run from Renmin is in the format of a self-implemented faceted search module. So we only present the evaluation results for the 11 static runs. Most of the runs are based on the reference result file provided by Anne Schuth, who generated the reference result file using XPath and Lucene. Two runs from Amsterdam (Jaap) are based on a result file generated by Indri and one run from Max-Plank Institute is based on the result file generated by TopX.

13 out of 15 general topics have relevance results. Table 10 shows, for each topic, the number of relevant results, and the rank of the first relevant result in the three result lists generated by Indri, Lucene and TopX respectively, which is in fact the cost that users sequentially scan through the list of results to find the first relevant answer without using the faceted-search facility. We call it raw cost, which is actually equal to 1/RR. "-" means that the result file contains no relevant result for this topic. It can be observed that the Indri result file contains relevant results for all topics and ranks them quite high. The TopX result file ranks the first relevant results for 7 topics highest among the three result files, but it fails in containing relevant results for 3 topics. The Lucene reference result file, however, is the worst one.

Table 10. Raw costs (1/RR) of faceted search topics on 3 different result files.

Topic ID	Number of relevant results	Raw cost of Indri result file	Raw cost of Lucene result file	Raw cost of TopX result file
2011201	48	**45**	-	97
2011202	327	**11**	19	85
2011203	138	**114**	451	-
2011204	342	**306**	989	-
2011205	141	9	316	**1**
2011207	23	69	850	**44**
2011208	285	2	11	**1**
2011209	76	**1**	2	**1**
2011210	23	217	-	**49**
2011211	72	61	45	**40**
2011212	156	1110	-	**344**
2011213	35	828	-	-
2011214	176	**4**	44	16

We use two metrics to evaluate the effectiveness of recommended facet-values by each run. One is the interaction cost based on a simple user simulation model, and the other is the NDCG of facet-values [2].

As described in Section 5, the interaction cost is defined as the number of results, facets or facet-values that the user examined before he/she encounters the first relevant result. This cost can be compared with the raw cost, which is the number of results sequentially examined in the result list without using faceted search facility, to see if the faceted search facility is effective or not. We name their difference as the *Gain* of faceted search. To compare systems across multiple topics, we define the *Normalized Gain (NG)* and *Average Normalized Gain (ANG)* as the following. Note that *NG* is a number between 0 and 1.

$$NG = \max(0, (rawCost - Cost) / rawCost) \quad (1)$$

$$ANG = \frac{1}{|Q|} \sum_{i=1}^{|Q|} NG_i \quad (2)$$

Table 11 shows the evaluation results for all the 11 runs in terms of *NG* and *ANG*. Two runs from Amsterdam (Jaap), p4-UAms2011indri-c-cnt and p4-UAms2011indri-cNO-scr2, are based on the Indri result file, and p48-MPII-TOPX-2.0-facet-entropy (TopX) from Max-Plank is based on the TopX result file. All the other 8 runs are based on the Lucene result file. Because the Indri result file is superior to the TopX and Lucene result files, the two runs based on it perform also better than other runs, and the best one is p4-UAms2011indri-cNO-scr2 (0.35). Among all the 8 runs based on the Lucene result file, p2-2011Simple1Run1 (0.33) from Renmin performs best in terms of ANG. It is followed by p4-UAms2011Lucene-cNO-lth (0.24) from Amsterdam (Jaap), p18-2011UPFfixGDAh2 (0.21) from Universitat Pompeu Fabra and p4-2011IlpsNumdoc (0.20) from Amsterdam (Maarten).

The NDCG scores calculated using the method described in [2] for all 11 static runs are listed in Table 12. For *p* we chose 10 (we thus consider the top 10 documents per facet-value) and we also limited the number of facet-values to be evaluated to 10. Note that we did not evaluate the runs using NRDCG.

Table 11. Evaluation results of all static runs in terms of NGs and ANG.

run / NG	p4-UAms2011indri-c-cnt	p4-UAms2011indri-cNO-scr2	p4-UAms2011lucene-cNO-lth	p48-MPII-TOPX-2.0-facet-entropy (TopX)	p48-MPII-TOPX-2.0-facet-entropy (Lucene)	p4-2011llpsFtScore	p4-2011llpsNumdoc	p18-2011UPFfixG7DAnh	p18-2011UPFfixGDAh	p18-2011UPFfixGDAh2	p2-2011Simple1Run1
201	0.64	0.60	-	0	-	-	-	-	-	-	-
202	0	0	0.21	0	0	0	0.21	0.11	0.11	0.11	0.21
203	0	0	0	-	0	0.83	0.82	0	0	0	0.86
204	0.63	0.75	0.94	-	0	0	0.90	0	0	0.98	0.91
205	0	0	0.81	0	0.75	0.79	0.72	0.95	0.95	0.95	0.81
207	0	0.77	0	0	0	0	0	0	0	0	0.94
208	0	0	0	0	0	0	0	0	0	0	0
209	0	0	0	0	0	0	0	0	0	0	0
210	0.75	0.74	-	0	-	-	-	-	-	-	-
211	0.18	0	0.53	0	0	0	0	0.71	0.71	0.71	0.60
212	0.89	0.88	-	0	-	-	-	-	-	-	-
213	0.76	0.76	-	-	-	-	-	-	-	-	-
214	0	0	0.64	-	0	0	0	0	0.09	0	0
ANG	0.30	**0.35**	0.24	0	0.06	0.12	0.20	0.14	0.14	0.21	**0.33**

Table 12. Evaluation results for the 11 statics runs in terms of NDCG. Results are per topic and the mean over all topics. Highest scores per topic are highlighted.

run / NDCG	p4-UAms2011indri-c-cnt	p4-UAms2011indri-cNO-scr2	p4-UAms2011lucene-cNO-lth	p48-MPII-TOPX-2.0-facet-entropy (TopX)	p48-MPII-TOPX-2.0-facet-entropy (Lucene)	p4-2011llpsFtScore	p4-2011llpsNumdoc	p18-2011UPFfixG7DAnh	p18-2011UPFfixGDAh	p18-2011UPFfixGDAh2	p2-2011Simple1Run1
201	**0.03**	**0.03**	0	0	0	0	0	0	0	0	0
202	0	0	0	0	0	0	0	0	0	0	0
203	0	0	0	0	0	0	0	0	0	0	0
204	0	0	0	0	0	0	0	0	0	0	0
205	0	**0.43**	0.21	0	0	0	0.13	0	0	0	0.07
207	0	**0.16**	0	0	0	0	0	0	0	0	0
208	0	**0.45**	0	0	0	0	0	0	0	0	0
209	0	0	0	0	0	0	**0.24**	0	0	0	0
210	0	0	0	0	0	0	0	0	0	0	0
211	0	0	0	0	0	0	0	0	0	0	0
212	0	0	0	0	0	0	0	0	0	0	0
213	0	0	0	0	0	0	0	0	0	0	0
214	**0.18**	**0.18**	0.09	0	0	0	0	0	0	0	0
mean	0.02	**0.10**	0.02	0	0	0	0.03	0	0	0	0.01

Note that the NDCG calculation used the union of relevance judgments in case there were multiple subtopics for a topic. Statistics for the relevance judgments used for the NDCG evaluation are listed in Table 13.

Table 13. Relevance judgments for faceted search topics.

Topic Type	Topics	Min	Max	Median	Mean	Std.	Total
Faceted	13	35	774	156	233	229.3	3029

7 Conclusions and Future Work

We presented an overview of the INEX 2011 Data-Centric Track. This track has successfully run its second year and has introduced a new task, the *faceted search* task. The IMDB collection has now a good set of assessed topics that can be further used for research on richly structured data. Our plan for next year is to extend this collection with related ones such as DBpedia and Wikipedia in order to reproduce a more realistic scenario for the newly introduced faceted search task.

Acknowledgements

Thanks are given to the participants who submitted the topics, runs, and performed the assessment process. Special thanks go to Shlomo Geva for porting the assessment tools, to Anne Schuth, Yu Sun and Yantao Gan for evaluating the faceted search runs, and to Ralf Schenkel for administering the web site.

References

1. A. Trotman, Q. Wang, Overview of the INEX 2010 Data Centric Track, INEX 2010.
2. A. Schuth, M.J. Marx, Evaluation Methods for Rankings of Facetvalues for Faceted Search, Proceedings of the Conference on Multilingual and Multimodal Information Access Evaluation 2011.
3. J. Koren, Y. Zhang, X. Liu, Personalized Interactive Faceted Search, WWW 2008.
4. A. Kashyap, V. Hristidis, M. Petropoulos, FACeTOR: Cost-Driven Exploration of Faceted Query Results, CIKM 2010.

Appendix 1: All the Fields or Facets in IMDB Collection

Field Type	Field (or Facet) expressed in XPath
free-text	/movie/title
numerical	/movie/overview/rating
categorical	/movie/overview/directors/director
categorical	/movie/overview/writers/writer
numerical	/movie/overview/releasedates/releasedate
categorical	/movie/overview/genres/genre
free-text	/movie/overview/tagline

free-text	/movie/overview/plot
categorical	/movie/overview/keywords/keyword
categorical	/movie/cast/actors/actor/name
categorical	/movie/cast/actors/actor/character
categorical	/movie/cast/composers/composer
categorical	/movie/cast/editors/editor
categorical	/movie/cast/cinematographers/cinematographer
categorical	/movie/cast/producers/producer
categorical	/movie/cast/production_designers/production_designer
categorical	/movie/cast/costume_designers/costume_designer
categorical	/movie/cast/miscellaneous/person
free-text	/movie/additional_details/aliases/alias
categorical	/movie/additional_details/mpaa
numerical	/movie/additional_details/runtime
categorical	/movie/additional_details/countries/country
categorical	/movie/additional_details/languages/language
categorical	/movie/additional_details/colors/color
categorical	/movie/additional_details/certifications/certification
categorical	/movie/additional_details/locations/location
categorical	/movie/additional_details/companies/company
categorical	/movie/additional_details/distributors/distributor
free-text	/movie/fun_stuff/trivias/trivia
free-text	/movie/fun_stuff/goofs/goof
free-text	/movie/fun_stuff/quotes/quote
categorical	/person/name
categorical	/person/overview/birth_name
numerical	/person/overview/birth_date
numerical	/person/overview/death_date
numerical	/person/overview/height
categorical	/person/overview/spouse
free-text	/person/overview/trademark
free-text	/person/overview/biographies/biography
categorical	/person/overview/nicknames/name
free-text	/person/overview/trivias/trivia
free-text	/person/overview/personal_quotes/quote
free-text	/person/overview/where_are_they_now/where
categorical	/person/overview/alternate_names/name
numerical	/person/overview/salaries/salary
free-text	/person/filmography/act/movie/title
numerical	/person/filmography/act/movie/year
categorical	/person/filmography/act/movie/character
free-text	/person/filmography/direct/movie/title
numerical	/person/filmography/direct/movie/year
categorical	/person/filmography/direct/movie/character
free-text	/person/filmography/write/movie/title
numerical	/person/filmography/write/movie/year
categorical	/person/filmography/write/movie/character
free-text	/person/filmography/compose/movie/title
numerical	/person/filmography/compose/movie/year
categorical	/person/filmography/compose/movie/character
free-text	/person/filmography/edit/movie/title
numerical	/person/filmography/edit/movie/year
categorical	/person/filmography/edit/movie/character
free-text	/person/filmography/produce/movie/title
numerical	/person/filmography/produce/movie/year
categorical	/person/filmography/produce/movie/character
free-text	/person/filmography/production_design/movie/title
numerical	/person/filmography/production_design/movie/year
categorical	/person/filmography/production_design/movie/character

free-text	/person/filmography/cinematograph/movie/title
numerical	/person/filmography/cinematograph/movie/year
categorical	/person/filmography/cinematograph/movie/character
free-text	/person/filmography/costume_design/movie/title
numerical	/person/filmography/costume_design/movie/year
categorical	/person/filmography/costume_design/movie/character
free-text	/person/filmography/miscellaneous/movie/title
numerical	/person/filmography/miscellaneous/movie/year
categorical	/person/filmography/miscellaneous/movie/character
free-text	/person/additional_details/otherworks/otherwork
free-text	/person/additional_details/public_listings/interviews/interview
free-text	/person/additional_details/public_listings/articles/article
free-text	/person/additional_details/public_listings/biography_prints/print
free-text	/person/additional_details/public_listings/biographical_movies/biographical_movie
free-text	/person/additional_details/public_listings/portrayed_ins/portrayed_in
free-text	/person/additional_details/public_listings/magazine_cover_photos/magazine
free-text	/person/additional_details/public_listings/pictorials/pictorial

Edit Distance for XML Information Retrieval : Some experiments on the Datacentric track of INEX 2011

Cyril Laitang, Karen Pinel-Sauvagnat, and Mohand Boughanem

IRIT-SIG,
118 route de Narbonne,
31062 Toulouse Cedex 9,
France

Abstract. In this paper we present our structured information retrieval model based on subgraphs similarity. Our approach combines a content propagation technique which handles sibling relationships with a document query matching process on structure. The latter is based on tree edit distance which is the minimum set of insert, delete, and replace operations to turn one tree to another. As the effectiveness of tree edit distance relies both on the input tree and the edit costs, we experimented various subtree extraction techniques as well as different costs based on the DTD associated to the Datacentric collection.

1 Introduction

XML documents could be naturally represented through trees in which nodes are elements and edges hierarchical dependencies. Similarly structural constraints of CAS queries can be expressed though trees. Based on these common representations we propose a SIR model using both graph theory properties and content scoring. Figure 1 shows a conversion example of an XML document and a CAS query expressed in the *Narrowed Extended XPath* (NEXI) language. The target element is "m". For clarity reasons we shorten the tags. In real case the "m" is equivalent to "movie". This document and query will be used all along this article to illustrate the different steps of our algorithms.

The rest of this paper is organized as follows: Section 2 presents work related to the different steps of our approach; Section 3 presents state of the art approaches and finally Section 4 discusses the results obtained over the Datacentric track of INEX 2011 for each of our approaches.

2 Related Works

In this section we will first overview document structure representation and extraction techniques and then give a brief survey on tree edit distance algorithms.

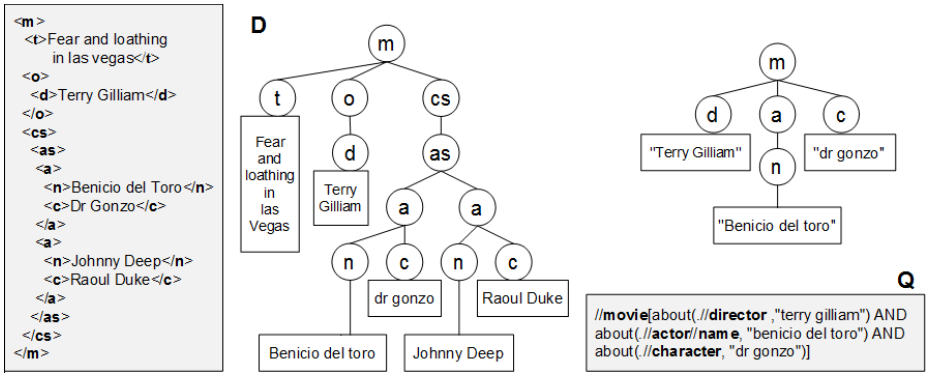


Fig. 1. Tree representation of an XML document and a query in which *we want a "movie" directed by "Terry Gilliam"' with the actor "Benicio del Toro" and with a character named "Dr Gonzo".*

2.1 Document structure representation and extraction

In the litterature we identify two families of approaches regarding how to handle document structure regardless of content. The first one is relaxation. In these approaches the main structure is atomized into a set of node-node relationships. The weight of these relationships is then a representation of the distance between nodes in the original structure. These relationships could then be used with a language model [2]. The second family of approaches is linked to subtree extraction. The most widely used is the *lowest common ancestor*. The LCA is the tree rooted by the first common ancestor of two or more selected nodes [3]. In information retrieval it aims at finding the subtrees in which all the leaves contains at least on term of the query [1]. The root node of that particular subtree is then considered as a candidate to return in answer to the query.

2.2 Edit distance

Two graphs are called isomorphic if they share the same nodes and edges. Evaluating how isomorphic they are is called graph matching. We make the distinction between approximate matching and exact matching. The first one attempts to find a degree of similarity between two structures while exact matching tries to validate the similarity. Because of the context of our work, we will focus here on approximate tree matching. There are three main families of approximate tree matching: *edit distance*, *alignment* and *inclusion*. As tree edit distance offers the most general application we will focus on this one. Tree edit distance algorithms [13] generalize Levenshtein *edit distance*[9] to trees. The similarity is the minimal set of operations (adding, removing and relabeling) to turn one tree to another. Given two forests (set of trees) F and G, Γ_F and Γ_G their rightmost nodes, $T(\Gamma_F)$ the tree rooted in Γ_F and the cost functions $c_{del}()$ and $c_{match}()$ for removing (or adding) and relabeling; the distance $d(F, G)$ is evaluated according to the following recursive lemma:

$$
\begin{aligned}
&d(F,\emptyset) = d(F - \Gamma_F, \emptyset) + c_{del}(\Gamma_F) \\
&d(\emptyset,G) = d(\emptyset, G - \Gamma_G) + c_{del}(\Gamma_G) \\
&d(F,G) = min \begin{cases} (a)\ d(F - \Gamma_F, G) + c_{del}(\Gamma_F) \\ (b)\ d(F, G - \Gamma_G) + c_{del}(\Gamma_G) \\ (c)\ \begin{array}{l} d(T(\Gamma_F) - \Gamma_F, T(\Gamma_G) - \Gamma_G) \\ +d(F - T(\Gamma_F), G - T(\Gamma_G)) + c_{match}(\Gamma_F, \Gamma_G) \end{array} \end{cases}
\end{aligned} \tag{1}
$$

Operations *(a)* and *(b)* are respectively the cost $c_{del}()$ of removing Γ_F or Γ_G while *(c)* is the cost $c_{match}()$ of relabeling the Γ_F by Γ_G. Later, Klein et al. [8] reduced the overall complexity in time and space by splitting the tree structure based on the heavy path (defined in Section 3.2). Demaine et al. [4] further improved this algorithm by storing subtrees scores in order to reduce calculation time. Finally Touzet et al. [5] used a decomposition strategy to dynamically select the best nodes to recurse on between rightmost and leftmost, which reduces the number of subtrees in memory.

Regarding the costs, a common practice is to use *apriori* fixed costs for the primitive operations [10], i.e. : 1 for removing a node, 0 for relabeling a node by another if their tags are similar and 1 otherwise. However as these costs strongly impact the isomorphism evaluation ones can find some approaches that try to estimate these costs in a more precise manner. Most of the non-deterministic approaches are based on learning and training techniques. As tree-edit distance is a generalization of the string edit distance ones can find approaches regarding cost estimation in this domain. For example in their paper Oncina et al. [12] used stochastic transduction for costs learning. Regarding tree edit distance itself, in their paper Neuhaus et al.[11] used an automated cost estimation techniques based of probability distribution of edit distance results.

3 Tree-edit distance for structural document-query matching

We assume that a query is composed of content (keywords) and structure conditions, as shown in Figure 1. The document-query similarity is evaluated by considering content and structure separately. We then combine these scores to rank relevant elements. In this section, we first describe the content evaluation and then detail our structure matching algorithm based on tree edit distance.

3.1 Content relevance score evaluation

First, we used a $tf \times idf$(Term Frequency $\times$ Inverse Document Frequency [7]) formula to score the document leaf nodes according to query terms contained in content conditions. We evaluated two main propagation approaches which only differ on how to handle these content conditions. In the first model (which we call *Vague*) content parts of the query are merged and the content score is

evaluated in a one time pass. In our second approach which we define as *Strict*, the content conditions are considered separately and summed at the end of the process.

Regarding our propagation algorithm, our intuition is that an inner node score must depend on three elements. First, it must contain its leaves relevance (this is what we call the intermediate score). Second we should score higher a node located near a relevant element than a node located near an irrelevant one. Finally, there must be a way to balance the hierarchical effect on the node score. Based on these constraints we define the content score $c(n)$ of an element n as the *intermediate content score of the element itself plus its father's intermediate score plus all its father's descendants score*. Recursively, and starting from the document root:

$$c(n) = \begin{cases} \underbrace{\frac{p(n)}{\mid leaves(n) \mid}}_{(i)} + \underbrace{\frac{p(a_1) - p(n)}{\mid leaves(a_1) \mid}}_{(ii)} + \underbrace{\frac{c(a_1) - \frac{p(a_1)}{|leaves(a_1)|}}{\mid children(a_1) \mid}}_{(iii)} & \text{if } n \neq root \\ \overbrace{\frac{p(n)}{\mid leaves(n) \mid}} & \text{otherwise} \end{cases} \quad (2)$$

(i) is the *intermediate content score* part, with $\mid leaves(n) \mid$ the number of leaf nodes descendants of n and $p(n)$ the intermediate score of the node based on the sum of the scores of all its leaf nodes: $p(n) = \sum_{x \in leaves(n)} p(x)$, with $p(x)$ evaluated using a $tf \times idf$ formula.
(ii) is the *neighborhood score* part which allows us to convey a part of the relevance of a sibling node through its father a_1. $p(a_1)$ is the intermediate score of a_1 and $\mid leaves(a_1) \mid$ the number of leaves of a_1.
(iii) is the *ancestor scores*, evaluated with $c(a_1)$ the final score of the father a_1 minus its intermediate score.

3.2 Structure relevance score evaluation

The second part of our approach is the structure score evaluation. Our structural evaluation process follows three steps. The first one is the subtree selection and extraction. The second part is the tree edit distance. The final step is then the structure score combination. As the final part is strongly related to the subtree extraction we will post-pone our explanations on it at the end of this section.

Edit distance optimal path As seen in Section 2.2, the tree edit distance is a way of measuring similarity based on the minimal cost of operations to transform one tree to another. The number of subtrees stored in memory during this recursive algorithm depends on the direction we choose when applying the operations. Our algorithm is an extension of the optimal cover strategy from Touzet et al. [5]. The difference is that the optimal path is computed with the

help of the *heavy path* introduced by Klein et al. [8]. The heavy path is the path from root to leaf which passes through the rooted subtrees with the maximal cardinality. This means that selecting always the most distant node from this path allows us to create the minimal set of subtrees in memory during the recursion : this is the *optimal cover strategy*. Formally a heavy path is defined as a set of nodes $[n_1, ..., n_z]$ satisfying:

$$\forall (n_i, n_{i+1}) \in heavy \begin{cases} n_{i+1} \in children(n_i) \\ \forall x \in children(n_i), x \notin n_{i+1}, \mid T(n_{i+1}) \mid \geq \mid T(x) \mid \end{cases} \quad (3)$$

This strategy is used on the document and the query as input to our following tree edit distance algorithm.

Algorithm 1: Edit distance using optimal paths

```
p_F, p_G = 1;
d(F, G, p_F, p_G) begin
    if F = ∅ then
        if G = ∅ then
            return 0;
        else
            return d(∅, G - O_G.get(p_G)), p_F, p_G++) + c_del
            (O_G.get(p_G));
        end
    end
    if G = ∅ then
        return d(F - O_F.get(p_F)), ∅, p_F++, p_G) + c_del (O_F.get(p_F));
    end
    a = d(F - O_F.get(p_F), G, p_F++, p_G) + c_del (O_F.get(p_F));
    b = d(F, G - O_F.get(p_F), p_F, p_G++) + c_del (O_G.get(p_G));
    c = d(T(O_F.get(p_F)) - O_F.get(p_F), T(O_G.get(p_G)) - O_G.get(p_G),
    p_F++, p_G++) + d(F - T(O_F.get(p_F)), G - T(O_G.get(p_G)),
    next(p_F), next(p_G)) + c_match (O_F.get(p_F), O_G.get(p_G));
    return min(a, b, c);
end
```

F, G are two forests (i.e. the document and the query as first input), p_F and p_G are positions in O_F and O_G the *optimal paths* (i.e. paths of the *optimal cover strategy*). Function $O.get(p)$ returns the node in path O corresponding to position p.

Edit distance costs evaluation As seen in section 2.2, tree edit distance operation costs are generally set to 1 for removing, to 0 for relabeling similar tags and to 1 otherwise [13]. This could be explained by the fact that edit distance is usually used to score slightly different trees. However in our approach document trees are usually larger than query trees which means that the edit costs must be more precise in their representation of the structural distance of two tags. There is two more constraints in estimating these costs. The first constraint is

formal. As relabeling is equivalent to removing and then adding a node, its cost should be at most equivalent to two removings. The last constraint is a domain one. An IR model should be efficient as well as effective. That's why we need to reduce the estimation of these costs to the minimum. For all these reasons we propose to use the DTD of the considered collection which contains all the transition rules betweens the document elements.

We use this DTD to create an undirected graph representing all the possible transitions between elements. We choose it to be undirected in order to make elements strongly connected. The idea behind this is that the more a node is isolated the less its cost will be. Figure 2 illustrates the transformation of both DTDs of the Datacentric collection into graphs.

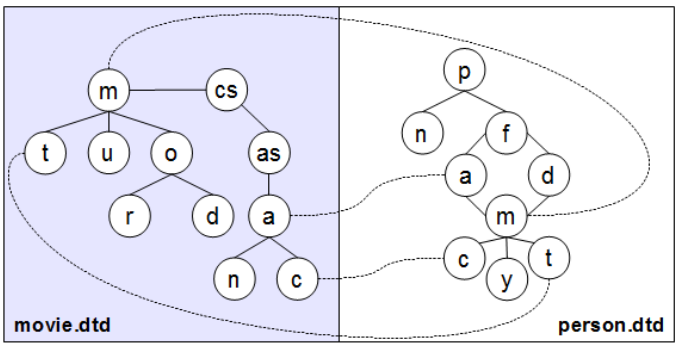


Fig. 2. Partial representation of the graphs created from both "person" and "movie" DTDs.

As the Datacentric collection comes up with two distinct DTDs (respectively *movie* and *person*) we choose to create three graphs : one for each DTD and a last one merged on the labels equivalent in the two (hashed links on figure 2). This merged DTD is used when the structure specified in the query is not explicit enough to determine if the document is following either of the two available specific DTDs. For example in the query *//movie[about(*, terry)]//character[about(*, gonzo)]* relevant nodes could be find either in a movie document or in a person document. The merged DTD should thus be used.

In order to process the substitution cost $c_{match}(n_1, n_2)$ of a node n_1 by a node n_2, respectively associated with the tags t_1 and t_2, we seek the shortest path in these DTD graphs through a Floyd-Warshall [6] algorithm. The shortest path allows to overcome the cycle issues we can encounter in a regular graph. We divide this distance by the longest of all the shortest paths that can be computed from this node label to any of the other tags in the DTD graph. Formally, with $sp()$ our shortest path algorithm :

$$c_{match}(n_1, n_2) = \frac{sp(t_1, t_2)}{max(sp(t_1, t_x))} \forall x \in DTD \tag{4}$$

Following the above formula and the figure 2 the relabeling cost for a node with a label m with a node labeled d will be $2/4$ as the shortest path from $m \rightarrow d$

has a distance of 2 and the longest of the shortest path is $m \rightarrow c$ which has a distance value of 4.

Similarly the removing cost is the highest cost obtained from all the substitution costs between the current document node and all of the query nodes. Formally

$$c_{del}(n_1) = max(\frac{sp(t_1, t_y)}{max(sp(t_1, t_x))}) \forall x \in DTD; \forall y \in Q \tag{5}$$

In our example in figure 3 the deletion cost of the node associated with tag o will be the max between $o \rightarrow d$, $o \rightarrow a$, $o \rightarrow n$ and $o \rightarrow p$ which is 5. The longest shortest path being 5 our deletion cost is equal to 1.

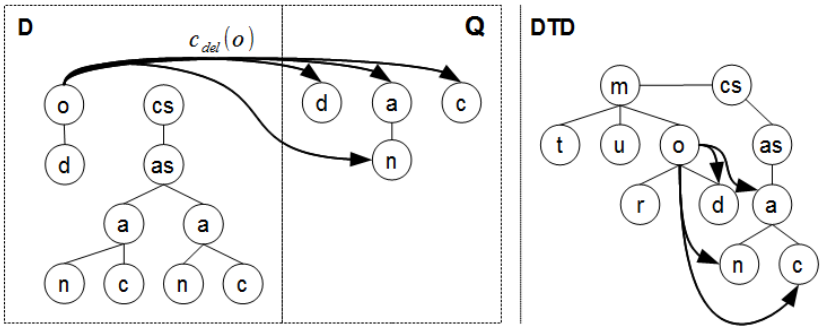


Fig. 3. Example of the removing cost evaluation of a node labeled "o".

Subtree extraction and evaluation As said previously we used two main models to score and rank relevant nodes, namely *Strict* and *Vague*. These models also differ on their way to score structure. In the *Strict* model we use the minimal subtree representing all the relevant nodes labeled with a label contained in the query as input for the matching process. This subtree is reconstructed from all branches extracted from the relevant nodes. On the other hand, in the *Vague* algorithm we extract all the subtrees rooted from the first node with a label matching a label in the query to the documents root.

Formally, for the *Vague* approach, with $Anc(n)$ the set of n ancestors; $a \in Anc(n)$; $T(a)$ the subtree rooted in a; $d(T(a), Q)$ the edit distance between $T(a)$ and Q, the structure score $s(n)$ is :

$$s(n) = \frac{\sum_{a \in \{n, Anc(n)\}} (1 - \frac{d(T(a),Q)}{|T(a)|})}{\mid Anc(n) \mid} \tag{6}$$

This method is illustrated in the left part of the figure 4. Starting from the first ancestor of a relevant leaf matching a label from the query to the root we have eight different subgraphs. The final structure score of a node will be the combination of all its father's rooted subtrees. For example, the subtrees used for the score of the node associated with the label d are 1, 2 and 8. The idea behind this extraction is that a node located near another one matching the structural constraint should get an improvement to its score.

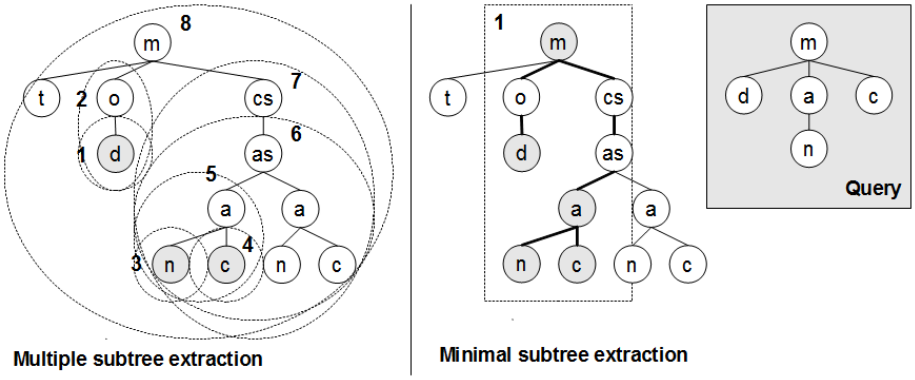


Fig. 4. Subtree extracted regarding each of our methods. On the left the multiple subtree extraction. On the right the minimal subtree extraction.

In our second technique which we call *strict* the subtree S is created from the combination of all the paths from the deepest relevant nodes which contain a label of the query to the higher in the hierarchy containing a label of the query. The subtree is then the merged paths rooted by a node having the same label than the query root. This is used to get the minimal relevant subtree possible for the edit distance algorithm input. Formally our subtree is composed of all the nodes a extracted as follow

$$\{a \in G \mid a \in \{n, Anc(n)\}, \forall n \in leaves \wedge p(n) \neq 0\} \tag{7}$$

In this particular case there is no need to combine the final score from various edit distances. We only have one edit distance for all the nodes in the created subtree. The final score is then

$$s(n) = \frac{d(S,Q)}{\mid S \mid} \tag{8}$$

This second method is illustrated in the right part of the figure 4. Starting from the first relevant leaves ancestor matching a label from the query (in our case "d", "n" and "c") we extract three branches. We then merge them into one subgraph.

3.3 Final structure and content combination

For both models, the final score $score(n)$ for each candidate node n is evaluated through the linear combination of the previously normalized scores $\in [0, 1]$. Then the elements corresponding to the target nodes are filtered and ranked. Formally, with $\lambda \in [0, 1]$:

$$score(n) = \lambda \times c(n) + (1 - \lambda) \times s(n). \tag{9}$$

4 Experiments and evaluation

We submitted a total of three runs in the INEX 2011 Datacentric track. These runs are *Strict with split DTD* in which we used the three DTD graphs; *Strict with merged DTD* with only the merged DTD and *Vague with no DTD* for our solution in which the edit distance operation costs are fixed to 1 for removing a node not in the query 0.5 for a node with a tag in the query, 0 for a relabeling of one node with another if their label are equivalent and 1 otherwise. As the Datacentric task asks to return whole documents, and as our method retrieve elements, we decided to score documents with the score of the best element they contain. For the *Strict* runs ans according to previous experiments we choose the λ parameter from the equation (9) to 0.4 while for the *Vague* run λ is set to 0.6. Before going further it is important to notice that the runs we submitted for the INEX 2011 Datacentric track were launched over a corrupted index missing around 35% of the documents (mostly movie ones). This can explain the very low official results obtained. Results are presented in table 1.

Runs	**MAP**	**P@5**	**P@10**	**P@20**	**P@30**
Strict with split DTD	0.1046	0.2526	0.2605	0.2487	0.2360
Strict with merged DTD	0.0801	0.1895	0.2026	0.1724	0.1702
Vague with no DTD	0.041	0.0737	0.0684	0.0684	0.0684

Table 1. Our official INEX 2011 results with λ set to 0.4 for our *Strict* method and 0.6 for our *Vague* approach over our previous corrupted index.

The same runs evaluated over a non-corrupted index are presented in table 2. For this article we also decided to experiment all the possible combinations of our various algorithms in order to study the influence of the DTD use as well as our choice to split the content constraints.

Runs	**MAP**	**P@5**	**P@10**	**P@20**	**P@30**
Strict with split DTD	0.1613	0.2722	0.2583	0.2593	0.2407
Strict with merged DTD	0.1289	0.2389	0.2000	0.2167	0.2204
Strict with no DTD	0.1280	0.2278	0.2278	0.2083	0.2102
Vague with split DTD	0.1143	0.1947	0.1816	0.1711	0.1553
Vague with merged DTD	0.1206	0.1842	0.1684	0.1553	0.1360
Vague with no DTD	0.1667	0.3368	0.2912	0.2684	0.2588

Table 2. Our corrected INEX 2011 results with λ set to 0.4 for our *Strict* method and 0.6 for our *Vague* approach over a clean index.

Surprisingly our most recent method which gave better results on the Datacentric 2010 test collection, namely the *Strict* one, scores less (in average) than our previous method *Vague* which merges all the content constraints. This could due to the fact that since we return the whole document and not the element there is less interests in limiting the content conditions to the structure. Regarding the DTD itself we cannot conclude on its general effectiveness. It seems to be a good way to estimate costs for edit distance when the input trees are similar (Strict runs) while it decreases results in case of trees being of different cardinalities. Finally our revised runs score less than the top ten participants.

We suppose that one of the critical issue is our choice about global document scoring: as said previously our algorithm is not designed to retrieve a whole document but an element.

4.1 Conclusions and future work

In this paper we presented two of our XML retrieval models whose main originality is to use graph theory through tree edit distance. We proposed a way of estimating the tree edit distance operation costs based on the DTD. It appears that the use of the DTD in our case seems to be only relevant in the context of matching trees with approximatively the same cardinality as the more the difference between tree cardinalities increases, the more it decreases the results. Finally, as our system is designed to retrieve elements and not whole documents we made the choice to set the documents score to the score of it's best element. This decision could have impacted our results over the other INEX participants.

References

1. Evandrino G. Barros, Mirella M. Moro, and Alberto H. F. Laender. An Evaluation Study of Search Algorithms for XML Streams. *JIDM*, 1(3):487–502, 2010.
2. M. Ben Aouicha, M. Tmar, and M. Boughanem. Flexible document-query matching based on a probabilistic content and structure score combination. In *Symposium on Applied Computing (SAC), Sierre, Switzerland*. ACM, mars 2010.
3. Michael A. Bender and Martin Farach-Colton. The lca problem revisited. In *Proceedings of the 4th Latin American Symposium on Theoretical Informatics*, LATIN '00, pages 88–94, London, UK, 2000. Springer-Verlag.
4. E. D. Demaine, S. Mozes, B. Rossman, and O. Weimann. An optimal decomposition algorithm for tree edit distance. *ACM Trans. Algorithms*, 6:2:1–2:19, December 2009.
5. S. Dulucq and H. Touzet. Analysis of tree edit distance algorithms. In *Proceedings of the 14th annual symposium of combinatorial pattern matching*, pages 83–95, 2003.
6. Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5:345–, June 1962.
7. K. Sparck Jones. Index term weighting. *Information Storage and Retrieval*, 9(11):619–633, 1973.
8. P N. Klein. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms*, ESA '98, pages 91–102, London, UK, 1998. Springer-Verlag.
9. VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.
10. Yashar Mehdad. Automatic cost estimation for tree edit distance using particle swarm optimization. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 289–292, 2009.
11. Michel Neuhaus and Horst Bunke. Automatic learning of cost functions for graph edit distance. *Information Science*, 177(1):239–247, 2007.
12. Jose Oncina and Marc Sebban. Learning stochastic edit distance: Application in handwritten character recognition. *Pattern Recogn.*, 39:1575–1587, September 2006.
13. K-C. Tai. The tree-to-tree correction problem. *J. ACM*, 26:422–433, July 1979.

UPF at INEX 2011
Data Centric and Books and Social Search tracks

Georgina Ramírez

Universitat Pompeu Fabra, Barcelona, Spain
georgina.ramirez@upf.edu

Abstract. This paper describes our participation at INEX 2011. We participated in two different tracks: Data Centric and Books and Social Search. In the Books and Social Search track we only participated in one of the tasks: Social Search for Best Books (SB). We studied the performance effects of using different query fields for query expansion. In the Data Centric track we participated in both tasks: Adhoc and Faceted Search. In the Adhoc task we studied the effects of using different indices depending on what type of object is asked for. For the Faceted task we use a fixed set of facets and experiment with the hierarchical and non hierarchical presentation of them.

Keywords: XML, focused retrieval, INEX, faceted search, query expansion

1 Introduction

We describe the experiments performed on the INEX 2011 data for two different tracks: data centric and books and social search.

2 Social Search for Best Books

From the *Books and Social Search* track we only participated in the *Social Search for Best Book* task (SB). The goal of this task is to investigate the value of user-generated metadata (e.g., 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.

Thus the task is to find the most relevant books given a topic. After performing a few experiments on the training set, we observed the following: 1) There are, in general, a few relevant books per topic. Most of the time it is not difficult to find them. The main problem is to rank them high, to obtain a good early precision. 2) When giving emphasis to the words contained in the *title* field we improve precision but we miss some relevant results, we loose in recall. 3) When giving emphasis to the words contained in the *tags* field, we improve slightly both, precision and recall.

Besides expecting to verify whether these observations hold in the larger set, in this track we investigate the following research questions:

1.- Do the *genre* and *group* fields from the topic description contain useful terms for query expansion?
2.- How does the use of the *tag count* information affect retrieval performance?
3.- Can we improve precision by using the *category* information from the books?

2.1 Collection

The social data collection contains metadata for 2.8 million books crawled from the online book store of Amazon and the social cataloging web site of LibraryThing in February and March 2009 by the University of Duisburg-Essen. The data set is available as a MySQL database or as XML. For each book, the following metadata is included:

From 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). From LibraryThing: Tags (including occurrence frequency), blurbs, dedications, epigraphs, first words, last words, quotations, series, awards, browse nodes, characters, places, subjects.

2.2 Experiments setup

This section describes the setup of the experiments carried out for the *Social Search for Best Book* task of INEX 2011. 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 lambda 0.2. Topics and documents have been pre-processed using the Krovetz stemmer [2] and the Smart stop-word list [3]. We indexed all the fields in the collection.

2.3 Runs

The different NEXI [4] queries used for our official runs are presented in Table 2.3.

Runs number 1 and 2 are our baselines and simply give emphasis to the words contained in different fields of the documents. The first one gives a bit more importance to the words contained in the *tags* field. The second one gives emphasis to the words contained in the *tags* but also in the *title* fields. For these runs the query is formed by the words contained in the *title* field of the topic description.

Runs number 3, 4, and 5 perform query expansion with the terms contained in the *genre* and *group* fields of the topic description. These runs use the same NEXI query type as run 2 and should give us some insight whether the terms contained in these fields are useful terms for query expansion (first research question).

Table 1. Description of the official runs.

Run id	NEXI query
1 UPF_base_BT02	//book[about(.,q) AND about (.//tags, q)]
2 UPF_base_BTT02	//book[about(.,q) AND about (.//tags, q) AND about (.//title, q)]
3 UPF_QE_genre_BTT02	//book[about(.,q+ge) AND about (.//tags, q+ge) AND about (.//title, q+ge)]
4 UPF_QE_group_BTT02	//book[about(.,q+gr) AND about (.//tags, q+gr) AND about (.//title, q+gr)]
5 UPF_QE_genregroup_BTT02	//book[about(.,q+ge+gr) AND about (.//tags, q+ge+gr) AND about (.//title, q+ge+gr)]
6 UPF_QEGr_BTT02_RM	//book[about(.,q+ge+gr) AND about (.//tags, q+ge+gr) AND about (.//title, q+ge+gr)]

Run number 6 is the same as run number 5 applying a post-processing step were all isbn numbers from the *similar* and *dissimilar* fields of the topic description are removed. Our assumption for this run is that the books contained in the *similar* and *dissimilar* fields are examples of books that the user gives to help with the search but are not books the user wants to see in the result set. Note that only 56 out of the 211 topics contain any of these informations in its description.

2.4 Results

The results of our official runs are shown in Table 2.4.

Table 2. Official results for the SB runs.

Run id	MAP	MRR	P@10	P@20	R-precision
UPF_base_BT02	0.1048	0.2039	0.0796	0.0756	0.0949
UPF_base_BTT02	0.1018	0.2135	0.0863	0.0706	0.0909
UPF_QE_genre_BTT02	0.0910	0.2089	0.0844	0.0725	0.0841
UPF_QE_group_BTT02	**0.1223**	**0.2478**	**0.0995**	**0.0834**	**0.1265**
UPF_QE_genregroup_BTT02	0.1001	0.2283	0.0934	0.0787	0.1042
UPF_QEGr_BTT02_RM	0.0973	0.2183	0.0872	0.0718	0.1049
Best performing run at INEX 2011	0.2283	0.4811	0.2071	0.1569	0.2225

Our best run is the one that performs query expansion using the terms contained in the *group* field of the topic (fourth row). This run improves significantly over its baseline (second row). However, it performs very poorly compared to the best performing run in this task (last row). We plan to investigate why the performance of our baselines are so poor.

Note that when removing the similar and dissimilar books from our best run, we obtain a much worse overall performance (sixth row). This indicates that our assumption that the user does not want to see the examples he or she suggests is not true.

Performing query expansion with the *genre* information performs much worse than performing it with the *group* one. That is probably because the terms contained in the *genre* field are generally more generic. For example, in topic number 399 the *title* is "cakes", the *group* is "sweet treat" and the *genre* is "technology home economics cooking". It could be that when adding the *genre* terms in the query terms such as *technology* or *economics* introduce some noise, irrelevant results. Further analysis needs to be done in order to confirm this hypothesis. We also plan to analyze, on a topic base, in which cases does the use of the *genre* and *group* information help most effectively to improve retrieval performance.

Having a look at the relevance assessments we can see that there are very few relevant documents per topic, an average of 11.3 (median 7). This confirms our observation from the training set. Furthermore, for more than a third of the topics, the task can be seen as a know-item search since a very small set of relevant results can be found: 17 topics have only 1 relevant result, 37 topics have 1 or 2 relevant results, and 79 topics have less or 5 relevant results.

The extra experiments performed to address the other research questions and its results will be reported in the final version of this paper.

3 Data Centric track

The goal of the Data Centric Track is to investigate retrieval over a strongly structured collection of documents, in particular, the IMDB collection. The track features two tasks. In the *Ad Hoc* Search Task retrieval systems are asked to answer informational requests with the entities contained in the collection (movies, actors, directors, etc.). In the *Faceted* Search Task retrieval systems are asked to provide a restricted list of facets and facet-values that will optimally guide the searcher toward relevant information.

3.1 Collection

The track uses the IMDB data collection generated from the plain text files published on the IMDb web site on April 10, 2010. There are two kinds of objects in the 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. Each XML file contains information about one object, i.e. a single movie or person. In total, the IMDB data collection contains 4,418,081 XML files, including 1,594,513 movies, 1,872,471 actors, 129,137 directors who did not act in any movie, 178,117 producers who did not direct nor act in any movie, and 643,843 other people involved in movies who did not produce nor direct nor act in any movie.

3.2 Experiments setup

This section describes the setup of the experiments carried out for both tasks in the *Data-Centric* track of INEX 2011. 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 lambda 0.15 (based on our last year experiments in the same collection). Topics and documents have been pre-processed using the Krovetz stemmer [2] and the Smart stop-word list [3].

We created two different indices for the collection: one for movies and one for persons. We indexed all fields for both types of documents.

3.3 Adhoc Search Runs and Results

Runs Our approach for the *Adhoc Search* task is based on the results obtained from our last year participation in the track, where we found out that indexing only the *movie* documents of the collection performed much better than indexing all the collection, one of the best performing runs of last year. For this year, we wanted to check whether we can improve those results by running topics in different indices according to the object they ask for.

Therefore, we manually classified all topics according to which type of object the users are searching for: movies or persons. We found that only 9 out of the 45 topics were asking for persons. We then run each topic on its specific index, either movies or persons, and merged the results for submission.

We only submitted two runs, both following the same approach. The first one, UPFbaseCO2i015, uses the CO title of the topic while the second one, UPFbaseCAS2i015, uses the CAS one.

Results The results of our official runs are shown in Table 3.3.

We can see that using the CO title of the topic performs much better than using the CAS version of it. That could be due to the strictness we treat the structure of the query in our run but further analysis will be done in order to confirm this hypothesis.

Our runs are clearly better at precision than recall (when looking at their ranking position). This suggests that the use of independent indices might indeed help to improve on early precision. The low performance at high recall levels could be due to the strictness of retrieving only one type of object per topic.

Table 3. Official results for the Data Centric, adhoc search runs. The number in parentheses indicates the run position in the official ranking.

Run id	MAP	P@5	P@10	P@20	P@30
UPFbaseCO2i015	**0.2696** (9)	**0.4211** (9)	**0.4342** (5)	**0.4171** (3)	**0.3825** (5)
UPFbaseCAS2i015	0.1117 (26)	0.3579 (18)	0.3474 (14)	0.3211 (13)	0.3070 (13)

Many topic authors have assessed both types of objects as relevant. A deeper analysis on these issues will be reported in the final version of this paper.

3.4 Faceted Search Runs and Results

Runs For the *Faceted search* task we used a fixed set of facets, namely *directors*, *actors*, and *genres*, and experiment with the hierarchical and non-hierarchical presentation of them. For that, we first took the reference run and extracted the most popular facets values for each of our facets. By popular we mean the most repeated facet-value in the result set. We then presented these facet-values in different ways.

We submitted three runs. The parameters and description of our runs can be found in Table 3.4.

Table 4. Description of the official faceted search runs.

Run id	Type	facet-value number
1 UPFfixGDAnh2	Non hierarchical	genre (7), director (5), actor (8)
2 UPFfixGDAh	Hierarchical	genre (7), director (5), actor (8)
3 UPFfixGDAh2	Hierarchical	genre (20), director (20), actor (20)

The first run is a non hierarchical run, which means that presents all the facet-value pairs at the same level. Thus, we return 20 facet-value pairs per topic. The second and third run are hierarchical, meaning that for each genre-value pair, we return all director-value pairs, and within each director-value pair we return all actor-value pairs. Thus we return 280 and 8000 results per topic respectively.

Results Official results for the *Faceted search* task are not yet available.

4 Discussion and Conclusions

This paper described our participation at INEX 2011. We participated in two different tracks: Data Centric and Books and Social Search.

Acknowledgments This work has been supported by the Spanish Ministry of Science and Education under the HIPERGRAPH project and the Juan de la Cierva Program.

References

1. T. Strohman, D. Metzler, H. Turtle, and W. B. Croft, Indri: a language model based search engine for complex queries. Proceedings of the International Conference on Intelligent Analysis, 2005.
2. R. Krovetz, Viewing morphology as an inference process. In Proc. of the 16th ACM SIGIR Conference, Pittsburgh, June 27-July 1, 1993; pp. 191-202.
3. G. Salton, The SMART Retrieval SystemExperiments in Automatic Document Processing. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1971.
4. A. Trotman, B. Sigourbjörnsson Narrowed Extended XPath I (NEXI). In Advances in XML Information Retrieval, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2005.

University of Amsterdam Data Centric Ad Hoc and Faceted Search Runs

Anne Schuth and Maarten Marx

ISLA, University of Amsterdam, The Netherlands
{anneschuth,maartenmarx}@uva.nl

Abstract. We describe the ad hoc and faceted search runs for the 2011 INEX data centric task that were submitted by the ILPS group of the University of Amsterdam.

Description of the runs

As our XPath/XQuery processor we used eXist version 1.4.1, a native XML database (Meier, 2009). Running NEXI queries (Trotman and Sigurbjörnsson, 2005) —all CAStitles are NEXI expressions— is easy in eXist because it supports full integration of XQuery with full-text search. Full-text search is implemented using Lucene 2.9.2. We defined Lucene indexes on all elements E, for which a topic exists which checks whether E is `about()` some full text expression T. This includes the two "document-indexes" on `movie` and `person` elements.

The INEX NEXI queries are easily translated into the specific syntax of XQuery with full-text search employed by eXist. Table 1 describes the rewrite rules we used.

We also translated each content-and-structure query into a content-only query. We did so by first repacing all paths by '.', and then replacing expressions of the form `.[Q1]/.[Q2]` by `.[Q1 and Q2]`. Boolean and's and or's were kept. For example (based on the Dr Gonzo query),

CAS `//A[ about(.//B,'b')]//C[ about(.//D,'d') or about(.//E,'e')]`
CO `.[about(.,'b') and ( about(.,'d') or about(.,'c') )]`

Both our ad hoc and faceted search runs are implemented as XQueries. We now describe the specific settings used.

Ad hoc runs. For ad hoc, we created mixture models. Experiments with estimating the best value for λ on the INEX 2010 collection had no success. Thus we used the extreme values: 0.0, 0.5 and 1.0. For each document d, we calculate the Document Score ($ds(d)$) and the Element Score ($es(d)$), as follows

$ds(d)$ is simply the Lucene score of the document for the CO version of the CAS query. For `$d` a document and `Q` a NEXI expression, this score is available in eXist by the function `ft:score($d/Q)`).

$es(d)$ is the maximum Lucene-score for any element in document d answering the NEXI query. For `$d`a document and `Q` a NEXI expression, this score is calculated as `max( for $e in $d/Q return ft:score($e) )`.

Table 1. Rewrite rules for CAStitles

before	after
`OR`	`or`
`AND`	`and`
`about(P,T)`	`ft:query(P,'T')`

Faceted runs We have to define a limited number of facets on our documents. We selected facets in such a way that each can cover a broad range of values while these facetvalues are not unique for a single document. Movies and persons have different facets. For each facet, we define its name and an XPath expression by which the facetvalues can be retrieved. The possible values are dictated by the data. We list our selection in Table 2.

We defined two strategies for the selection and ordering of facetvalues. The first system, which we will call *count*, ranks facetvalues based on the number of —not necessarily relevant— hits that would be the result if the facetvalue were selected. This is captured by the following calculation:

```
count($hits[facet-path eq $value])
```

A slightly more elaborate system, which we call *sumscore*, orders facetvalues based on the sum of the Lucene full-text score of a certain subset of the hits that would be the result if the facetvalue were selected. Exactly what kind of subset is a parameter in this function. We have used the 10 documents with the highest Lucene score.

Both runs use the provided standard run for retrieval of the documents.

Acknowledgments

The authors acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under the FET-Open grant agreement FOX, number FP7-ICT-233599. This research was supported by the Netherlands organization for Scientific Research (NWO) under project number 380-52-005 (PoliticalMashup).

Bibliography

Meier, W. (2009). eXist: An open source native XML database. *Web, Web-Services, and Database Systems*, pages 169–183.

Trotman, A. and Sigurbjörnsson, B. (2005). Narrowed extended xpath i (NEXI). *Advances in XML Information Retrieval*, pages 16–40.

Table 2. Names and paths, defined as XPath expressions, of our selections of facets. Some apply to topics that ask for movies, others to topics that ask for persons, there is no overlap.

name	facet-path	applies to
director	.//directors/director	movie
writer	.//writers/writer	movie
genre	.//genres/genre	movie
keyword	.//keywords/keyword	movie
actor	.//cast/actors/actor/name	movie
producer	.//cast/producers/producer	movie
certification	.//certifications/certification	movie
language	.//additional_details/languages/language	movie
country	.//additional_details/countries/country	movie
name	.//person/name	person
height	.//height	person
other-movie	.//filmography/act/movie/title	person
other-movie-year	.//filmography/act/movie/year	person
nickname	.//nicknames/name	person

BUAP: A Recursive Approach to the Data-Centric track of INEX 2011*

Darnes Vilariño, David Pinto, Saul León[1], Esteban Castillo[2], Mireya Tovar
{darnes,dpinto,mtovar}@cs.buap.mx, [1]saul.ls@live.com,
[2]ecjbuap@gmail.com

Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla, México

Abstract. A recursive approach for keyword search on XML data for the Ad-Hoc Search Task of INEX 2011 is presented in this paper. The aim of this approach was to detect the concrete part (in the representation tree) of the XML document containing the expected answer. For this purpose, we initially obtain a tree structure, which represents an XML document, tagged by levels. A typical search engine based on posting lists is used in order to determine those documents that match in some degree with the terms appearing in the given query(topic). Thereafter, in a recursively process, we navigate into the tree structure until we find the best match for the topic. The obtained results are shown and compared with those presented by other teams in the competition.

1 Introduction

The Data-Centric track, introduced first at 2010 and now presented in its second edition at INEX 2011, 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 with the traditional information retrieval process, where whole documents are usually indexed and retrieved as single complete units, information retrieval from XML documents creates additional retrieval challenges. In the task of document-centric XML keyword search, a simple structure of long text field predominates, however, in Data-Centric XML, the structure of document representation is very rich and carries important information about objects and their relationships [1].

Until recently, the need for accessing the XML content has been addressed diferently by the database (DB) and the information retrieval (IR) research communities. The DB community has focussed on developing query languages and eficient evaluation algorithms used primarily for Data-Centric XML documents. On the other hand, the IR community has focussed on document-centric XML documents by developing and evaluating techniques for ranked element retrieval.

* This work has been partially supported by the CONACYT project #106625, VIEP # VIAD-ING11-I, as well as by the PROMEP/103.5/09/4213 grant.

Recent research trends show that each community is willing to adopt the well-established techniques developed by the other to efectively retrieve XML content [3].

The Data-Centric track uses the IMDB data collection gathered 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 standard test collection that will allow participating groups to undertake future comparative experiments.

The rest of this paper is structured as follows. Section 2 describes the approach used for preparing the run submitted to the competition. Section 3 show the results obtained, as well as the scores reported by the rest of the teams. Finally, in Section 4 we discuss findings and future work.

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 the XML tags to a simple string containing both the corresponding XML tag and a numeric value which indicates the level at the original structure. In Figure 1 we may see the original structure of a part of an XML document ([XML]). The same Figure depicts the corresponding strings obtained as a result of the document representation transformation ([TXT]).

For the presented approach we have used an inverted index tree in order to store the XML templates of the corpus. In this kind of data structure we have considered to include both, the term and the XML tag (with its corresponding hierarchy). The aim was to be able to find the correct position of each term in the XML hierarchy and, therefore, to retrieve those parts of the XML file containing the correct answer of a given query. The numeric value introduced in the previously mentioned document representation allows to store the same term in the inverted index, even when this term occurs in different contexts. In Figure 2, we show an example of the inverted index. We may see that the dictionary entry "person.overview.alternate_names.name[1].smith" refers to the term "smith" which has a document frequency of 699, a term frequency of 1 at the document identified as "person_990001", etc. The complete name of this

```
[XML]
      <alternate_names>
         <name>Smith, Kamani Ray
         </name>
         <name>Smith, Kimani
         </name>
         <name>Smithlou, Kimani Ray
         </name>
      </alternate_names>

[TXT]
person.overview.alternate_names.name[1] smith
person.overview.alternate_names.name[1] kamani
person.overview.alternate_names.name[1] ray
person.overview.alternate_names.name[2] smith
person.overview.alternate_names.name[2] kimani
person.overview.alternate_names.name[3] smithlou
person.overview.alternate_names.name[3] kimani
person.overview.alternate_names.name[3] ray
       :                                          :
```

Fig. 1. Example of the transformation for the document representation

"smith" instance is "smith kamani ray", but there is, at least at the example showed, another "smith" whose complete name is "smith kimani". Therefore, the numeric value introduced allows to avoid confusions for identifying each of the different instances.

```
person.overview.alternate\_names.name[1].smith : (699) person\_990001:1,
                                                       person\_993004:1 ...
       :                                          :
```

Fig. 2. Example of the type of inverted index used in the experiments

We have created five different inverted indexes, for the each one of the following categories: actors, directors, movies, producers and others. 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) in a recursive manner, i.e., we produce answers for the inner operators first, and recursively we merge the results until we reach the external operators, which lead us to obtain the complete evaluation of the query.

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.

$$\mathrm{SIM}(q,d) = \sum_{c_k \in B} \sum_{c_l \in B} CR(c_k, c_l) \sum_{t \in V} weight(q,t,c_k) \frac{weight(d,t,c_l)}{\sqrt{\sum_{c \in B, t \in V} weight(d,t,c)^2}} \tag{1}$$

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 $weight(q,t,c)$ and $weight(d,t,c)$ are the weights of term t in XML context c in query q and document d, respectively (as shown in Eq. (3)).

$$\mathrm{CR}(c_q, c_d) = \begin{cases} \frac{1+|c_q|}{1+|c_d|} & \text{if } c_q \text{ matches } c_d \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where c_q and c_d are the number of nodes in the query path and document path.

$$weight(d,t,c) = idf_t * wf_{t,d} \tag{3}$$

where idf_t is the inverse document frequency of term t, and $wf_{t,d}$ is the frequency ot 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 ⟨c_q, t⟩ ∈ q do
4          w_q = weight(q, t, c_q)
5          foreach c ∈ B do
6              if CR(c_q, c) > 0 then
7                  postings = GetPostings(c, t)
8                  foreach posting ∈ 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
```

3 Experimental results

We have evaluated 45 topics with the corpus provided by the competition organizers. This dataset is made up of 1,594,513 movies, 1,872,471 actors, 129,137 directors, 178,117 producers and, finally, 643,843 files categorized as others.

In this competition we submitted one run which we was named: "p47-FCC-BUAP-R1", and the obtained results (scores and ranking) are presented as follows. Table 1 **a)** shows the Mean Average Precision for the differents runs submitted by all the teams at the competition (included ours). Table 1 **b)** shows the Precision at 5 (P@5) score for the differents runs submitted by all the teams at the competition (included ours). Here we may see that our approach obtained, in both cases, scores above median.

In Figure 3 we may see the Precision-Recall graph for the best runs measured with MAP. The curve behaviour clearly shows that we have retrieve some relevant documents at the first positions of the ranking list, however, as the number of answers increases, we introduce a number of documents that were not considered at the gold standard. This analysis lead us to consider a better way of filtering the noisy documents in order to bring the rest of the relevant documents in a better position at the ranking list.

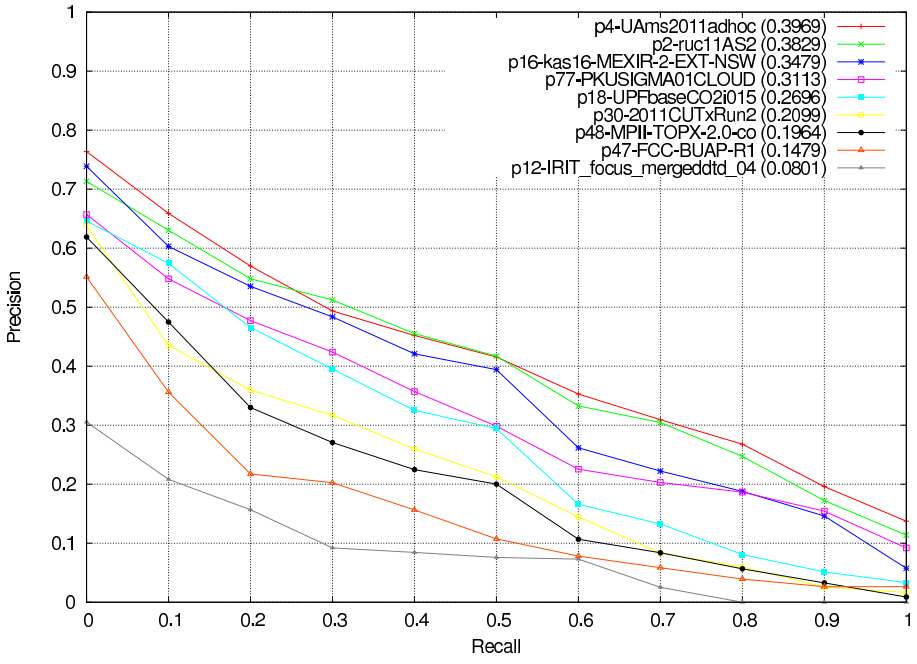


Fig. 3. Precision-Recall graph

Run	Score
p4-UAms2011adhoc	0.3969
p2-ruc11AS2	0.3829
p2-ruc11AMS	0.3655
p16-kas16-MEXIR-2-EXT-NSW	0.3479
p77-PKUSIGMA01CLOUD	0.3113
p77-PKUSIGMA02CLOUD	0.2997
p77-PKUSIGMA04CLOUD	0.2939
p77-PKUSIGMA03CLOUD	0.2874
p18-UPFbaseCO2i015	0.2696
p4-2011IlpsEs	0.2504
p4-2011IlpsEs	0.2318
p2-ruc11AI2	0.2166
p16-kas16-MEXIR-2-ANY-NSW	0.2125
p30-2011CUTxRun2	0.2099
p2-ruc-casF-2011	0.2082
p30-2011CUTxRun3	0.1968
p48-MPII-TOPX-2.0-co	0.1964
p16-kas16-MEXIR-2-ALL-NSW	0.1937
p30-2011CUTxRun1	0.1898
p4-2011IlpsDs	0.1884
p16-kas16-MEXIR-EXT2-NSW	0.183
p2-ruc11AMI	0.1804
p2-ruc11AL2	0.1677
p2-ruc11AML	0.1661
p47-FCC-BUAP-R1	0.1479
p18-UPFbaseCAS2i015	0.1117
p16-kas16-MEXIR-ANY2-NSW	0.0871
p16-kas16-MEXIR-ALL2-NSW	0.0857
p12-IRIT_focus_mergeddtd_04	0.0801
p16-kas16-BM25W-SS-SW	0.0643
p16-kas16-BM25W-NSS-SW	0.0641
p16-kas16-BM25W-SS-NSW	0.0606
p12-IRIT_large_nodtd_06	0.041
p48-MPII-TOPX-2.0-cas	0.0194

a) Mean Average Precision (MAP)

Run	Score
p77-PKUSIGMA04CLOUD	0.5158
p77-PKUSIGMA02CLOUD	0.5158
p16-kas16-MEXIR-2-EXT-NSW	0.5053
p77-PKUSIGMA03CLOUD	0.4895
p4-UAms2011adhoc	0.4842
p77-PKUSIGMA01CLOUD	0.4737
p2-ruc11AMS	0.4632
p2-ruc11AS2	0.4474
p18-UPFbaseCO2i015	0.4211
p30-2011CUTxRun2	0.4105
p30-2011CUTxRun1	0.4105
p47-FCC-BUAP-R1	0.4
p30-2011CUTxRun3	0.3895
p4-2011IlpsEs	0.3842
p48-MPII-TOPX-2.0-co	0.3632
p2-ruc11AI2	0.3632
p2-ruc-casF-2011	0.3632
p18-UPFbaseCAS2i015	0.3579
p4-2011IlpsEs	0.3316
p16-kas16-MEXIR-EXT2-NSW	0.3316
p16-kas16-MEXIR-2-ANY-NSW	0.3158
p16-kas16-MEXIR-2-ALL-NSW	0.2947
p4-2011IlpsDs	0.2895
p2-ruc11AMI	0.2789
p2-ruc11AML	0.2526
p2-ruc11AL2	0.2421
p16-kas16-MEXIR-ANY2-NSW	0.1947
p12-IRIT_focus_mergeddtd_04	0.1895
p16-kas16-MEXIR-ALL2-NSW	0.1789
p16-kas16-BM25W-SS-SW	0.0789
p16-kas16-BM25W-NSS-SW	0.0789
p12-IRIT_large_nodtd_06	0.0737
p16-kas16-BM25W-SS-NSW	0.0632
p48-MPII-TOPX-2.0-cas	0.0474

b) Precision at 5 (P@5)

Table 1. Scores reported at the Ad-Hoc Track of INEX 2011

In Figures 4, 5 and 6 we may see the results obtained by all the teams considering Precision at 10, Precision at 20 and Precision at 30, respectively. Again, in all the different metrics we have obtained scores above median.

With respect to our participation at the last year, we have greatly outperformed the obtained results. We consider that this result is associated with both, a better mechanism of translating the topics, but most of all, because in this case we have introduced a better way of indexing the documents, by including the reference to the complete hierarchy.

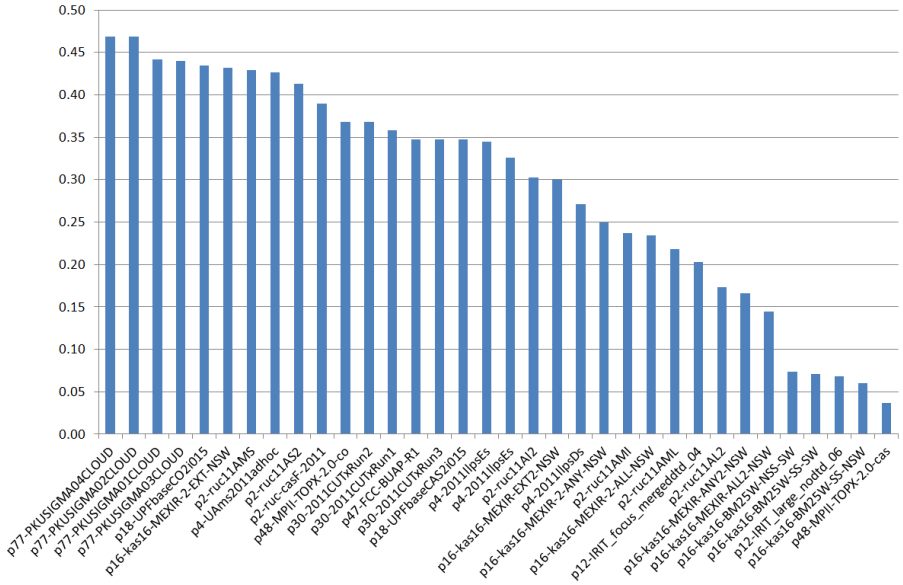


Fig. 4. Precision at 10 (P@10)

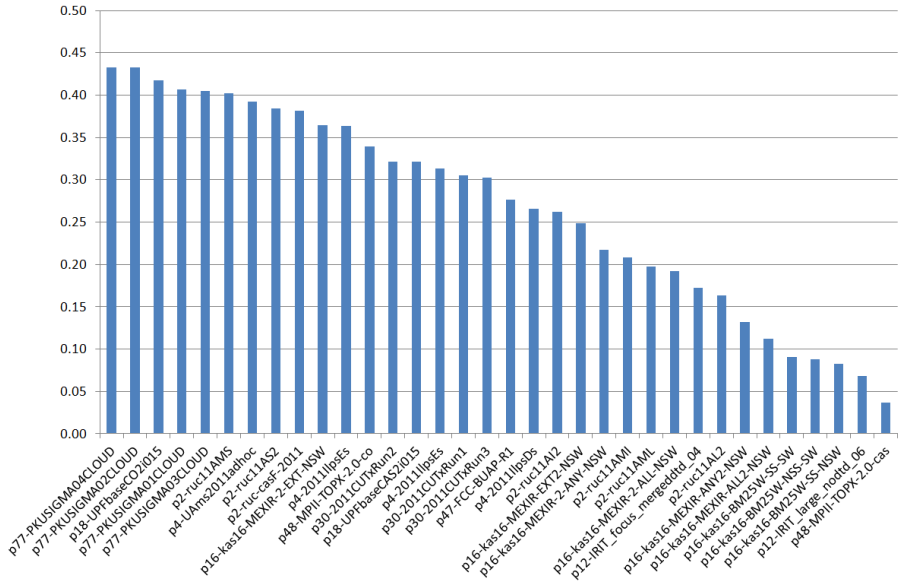


Fig. 5. Precision at 20 (P@20)

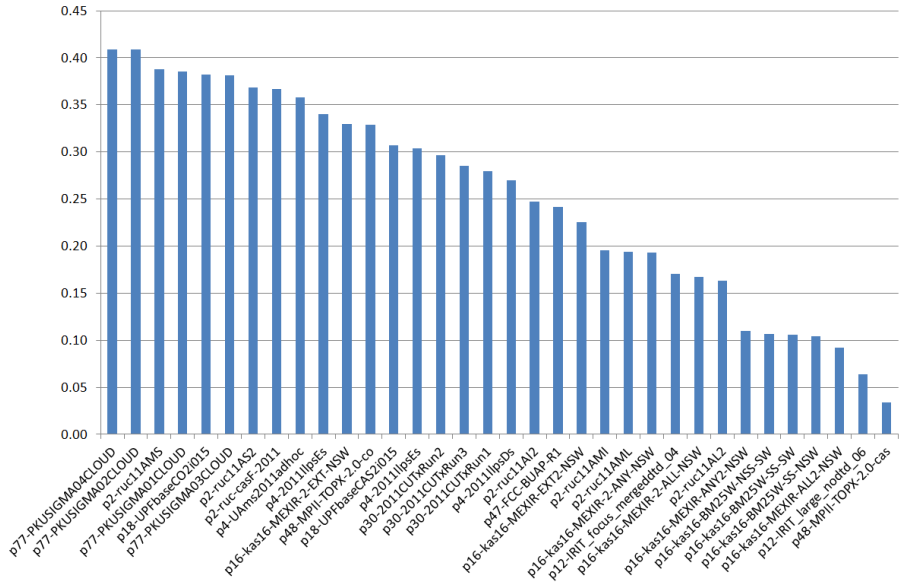


Fig. 6. Precision at 30 (P@30)

4 Conclusions

In this paper we have presented details about the implementation of an information retrieval system which was used to evaluate the task of Ad-Hoc retrieval of XML documents, in particular, in the Data-Centric track of the Initiative for the Evaluation of XML retrieval (INEX 2011).

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 with its corresponding hierarchy to correctly identify the specific part of the XML file associated to the topic.

A recursive method for evaluating the topic was used considering only the "castitle" tag. The obtained results are all above median which encourages us to still participating in this competition forum, after analyzing the manner we may improve the document representation, document indexing and document retrieval.

References

1. Wang, Q., Li, Q., Wang, S., Du, X.: Exploiting semantic Tags in XML retrieval. In: In Proc. of the INEX 2009. (2009) 133–144
2. Wang, Q., Trotman, A.: Task description of INEX 2010 Data-Centric track. In: In Proc. of INEX 2010 (same volume). (2010)

3. Amer-Yahia, S., Curtmola, E., Deutsch, A.: Flexible and efficient XML search with complex full-text predicates. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data. SIGMOD '06, New York, NY, USA, ACM (2006) 575–586
4. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK (2009)

RUC at INEX 2011 Data-Centric Track

Qiuyue Wang[1,2], Yantao Gan[1], Yu Sun[1]

[1] School of Information, Renmin University of China,
[2] Key Lab of Data Engineering and Knowledge Engineering, MOE,
Beijing 100872, P. R. China
qiuyuew@ruc.edu.cn, {ganyantao19901018, everysecondbetter}@163.com

Abstract. We report our experiment results on the INEX 2011 Data-Centric Track. We participated in both the ad hoc and faceted search tasks. On the ad hoc search task, we employ language modeling approaches to do structured object retrieval, trying to capture both the structure in data and structure in query and unify the structured and unstructured information retrieval in a general framework. However, our initial experiment results using INEX test bed show that the unstructured retrieval model performs better than structured retrieval models. On the faceted search task, we propose a simple user-simulation model to evaluate the effectiveness of a faceted search system's recommending facet-values. We implemented the evaluation system and conducted the evaluations for the track. The results show that our basic approach of recommending most frequent facet-values in the result set performs quite well.

1 Introduction

2 Ad Hoc Search

Language modeling approach has a solid statistical foundation, and can be easily adapted to model various kinds of complex and special retrieval problems, such as structured document retrieval. In particular, mixture models [1] and hierarchical language models [2][3][4] were proposed to be applied in XML retrieval. On the ad hoc search task in INEX 2011 data-centric track, we employ the language modeling approach to do structured object retrieval as the IMDB data collection can be viewed as a set of structured objects, i.e. movies and persons. With the rich structural information in data, we intend to investigate how to capture the structural information in data as well as that in query in language models to retrieve more accurate results for an ad hoc information need.

In this section, we discuss different ways of adapting language modeling approach to structured object retrieval, and evaluate them on the IMDB data collection.

2.1 Unstructured Data, Unstructured Query

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

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

$$-D(\theta_Q \,||\, \theta_D) = -\sum_{w \in V} P(w \mid \theta_Q) \log \frac{P(w \mid \theta_Q)}{P(w \mid \theta_D)} . \quad \textbf{(2)}$$

On the surface, the KL-divergence model appears to be quite different from the query likelihood method. However, it turns out that the KL-divergence model covers the query likelihood method as a special case when we use the empirical distribution to estimate the query language model, i.e. maximum-likelihood estimate.

In IMDB data collection, each document is a structured object, i.e. movie or person. Our first retrieval strategy is to ignore the structural information in each object, estimate a language model for each object based on their free-text content, and rank them using query likelihood. We only consider CO queries, so query is unstructured too in this strategy.

2.2 Structured Data, Unstructured Query

When considering the structural information in data, the common way is to view each object as consisting of multiple fields and represent its language model as a combination of different language models estimated from its different fields [5][6], as shown in Equation 3.

$$P(Q \mid D) = \prod_{w \in Q} [\sum_{i=1}^{k} P(w \mid \theta_{D_i}) P(\theta_{D_i} \mid \theta_D)] \quad \textbf{(3)}$$

Here we assume that object *D* consists of *k* fields. $P(w \mid \theta_{D_i})$ is the language model estimated from its *i*th field, which is normally smoothed with the collection's *i*th field's language model. In the generative model, $P(\theta_{D_i} \mid \theta_D)$ is thought as the probability of object *D* generating the *i*th field, which can be uniform, i.e. *1/k*, or proportional to the frequency or length of this field in object *D* as proposed in [3]. It can be also interpreted as the importance weight of *i*th field to D, which can be learned or tuned to the task. In this paper, we propose a new approach using the normalized average IDF values of terms in a field to determine the weight of this field if no training data exist.

2.3 Structured Data, Structured Query

2.4 Results

3 Faceted Search Evaluation

3.1 Cost Model

3.2 User Model

3.3 User Simulation System

4 Faceted Search

4.1 Frequency based Approach

4.2 Results

5 Conclusions and Future Work

References

1. D. Hiemstra, “Statistical Language Models for Intelligent XML Retrieval”, Intelligent Search on XML data, H. Blanken et al. (Eds.), 2003.
2. P. Ogilvie, J. Callan, “Language Models and Structured Document Retrieval”, INEX 2003.

3. P. Ogilvie, J. Callan, “Hierarchical Language Models for XML Component Retrieval”, INEX 2004.
4. P. Ogilvie, J. Callan, “Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval”, INEX 2005.
5. P. Ogilvie, J. Callan, “Combining Document Representations for Known-Item Search”, SIGIR 2003.
6. Z. Nie, Y.Ma, S. Shi, J. Wen, W. Ma, “Web Object Retrieval”, WWW 2007.
7. J. Kim, X. Xue, W.B. Croft, “A Probabilistic Retrieval Model for Semistructured Data”, ECIR 2009.

MEXIR at INEX-2011

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 second year of Kasetsart University's participation in INEX. We participated in three tracks: Snippet retrieval, Data Centric, and Web Service Discovery. This year, we introduced an XML information retrieval system that uses MySQL and Sphinx which we call the More Efficient XML Information Retrieval (MEXIR). In our system, XML documents are stored into one table that has a fixed relational schema. The schema is independent of the logical structure of XML documents. Furthermore, we present a structure weighting function which optimizes the performance of MEXIR.

Keywords: XML Retrieval, Implementation System, Sphinx, MySQL

1 Introduction

According to the large collections in the availability of electronic information, the size of information collections is growing rapidly. Large collections are commonplace now. Since, the Extensible Markup Language (XML)[1] documents have additional information; document representation of these might be add up metadata to describe data in context respect to XML language design.

According to previous study, we are addressing on Content Only (CO) or only keywords search. In this year, we move forward to study the Content and Structure (CAS) for the Data Centric track of INEX. Furthermore, we presented the structure weight function for optimize the performance of our system.

This paper is organized as follows; Section 2 reviews related works. Section 3 explains the implementation of our system overview and new structure weight algorithm. Section 4 show the experiment, Section 5 explains the result and discussion, conclusions and further work are drawn in Section 6.

2 Related Work

In this section, we provide some historical perspectives on areas of XML research that have influenced to this article as follows.

2.1 XML Data Models

The basic XML data model [1] is a labeled, ordered tree. Fig. 1 shows the data tree of an XML document based on the node-labeled model. There are basically three types of nodes in a data tree as follows.

Element nodes correspond to tags in XML documents, for example, the *body* and *section* nodes.

Attribute nodes correspond to attributes associated with tags in XML documents, for example, the *id* node. In contrast to element nodes, attribute nodes are not nested (that is, an attribute cannot have any sub-elements), not repeatable (that is, two same-name attributes cannot occur under one element), and unordered (that is, attributes of an element can freely interchange their occurrence locations under the element).

Leaf nodes (i.e., text nodes) correspond to the data values in XML documents, for example, the *xml* node.

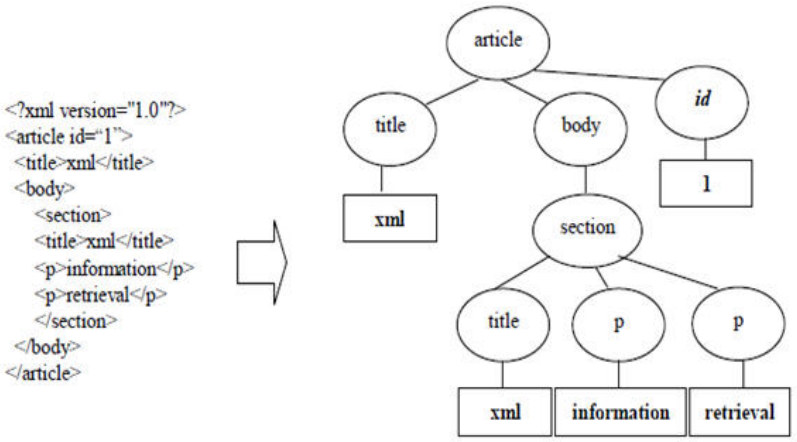


Fig. 1. The Example of XML Element Tree

3 Methods

3.1 An Implementation of XML Retrieval System

The More Efficient XML Information Retrieval (MEXIR) [2] is based on the leaf-node indexing scheme that uses a relational DBMS as a storage back-end. We discuss the schema setup using MySQL [3] and the full-text engine Sphinx [4], [5] with the MySQL dumps function.

For the initial step, we consider a simplified XML data model, but we disregard Meta mark-up such as, comments, links and attributes. In Fig. 2, depicts the overview of XML retrieval system. The main components of the MEXIR retrieval system are follows:

1. When new documents are entered, the ADXPI indexer parses and analyzes the tag and position to build a list of indices.
2. The ecADXPI compressor analyzes the tag and position to build the structure index, which is stored in the MySQL database.
3. The AutoMix analyzes the tag and position to build the SW index, which is stored in the MySQL database.
4. The Sphinx is used to analyze and build all full text indices.
5. The Score Sharing function is used to assign parent scores by sharing scores from leaf nodes to their parents using a top-down approach.
6. The Double Scoring function is used to adjust the leaf-node scores based on linear recombination.

3.2 Structure Weight Function

Our structural scoring model essentially counts the number of navigational (i.e., tag name-only) query conditions that are satisfied by a result candidate and thus connect the content conditions matched for the user queries. It assigns C for every navigational condition that is matched a part of an absolute path. When matching the structural constraints against the document tree, we calculate structural scoring using 2^c and recomputed the leaf element score as following:

$$LeafScore(Node) \leftarrow LeafScore(Node) * 2^c \tag{1}$$

Note that;

c is the frequency of navigational condition that is matched a part of an absolute path

4 Experiment Setup

In this section, we present and discuss the results based on the INEX collection. We also present the results of an empirical sensitivity analysis of various parameter performed on a Wikipedia collection. This experiment was performed on Intel Pentium i5 4 * 2.79 GHz with 6 GB of memory, Microsoft Windows 7 Ultimate 64-bit Operating System and Microsoft Visual C♯.NET 2008.

4.1 INEX Collections

1. On the Snippet retrieval track, the document collections are from the INEX-Wiki09 collection was created from the October 8, 2008 dump of English

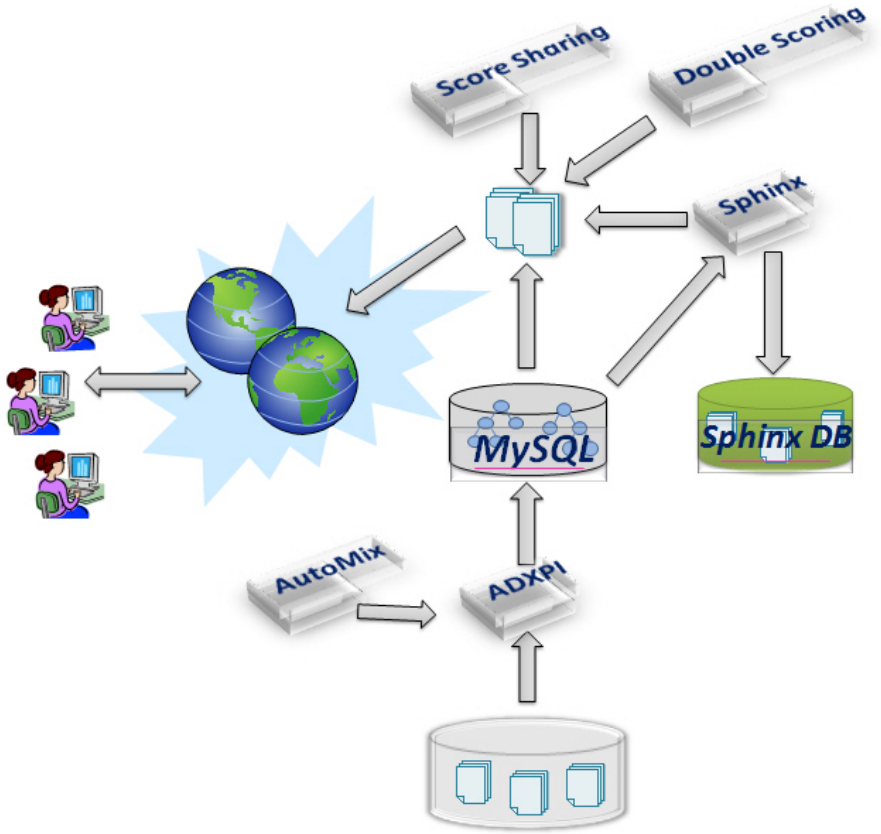


Fig. 2. MEXIR XML Retrieval System Overvie

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.

2. 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.
3. 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.

5 Results and Discussion

5.1 Snippet Retrieval Track

5.2 Data Centric Track

5.3 Web Service Discovery Track

6 Conclusions

References

1. Bray, T. et al, Markup Language (XML) 1.1 (Second Edition). Available Source: `http://www.w3.org/TR/xml11/` (2006).
2. Wichaiwong T. and Jaruskulchai C., MEXIR: An Implementation of High Performance and High Precision XML Information Retrieval, Computer Technology and Application, David Publishing Company, Volume 2, Issue No. 4, April. (2011).
3. Hinz, S. et al, MySQL Full-Text Search Functions, Available Source: `http://dev.mysql.com/doc/refman/5.1/en/fulltext-search.html` (Current 2009).
4. Aksyonoff, A. et al, Sphinx Open Source Search Server, Available Source: `http://www.sphinxsearch.com/` (Current 2009).
5. Aksyonoff, A. Introduction to Search with Sphinx, O'Reilly Media. (2011).

Overview of the INEX 2011 Question Answering Track (QA@INEX)

Eric SanJuan[1], Véronique Moriceau[2], Xavier Tannier[2], Patrice Bellot[1], and Josiane Mothe[3]

[1] LIA, Université d'Avignon et des Pays de Vaucluse (France)
{patrice.bellot,eric.sanjuan}@univ-avignon.fr
[2] LIMSI-CNRS, University Paris-Sud (France)
{moriceau,xtannier}@limsi.fr
[3] IRIT, Universtité de Toulouse (France)
josiane.mothe@irit.fr

Abstract. The INEX QA track (QA@INEX) aims to evaluate a complex question-answering task. In such a task, the set of questions is composed of complex questions that can be answered by several sentences or by an aggregation of texts from different documents. Question-answering, XML/passage retrieval and automatic summarization are combined in order to get closer to real information needs. Based on the groundwork carried out in 2009-2010 edition to determine the sub-tasks and a novel evaluation methodology, the 2011 edition of the track is contextualizing tweets using a recent cleaned dump of the Wikipedia.

Key words: Question Answering, Automatic Summarization, Focus Information Retrieval, XML, Natural Language Processing, Wikipedia

1 Introduction

The QA task to be performed by the participating groups of INEX 2011 is contextualizing tweets, i.e. answering questions of the form "what is this tweet about?" using a recent cleaned dump of the Wikipedia. The general process involves:

- tweet analysis,
- passage and/or XML element retrieval,
- construction of the answer.

We regard as relevant passage segments that both contain relevant information but also contain as little non-relevant information as possible (the result is specific to the question).

For evaluation purposes, we require that the answer uses only elements or passages previously extracted from the document collection. The correctness of answers is established by participants exclusively based on the support passages and documents.

The paper is organized as follows. Section 2 presents the description of the task. Section 3 details the collection of questions and documents. Section 4 describes the baseline system provided by the track organizers. Section 5 presents the techniques and tools used for manual evaluation and explains the final choice of metrics. Full results will be given during the workshop and published in the final version of INEX proceedings.

2 Task description

The underlying scenario is to provide the user with synthetic contextual information when receiving a tweet with an url on a small terminal like a phone. The answer needs to be built by aggregation of relevant XML elements or passages grasped from a local XML dump of the Wikipedia.

The aggregated answers will be evaluated according to the way they overlap with relevant passages (number of them, vocabulary and bi-grams included or missing) and the "last point of interest" marked by evaluators. By combining these measures, we expect to take into account both the informative content and the readability of the aggregated answers.

Each assessor will have to evaluate a pool of answers of a maximum of 500 words each. Evaluators will have to mark:

- The "last point of interest", *i.e.* the first point after which the text becomes out of context because of:
  - syntactic incoherence,
  - unsolved anaphora,
  - redundancy,
  - not answering the question.
- All relevant passages in the text, even if they are redundant.

Systems will be ranked according to:

- The length in words (number of words) from the beginning of the answer to the "last point of interest",
- The distributional similarities between the whole answer and the concatenation of all relevant passages from all assessors using the FRESA package[4] which computes both Kullback Leibler (KL) and Jenssen-Shanon (JS) divergences between n-grams ($1 \leq n \leq 4$).

3 Test data

3.1 Questions

The question data set is composed of 132 topics. Each topic includes the title and the first sentence of a New York Times paper that were twitted at least two months after the Wikipedia dump we use. An example is provided below:

[4] http://lia.univ-avignon.fr/fileadmin/axes/TALNE/Ressources.html

```
<topic id="2011005">
<title>Heat Wave Moves Into Eastern U.S</title>
<txt>The wave of intense heat that has enveloped much of the
central part of the country for the past couple of weeks is
moving east and temperatures are expected to top the 100-degree
mark with hot, sticky weather Thursday in cities from
Washington, D.C., to Charlotte, N.C.</txt>
</topic>
```

For each topic, we manually checked that there is related information in the document collection.

3.2 Document collection

The document collection has been built based on a recent dump of the English Wikipedia from April 2011. Since we target a plain XML corpus for an easy extraction of plain text answers, we removed all notes and bibliographic references that are difficult to handle and kept only the 3,217,015 non empty Wikipedia pages (pages having at least one section).

Resulting documents are made of a title (`title`), an abstract (`a`) and sections (`s`). Each section has a sub-title (`h`). Abstract and sections are made of paragraphs (`p`) and each paragraph can have entities (`t`) that refer to other Wikipedia pages. Therefore the resulting corpus has this simple DTD:

```
<!ELEMENT xml (page)+>
<!ELEMENT page (ID, title, a, s*)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT title (#PCDATA)><!ELEMENT a (p+)>
<!ELEMENT s (h, p+)>
<!ATTLIST s o CDATA #REQUIRED>
<!ELEMENT h (#PCDATA)>
<!ELEMENT p (#PCDATA | t)*>
<!ATTLIST p o CDATA #REQUIRED>
<!ELEMENT t (#PCDATA)>
<!ATTLIST t e CDATA #IMPLIED>
```

For example:

```
<?xml version="1.0" encoding="utf-8"?>
<page>
<ID>2001246</ID>
<title>Alvin Langdon Coburn</title>
<s o="1">
<h>Childhood (1882-1899)</h>
<p o="1">Coburn was born on June 11, 1882, at 134 East Springfield
Street in <t>Boston, Massachusetts</t>, to a middle-class family.
His father, who had established the successful firm of
Coburn &amp;amp; Whitman Shirts, died when he was seven. After that he
```

```
was raised solely by his mother, Fannie,  who remained the primary
influence in his early life, even though she remarried when he was
a teenager. In his autobiography, Coburn wrote, &quot;My mother was
a remarkable woman of very strong character who tried to dominate
my life. It was a battle royal all the days of our life
together.&quot;</p>
<p o="2">In 1890 the family visited his maternal uncles in
Los Angeles, and they gave him a 4 x 5 Kodak camera. He immediately
fell in love with the camera, and within a few years he had developed
a remarkable talent for both visual composition and technical
proficiency in the <t>darkroom</t>. (...)</p>
(...)
</page>
```

A complementary list of non Wikipedia entities has also been made available. The named entities (person, organisation, location, date) of the document collection have been tagged using XIP [1]. For example, for the previous documents, the extracted named entities are:

```
Alvin Langdon Coburn
1882-1899
Coburn
June 11, 1882
134 East Springfield Street
Boston, Massachusetts
Coburn Whitman
Fannie
Coburn
1890
Los Angeles
Kodak
```

This can be used for participants willing to use named entities in texts but not having their own tagger.

3.3 Submission format

Participants can submit up to 3 runs. One run out of the 3 should be completely automatic, using only available public resources. Submitted XML elements and/or passages up to 500 words in total. The passages will be read top down and only the 500 first words will be considered for evaluation and may not be overlapping for the same topic (we consider as a single word any string of alphanumeric characters without space or punctuation).

The format for results is a variant of the familiar TREC format with additional fields:

```
<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 <id> 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 participant group and for the method used.

The remaining three columns depend on the chosen format (text passage or offset).

For textual context, 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. Here is an example of such an output:

```
 1 Q0 3005204 1 0.9999 I10UniXRun1 The Alfred Noble Prize is an award
presented by the combined engineering societies of the United States,
given each year to a person not over thirty-five for a paper published
in one of the journals of the participating societies.
 1 Q0 3005204 2 0.9998 I10UniXRun1 The prize was established in 1929 in
honor of Alfred Noble, Past President of the American Society of Civil
Engineers.
 1 Q0 3005204 3 0.9997 I10UniXRun1 It has no connection to the Nobel
Prize , although the two are often confused due to their 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 I10UniXRun1 256 230
1 Q0 3005204 2 0.9998 I10UniXRun1 488 118
1 Q0 3005204 3 0.9997 I10UniXRun1 609 109
```

The 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 239 characters.

4 Baseline system

A baseline XML-element retrieval/summarization system has been made available for participants.

4.1 Online interface

The system is available online through a web interface[5] that allows to query:

[5] http://qa.termwatch.es

- an index powered by Indri[6] that covers all words (no stop list, no stemming) and all XML tags.
- a PartOfSpeech tagger powered by TreeTagger[7].
- a baseline summarization algorithm powered by TermWatch[8] used in [2].
- a summary content evaluation based on FRESA[3].

Three kind of queries can be used. Standard bag of words, sets of muti-word terms or more complex structured queries using Indri Language[4]. The system returns the 50 first documents retrieved by Indri in raw text format, PoS tagged text or XML document source.

It is also possible to get a summary of these retrieved documents powered by TermWatch which is based like most of the state of art sentence ranking algorithms [5] on a PageRank approach. Given a square stochastic matrix M derived from the matrix of term co-occurrences in text sentences, and a term weight vector $\overrightarrow{x}$, they compute $M^n\overrightarrow{x}$ for $n \approx 30$ as an approximation of the first eigenvector $\overrightarrow{e_1}$. $\overrightarrow{e_1}$ is then used to score and rank sentences. The idea is to progressively weight the sentences according to the number of sentences in their neighborhood (sentences sharing at least one word). The algorithm converges towards the same solution no matter the initial weights on vertex. In this baseline we stop the iterative process at $n = 2$ and only nominals (nouns or adjectives) are considered in sentences.

The resulting summary is evaluated against retrieved documents using Kullback-Leibler (KL) measure. The web interface also allows the user to test its own summary against the same set of documents. The system also gives a lower bound using a random set of 500 words extracted from the texts and an upper bound using an empty summary. Random summaries naturally reaches the closest word distribution but they are clearly unreadable.

4.2 Application Programming Interface

A perl API running on linux and using wget to query the server was also made available. By default this API takes as input a tabulated file with three fields: topic names, selected output format and query. The output format can be the baseline summary or the 50 first retrieved documents in raw text, PoS tagged or XML source. An example of such file allowing to retrieve 50 documents per topic based on their title was also released.

5 Evaluation

Due to the extension of run submission deadline, evaluation is not fully achieved. Results will be given during the workshop and published in the final version of INEX proceedings.

This section describes the techniques and metrics used to perform the evaluation.

[6] http://www.lemurproject.org/
[7] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/
[8] http://termwatch.es

5.1 Submitted runs

23 valid runs by 11 teams from 6 countries (France, Spain, Mexico, India, Canada, Brasil) were submitted. All runs are in raw text format and almost all participants used their own summarization system. Only three participants did not use the online Indri IR engine. Participants using the perl API generated queries based on both title and phrase fields using various query expansion techniques. Only one used XML tags.

The total number of submitted passages is 37,303. The median number of distinct passages per topic is 284.5 and the median length in words is 26.9. This relative small amount of distinct passages could be due to the fact that most of the participants used the provided Indri index with its perl API.

5.2 Pooling

It was planned to perform the evaluation by assessing participant runs by means of pooling technique. When relevant RSVs are provided by participants, passages are sorted by their RSVs. In other cases (RSVs are often all the same for a given summary), passages are sorted by their frequency through all participants' runs. The 20% best passages per run are then kept, and mixed, so that duplicates are removed. Only these passages are evaluated by organizers. In all, 20% of all passages would have been selected for assessment. It appears that this approach penalizes participants that did not use the Indri index provided by organizers, since some of their passages often only appear once among all runs. Therefore they would have been evaluated only on the passages at the beginning of their summaries, the rest of the summary being considered as non relevant. We finally chose to evaluate all passages on a consistent pool of topics. Each passage informativeness will be evaluated independently from others.

This assessment is intended to be quite generous towards passages. All passages concerning a protagonist of the topic are considered relevant, even if the main subject of the topic is not addressed. The reason is that missing words in the reference can lead to artificial increase of the *divergence*, which is a known and not desirable side effect of this measure.

5.3 Metrics

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. In the QA task, readability of answers [6] 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 [7] and TAC workshops [8]. A mong them, Kullback-Leibler (KL) and Jenssen-Shanon (JS) divergences have been used [9, 3] to evaluate the informativeness of short summaries based on a bunch of highly relevant documents.

In this edition we will use the KL one to evaluate the informative content of answers by comparing their n-gram distributions with those from all assessed relevant passages.

5.4 Interface for manual evaluation

Readability evaluation will rely on participants. Each will have to evaluate a pool of summaries of a maximum of 500 words each on a online web interface.

The interface will allow to mark:

1. The "last point of interest", i.e. the first point after which the text becomes out of context because of: syntactic incoherence, unsolved anaphora, redundancy or not answering the question.
2. all relevant passages in the text, even if they are redundant.

Systems will then be ranked according to the:

- length in words from the beginning of the answer to the "last point of interest".
- distributional similarities between the whole answer and the concatenation of all relevant passages from all assessors using the FRESA package.

Therefore, a second evaluation of summary content informativeness in context will complement the evaluation by organizers done out of context.

6 Conclusion

This tracks that brings together the NLP and the IR communities is getting more attention. The experimented measures used for evaluation based on textual content more than passage offsets seem to reach some consensus between the two communities. Taking into account readability of summary also encourages NLP and linguistic teams to participate. Next edition will start much more earlier, the corpus generation from a wikipedia dump is now completely automatic. We plan to propose a larger variety of questions from twitter, but also from web search engines provided by the CAAS ANR project (Contextual Auto-Adaptive Search). We also would like to encourage XML systems by providing more structured questions with explicit name entities. We also envisage to open the track to terminology extractor systems.

References

1. At-Mokhtar, S., Chanod, J.P., Roux, C.: Robustness beyond shallowness: Incremental deep parsing. Natural Language Engineering **8** (2002) 121–144
2. Chen, C., Ibekwe-Sanjuan, F., Hou, J.: The structure and dynamics of cocitation clusters: A multiple-perspective cocitation analysis. JASIST **61**(7) (2010) 1386–1409
3. Saggion, H., Moreno, J.M.T., da Cunha, I., SanJuan, E., Velázquez-Morales, P.: Multilingual summarization evaluation without human models. In Huang, C.R., Jurafsky, D., eds.: COLING (Posters), Chinese Information Processing Society of China (2010) 1059–1067
4. Metzler, D., Croft, W.B.: Combining the language model and inference network approaches to retrieval. Inf. Process. Manage. **40**(5) (2004) 735–750
5. Fernández, S., SanJuan, E., Moreno, J.M.T.: Textual energy of associative memories: Performant applications of enertex algorithm in text summarization and topic segmentation. In: MICAI. (2007) 861–871
6. Pitler, E., Louis, A., Nenkova, A.: Automatic evaluation of linguistic quality in multi-document summarization. In: ACL. (2010) 544–554
7. Nenkova, A., Passonneau, R.: Evaluating content selection in summarization: The pyramid method. In: Proceedings of HLT-NAACL. Volume 2004. (2004)
8. Dang, H.: Overview of the TAC 2008 Opinion Question Answering and Summarization Tasks. In: Proc. of the First Text Analysis Conference. (2008)
9. Louis, A., Nenkova, A.: Performance confidence estimation for automatic summarization. In: EACL, The Association for Computer Linguistics (2009) 541–548

A Dynamic Indexing Summarizer at the QA@INEX 2011 track

Luis Adrián Cabrera-Diego, Alejandro Molina, Gerardo Sierra

Instituto de Ingeniería, Universidad Nacional Autónoma de México, Torre de Ingeniería, Basamento, Av. Universidad 3000, Mexico City, 04510 Mexico
{lcabrerad, amolinav, gsierram}@iingen.unam.mx

Abstract. In this paper we present a question-answering system developed for the QA task of INEX 2011, where a set of tweets from the New York Times are used as long questions. The developed system indexes dynamically Wikipedia pages to extract a relevant summary. The results reveal that this system can bring related information despite the dump of Wikipedia is older than the tweets from the New York Times. The results obtained were evaluated using a framework for evaluating summaries automatically based on Jensen-Shannon divergence.

Keywords: Question-Answering, QA, Dynamic Indexing, Summarizer, Wikipedia

1 Introduction

Question-answering (QA) is the task of responding automatically an inquiry posed in natural language. Since the last advances in summarization methods and their automatic evaluation [1], several summarization systems have been used in question-answering systems [2, 3]. Typically, the methods employed in these systems, consist in extract the most important sentences from a given static-document collection (see "multi-document" summarization at [4]). However, recent studies have shown that dynamic indexing could be useful in information retrieval tasks [5].

In this paper we propose a question-answer system that dynamically retrieve a document collection and use it to extract the most important information by scoring paragraphs and sentences. The name of this system is *Dynamic Indexing Summarizer for Question-answering* (DISQ).

The utilization of DISQ is done in the track of QA of INEX 2011, where a set of long queries have to be answered with passages of Wikipedia. The inquiries of the track for 2011 are the tweets posted by the New York Time newspaper.

The organization of this paper is as follows, we firstly present the architecture of DISQ. Secondly, we show the experiments, some results and their evaluation. Finally, we give our conclusions and divers future tasks.

2 A Dynamic Indexing Summarizer for Question-Answering

DISQ is a system based on an optimized database created from the XML and SQL dumps of Wikipedia[1]. This database was built using the tools available in the JWPL API[2] [6]. As well, DISQ is based on four main modules, which are described below:

1. Query Preprocessor
2. Dynamic Indexing
3. Paragraph Ranking
4. Index Expansion

The Figure 1 shows a diagram of the system.

The Query Preprocessing module obtains the keywords from the long original query. To do this, DISQ firstly deletes from the query some punctuation marks, like commas, brackets or colons (although, it does not delete dashes, periods or apostrophes, because they could be part of important keywords like *student-teacher* or *U.S.*). Secondly, the system creates n-grams with a length that goes from 1 to 5. Finally, the list of n-grams is cleaned-up (we eliminate the n-grams that begin or end with stop-words). This last preprocessing task is done because we try to reduce the number of word constructions that possibly do not exist in Wikipedia. The final set of n-grams is considered the set of keywords.

The second main module, Dynamic Indexing, tries to elicit an index from the search of the keywords in Wikipedia. If a keyword retrieves an article, the page is added to a *Dynamic Document Collection* (DDC), i.e., a set of article pages of Wikipedia where DISQ will do future searches. As well, the keyword is indexed with the ID of the article page of Wikipedia. If the page retrieved by the keyword is instead a redirection page, the system tries to add new keywords and entries into the DDC. In a first step, the name of the page, where the system is redirected, is added to the keywords list. This is done because we can obtain, from the redirections, some synonyms, lemmas, different spellings, among other information that could be used to make a better search. In a second step, the redirected page is added to the DDC and index with its ID the original search keyword and the new one. A similar case happens when the name of a page has information between brackets, e.g. *Galicia (Spain)*. In this case, the name of the page is divided in two and the extra information is considered a related keyword instead a synonym or lemma. If a keyword is not found in Wikipedia, it is indexed with a special ID (null). In Table 1 we show some examples of new keywords that can be obtained from Wikipedia.

[1] The dumps of Wikipedia used to create the optimized database are from April 5th 2011, following the guidelines of QA@INEX 2011 to use the Wikipedia's dumps of April 2011. The dumps are available in the page of Wikimedia Foundation.

[2] http://code.google.com/p/jwpl/

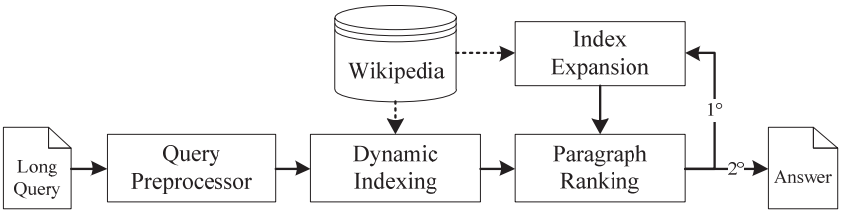


Fig. 1. A general diagram of the system SPRII

Table 1. Examples of the different keywords that can be obtained from Wikipedia using the redirection pages and information between brackets.

Original keyword	New keyword
U.N.	United Nations
Years	Year
Boardgame	Board Game
Obama	Barack Obama
Data (Computing)	Data, Computing

The Paragraph Ranking module starts when the DDC is completed. This module does two tasks at the same time. The first one is to search all the keywords in the paragraphs of each article that belongs to the DDC. The second one is to rank the paragraphs to obtain the most important ones. For the last task, we use the score (1).

$$R_p = \frac{(\sum L_k) F_k}{S_k} \tag{1}$$

Where, R_p is the value of relevance of the paragraph; L_k is the length of the keyword found in the paragraph (size of the n-gram); F_k is the number of keywords found in the paragraph[3] and S_k is the number of keywords searched in the paragraph.

To decrease the number of over-ranked paragraphs, we omit the keyword(s) that retrieved the article which the system is analyzing. Thus, the number of keywords to search in the paragraphs varies in each article.

After each paragraph is ranked, the system finds the highest value of R_p and selects the paragraphs that have the same value of R_p.

If a paragraph from the DDC is not selected or if the selected paragraphs have a value lower than 0.3[4] the system does a new ranking using the score (2) and then selects the most important paragraphs as in the first method.

$$R_p = \frac{\left(\sum f_k L_k\right) F_k}{S_k} \tag{2}$$

[3] This number of n-grams is not the frequency of apparition of them. It is the number of n-grams searched that appeared in the selected paragraph.

[4] This figure was chosen experimentally.

Where f_k is the frequency of the keyword in the paragraph.

In both formulae the size of the keyword is a parameter, because we consider that a paragraphs with larger keywords are going to be more related with the query than one with smaller keywords.

The fourth main module, Index Expansion, expands the DDC to find new related information but at the same time look for new keywords. To do this, the system first extracts the links that belong to the best ranked paragraphs (previous module) and that target to other articles of Wikipedia. For each link extracted, the system retrieves the corresponding article of Wikipedia and adds it to the *Link List*. When the Link List is finished, the system searches in it the keywords and ranks all the paragraphs using the same methods described previously. The articles of the Link List from where the new selected paragraphs come from are added to the DDC. Additionally, the name of the articles, where the new selected paragraphs were obtained, becomes new keywords that are linked to the respective article ID. All the selected paragraphs at this point become the first result, which it is called R12.

We repeat the process of the module Page Ranking one more time, using the new keywords and the updated DDC, to obtain a more refined result. This result is called R3. As well, to do the finals summaries of R12 and R3, we use an implementation of LexRank [7].

3 Experiments and Results

The experiments were done with the set of questions (tweets from the New York Times) that INEX 2011 prepared for the Question-Answering track. From the experiments we got two summaries, the R12 and the R3; both are frequently different.

In order to evaluate our system we use FRESA[5], a framework for evaluating summaries automatically based on Jensen-Shannon divergence [8-10]. The Table 2 shows an example of the evaluation of FRESA, between our summaries and the one from the online-tools of QA@INEX 2011 webpage[6]. The query used for this example was "*Heat Wave Moves Into Eastern U.S"*.

Although the evaluation shows that our summaries are worse than the INEX summary (higher divergences), our summaries are evaluated against summaries obtained from the texts retrieved by the tools QA@INEX 2011 and not with the original article. Thus, we did an evaluation using FRESA between the original article from the Ney York Times[7] and both summaries; the results can be seen in the Table 3.

We observe that DISQ has good results, but the performance against the baseline summary is variable. Nevertheless, our results are better than the random summaries.

---

[5] http://lia.univ-avignon.fr/fileadmin/axes/TALNE/downloads/index_fresa.html

[6] https://inex.mmci.uni-saarland.de/tracks/qa/

[7] The original articles were searched in internet using the tweets from the Ney York Times Newspaper.

Table 2. Example of the evaluation between our system and the baseline system of QA@INEX 2011 for the question "*Heat Wave Moves Into Eastern U.S*"

Distribution type	Unigram	Bigram	With 2-gap	Average
Baseline summary	55.44376	64.12505	64.25264	61.27382
Empty baseline	72.32032	81.33022	81.40719	78.35258
Random unigram	58.29775	67.32914	67.37218	64.33302
Random 5-gram	49.98983	58.64993	58.86784	55.83587
R12	61.28982	70.16201	70.27805	67.24329
R3	65.37411	74.23967	74.36819	71.32732

Table 3. Example of the evaluation with the original article between DISQ and the baseline system of QA@INEX 2011 for the question "*Heat Wave Moves Into Eastern U.S*"

Distribution type	Unigram	Bigram	With 2-gap	Average
Baseline summary	9.00524	16.91866	16.86023	14.26137
R12	8.72437	16.45770	16.26208	13.81472
R3	8.79530	16.11737	15.93532	13.61600

It is important to say, that we saw in some baseline summaries fragments of text that came from a more recent Wikipedia (e.g. from May or June 2011). In our opinion, this can affect the results, because the baseline summaries can be more related to the tweets that were posted in the months after April 2011.

4 Conclusion

We have presented a Question-Answering system that uses n-grams to evaluate paragraphs from Wikipedia articles and to retrieve them as related and relevant information for a certain question. The developed system was used for the track of QA@INEX 2011, where a set of tweets from New York Times newspaper worked as questions of the system.

The experiments have revealed that the system developed can bring related information despite the dump of Wikipedia is older than the tweets from the New York Times. Nevertheless, there are cases where the performance of our system can be very variable. In this case, we think this is due the fact sometimes the tweets have very ambiguous words that cannot be disambiguated by Wikipedia or they are not enough to find a correct relation between them to find a good answer. As well, the experiments have shown that it is not necessary to use cleaned dumps of Wikipedia to retrieve easily and clean information from the encyclopedia.

As future work, we have to develop a better way to find the correct answer when the words of the question are not interrelated. As well, we have to find how to treat the ambiguous terms of Wikipedia; this is to get the correct page and extract from it the possible related information. In addition, we have to develop a module to update the database of Wikipedia automatically.

References

1. San Juan, E., Bellot, P., Moriceau, V., Tannier, X.: Overview of the INEX 2010 Question-Answering Track (QA@INEX). Lecture Notes in Computer Science, Comparative Evaluation of Focused Retrieval. 9th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2010), Revised and Selected Papers, vol. 6932, pp. 269-281, Springer (2011).
2. Soriano-Morales, E.P., Medina-Urrea, A., Sierra, G., Méndez-Cruz, C.F.: The GIL-UNAM-3 summarizer: an experiment in the track QA@INEX'10. Lecture Notes in Computer Science, Comparative Evaluation of Focused Retrieval. 9th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2010), Revised and Selected Papers, vol. 6932, pp. 269-281, Springer (2011).
3. Torres-Moreno, J.M.: The Cortex automatic summarization system at the QA@INEX track 2010. Lecture Notes in Computer Science, Comparative Evaluation of Focused Retrieval. 9th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2010), Revised and Selected Papers, vol. 6932, pp. 269-281, Springer (2011).
4. Torres-Moreno, J.M.: Résumé automatique de documents une approche statistique. Ed. Hermes Lavoisier (2011).
5. Strohman, T.: Dynamic collections in Indri. Technical Report IR-426, University of Massachusetts Amherst (2005).
6. Zesch, T., Müller, C., Gurevych, I.: Extracting lexical semantic knowledge from Wikipedia and Wiktionary. In: 6th Conference on Language Resources and Evaluation (LREC), Marrakech (2008).
7. Erkan, G., Radev, D.R.: LexRank: Graph-Based Lexical Centrality As Salience in Text Summarization. Artificial Intelligence Research, vol. 22, pp. 457-479 (2004).
8. Saggion, H., Torres-Moreno, J.M., da Cunha, I., SanJuan, E., Velásquez-Morales, P.: Multilingual Summarization Evaluation without Human Models. In: 23rd International Conference on Computational Linguistics (COLING), Beijing (2010).
9. 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, vol. 42 (2010).
10. Torres-Moreno, J.M., Saggion, H., da Cunha, I., Velázquez-Morales, P., SanJuan, E.: Évaluation automatique de résumés avec et sans reference. In: 17e Conférence sur le Traitement Automatique des Langues Naturelles (TALN), Montreal (2010).

IRIT at INEX: Question answering task

Liana Ermakova, Josiane Mothe

Institut de Recherche en Informatique de Toulouse
118 Route de Narbonne, 31062 Toulouse Cedex 9, France

Abstract. This paper presents the approach we developed in the context of INEX QA track. The INEX QA track considers a tweet as an input and a summary up to 500 words as an output. There are 132 tweets which include the title and the first sentence of a New York Times articles. The summary should be made of extracted relevant sentences from April 2011 English Wikipedia dump, totally 3 217 015 non-empty pages. The approach we developed at IRIT considers first the 20 top documents retrieved by a search engine (Terrier using default settings). Those documents and tweets are then parsed using Stanford CoreNLP tool. We then computed indices for each section and each sentence. The index includes not only word forms and lemmas, but also NE. Sentence retrieval is based on standard TF-IDF measure enriched by named entity recognition, POS weighting and smoothing from local context. Three runs were submitted with different parameter settings.

Keywords: Information retrieval, question answering, contextual information, smoothing, part of speech tagging, name entity.

1 Introduction

INEX (Initiative for the Evaluation of XML Retrieval) is an evaluation forum for XML IR. It provides large structured test collections and scoring methods for IR system evaluation.

Question answering track (QA@INEX) is one of the INEX research task and aims at investigating how semi-structured data can be used to find real-world answer to a question in natural language (1).

QA task is a challengeable problem of IR. The major tasks of QA systems are searching for the relevant information and generation of answers.

In 2011 the INEX QA track aims at evaluating summarization systems. Summary is 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" (2). Summaries are either "extracts", if they contain the most important sentences extracted from the original text, or "abstracts", if these sentences are re-written or paraphrased, generating a new text (3).

In 2011 the INEX QA track considers a tweet as an input and a summary up to 500 words as an output. The motivation for the task is to contextualize tweets received on small terminals like phones by information from a local XML dump of Wikipedia. There are 132 tweets which include the title and the first sentence of a New York Times articles. The summary should be made of extracted relevant sentences from April 2011 English Wikipedia dump, totally 3 217 015 non-empty pages. XML

documents are made of title, abstract and sections. Each section has a sub-title. Abstract and sections are made of paragraphs and each paragraph can have entities that refer to Wikipedia pages. Results are presented in TREC-like format where each row corresponds to the retrieved sentence ranked by score. The systems will be evaluated by two criteria: relevance of the retrieved information and readability of the presented results. (https://inex.mmci.uni-saarland.de/tracks/qa/).

Usually a QA system includes following stages (4):

1. Transform the questions into queries
2. Associate queries to a set of documents
3. Filter and sort these documents to calculate various degrees of similarity
4. Identify the sentences which might contain the answers
5. Extract text fragments from them which constitute the answers

Similarly, our approach consists in:

1. Document retrieval by a search engine (documents are sorted by a search engine according to a similarity measure) for each tweets
2. Parsing of tweets and sets of retrieved documents
3. Index construction for sentences
4. Calculation of similarity for sentences and sentence sorting
5. Getting top n sentences which have the highest score and contain no more than 500 tokens in total

This approach we propose in IRIT runs includes two steps:

1. Preprocessing
2. Searching for relevant sentences.

Preprocessing consists of the three previous first stages, namely searching for the documents similar to queries, parsing and re-indexing.

The proposed techniques for relevant sentence searching is standard TF-IDF measure enriched by named entity recognition, POS weighting and smoothing from local context. Moreover, there is a possibility to assign different weights to abstracts and sections of Wikipedia pages, definitional sentences, and various labels.

The two next sections present these steps.

2 Preprocessing

The first step of our approach is preprocessing of texts.

We first looked for the documents similar to the queries. For this stage, document retrieval was performed by the Terrier Information Retrieval Platform (http://terrier.org/), an open-source search engine developed by the School of Computing Science, University of Glasgow. To this end we transformed queries into to the format accepted by Terrier. We used the default settings. Terrier provides up to 1000 documents as a result, however we considered just the top 20 which are the most similar to the query. Each document retrieved by Terrier was then split into ordered sections. Sections were considered as different documents since usually Wikipedia articles consists of sections related to the topic but not connected to each other.

Moreover it gives the possibility to keep relative sentence order to increase readability if we would like to generate coherent text from them. New documents have an attribute IS_ABSTRACT. Our system provides the possibility to set different weights to abstracts and sections. By default it is assumed that the weight of an abstract is 1.0 and the weight of other sections is 0.8.

The second stage of preprocessing is parsing of tweets and retrieved texts by Stanford CoreNLP developed by the Stanford Natural Language Processing Group. CoreNLP integrates such tools as POS tagger, Named Entity Recognizer, parser and the co-reference resolution system (http://nlp.stanford.edu/software/corenlp.shtml). It produces an XML document as an output. It uses the Penn Treebank tag set (5). In our approach, tweets were transformed into queries with POS tagging and recognized named entities (NE). It allows taking into account different weights for different tokens within a question, e.g. NE are considered to be more important than common nouns; nouns are more significant than verbs; punctuation marks are not valuable, …

The preprocessing ends with indexing. We computed indices for each section and each sentence. The index includes not only word forms and lemmas, but also NE.

3 Sentence Retrieval

"Sentence Retrieval is the task of retrieving a relevant sentence in response to a query, a question, or a reference sentence" (6).

The general idea of the proposed approach is to compute similarity between the query and sentences and to retrieve the most similar passages. To this end we used standard TF-IDF measure. We extended the basic approach by adding weight coefficients to POS, NE, headers, sentences from abstracts, and definitional sentences. Moreover sentence meaning depends on the context. Therefore we used an algorithm for smoothing from the local context. The sentences were sorted. The sentences with the highest score were added to the summary until the total number of words exceeds 500.

In the implemented system there is a possibility to choose one of the following similarity measures:

1. Cosine similarity:

$$similarity(Q,S) = \frac{\sum_{i=1}^{n} Q_i \times S_i}{\sqrt{\sum_{i=1}^{n}(Q_i)^2} \times \sqrt{\sum_{i=1}^{n}(S_i)^2}} \quad \text{(I)}$$

2. Dice similarity:

$$similarity(Q,S) = \frac{2\sum_{i=1}^{n} Q_i \times S_i}{\sum_{i=1}^{n}(Q_i)^2 + \sum_{i=1}^{n}(S_i)^2} \quad \text{(II)}$$

3. Jaccard similarity:

$$similarity(Q,S) = \frac{\sum_{i=1}^{n} Q_i \times S_i}{\sum_{i=1}^{n}(Q_i)^2 + \sum_{i=1}^{n}(S_i)^2 - \sum_{i=1}^{n} Q_i \times S_i} \quad \text{(III)}$$

where Q is a query, S is a sentence, Q_i is the occurrence of the i-th token in a query and S_i is the occurrence of the i-th token in a sentence. If the token is not presented in the query or in the sentence, Q_i (resp. S_i) is equal to 0 respectively.

We took into account only lexical vocabulary overlap between a query and a document. However it is possible also to consider morphological and spelling variants, synonyms, hyperonyms, …

Different words should not have the same weight, e.g. usually it is better not to take into account stop-words. Our system provides several ways to assign score to words. The first option is to identify stop-words by frequency threshold. The second way is to assign different weights to different parts of speech. When the option Rank_POS is true each vector component is multiplied by this rank, e.g. determiners have zero weight, proper names have the highest weight equal to 1.0, and nouns have greater weight than verbs, adjectives and adverbs. Another option gives a possibility to consider or not IDF.

NE comparison is hypothesized to be very efficient for contextualizing tweets about news. Therefore for each NE in query we searched corresponding NE in the sentences. If it is found the whole similarity measure is multiplied by NE coefficient computed by the formula:

$$NE_{COEF} = weight(NE) \times \frac{NE_{common} + 1}{NE_{query} + 1}, \quad \text{(IV)}$$

where $weight(NE)$ is floating point parameter given by a user (by default it is equal to 1.0), NE_{common} is the number of NE appearing in both query and sentence, NE_{query} is the number of NE appearing in the query. We used Laplace smoothing to NE by adding one to the numerator and the denominator. The sentence may not contain a NE from the query and it can be still relevant. However, if smoothing is not performed the coefficient will be zero. NE recognition is performed by Stanford CoreNLP. We considered only the exact matches of NE. Synonyms were not identified. However, it may be done later applying WordNet, which includes major NE.

We consider that *Headers*, *labels*, …. should not be taken into account since they are not "good" sentences for summarization. Therefore we assign them lower weights. Stanford parser allows making distinction between auxiliary verbs and main verbs, personal and impersonal verb forms. We assumed that such kinds of sentences do not have personal verbs. One of the settings allows assigning weights to sentences without personal verb forms. By default this parameter is equal to 0. Sentences with personal verb forms have the weight equal to 1.0.

As it has already been mentioned, it is possible to give smaller weights to sections than to abstracts. By default we assume that sections have the weight equal to 0.8 and for abstracts this parameter is 1.0.

Since sentences are much smaller than documents general document IR systems provides worse results to sentence retrieval. Moreover, document retrieval systems are based on the assumption that the relevant document is about the query. However this is not enough for sentence retrieval, e.g. in QA systems the sentence containing the answer is much more relevant that the sentence which is about the subject. General approach to document IR is underlined by TF-IDF measure. In contrast, usually the number of each query term in a sentence is no more than one (6).

Traditionally, sentences are smoothed by the entire collection, but there exist another approach namely smoothing from local context (6).

This method assigns the same weight to all sentences from the context. In contrast, we assume that the importance of the context reduces as the distance increases. So, the nearest sentences should produce more effect on the target sentence sense than others. For sentences with the distance greater than k this coefficient is zero. The total of all weights should be equal to one.

The system allows taking into account k neighboring sentences with the weights depending on their remoteness from the target sentence. In this case the total target sentence score R_t is a weighted sum of scores of neighboring sentences r_i and the target sentence r_0 itself:

$$R_t = \sum_{i=-k}^{k} w_i \times r_i$$	(V)
$$w_i = \begin{cases} \frac{1-w_t}{k+1} \times \frac{k-\lvert i \rvert}{k}, 0 < \lvert i \rvert \le k \\ w_t, i = 0 \\ 0, \quad \lvert i \rvert > k \end{cases}$$	(VI)
$$\sum_{i=-k}^{k} w_i = 1$$	(VII)

where w_t is a target sentence weight set by a user, w_i are weights of the sentences from k context. The weights become smaller as the remoteness increases (see Figure 1). If the sentence number in left or right context is less than k, their weights are added to the target sentence weight w_t. This allows keeping the sum equal to one. By default, $k = 1$, target sentence weight is equal to 0.8.

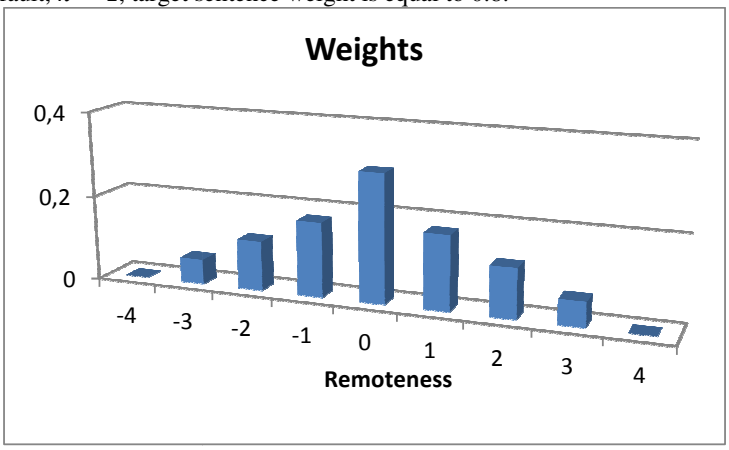


Figure 1. Weighting for smoothing from local context

We assumed that definitional sentences are extremely important to contextualizing task. Therefore they should have higher weights. We have taken into account only definitions of NE by applying the following linguistic pattern:

$$< NE >< Be_{pers} >< NounPhrase >.$$

Be_{pers} is a personal form of the verb *to be*. Noun phrase recognition is also performed by Stanford parser. We considered only sentences occurred in abstracts since they contain more general and condensed information and usually includes definitions in the first sentence. However, the number of extracted definitions was quite small and therefore we did not use them in our runs.

For the first run we used default settings, namely:

1. NE were considered with a coefficient 1.0;
2. Abstract had weight equal to 1.0, sections had score 0.8;
3. Headers, labels, … were not taken into account;
4. We did not consider stop-words;
5. Cosine similarity was applied;
6. Parts of speech were ranked;
7. Each term frequency was multiplied by IDF.

In the second run we changed the similarity measure to Dice similarity. The section weight was reduced to 0.7. The context was extended to two sentences in each direction and the target sentence weight was equal to 0.7. For NE we kept the weight equal to 1.0.

In the third run we applied Jaccard similarity measure and we set the weight to sections equal to 0.5.

4 Conclusion and discussion

The proposed method is based on computing similarity between the query and sentences using an extended TF-IDF measure. We enhance the basic approach by adding weight coefficients to POS, NE, headers, sentences from abstracts, and definitional sentences. Moreover the algorithm for smoothing from local context is provided. The sentences with the highest score are added to the summary until the total number of words exceeds 500.

Future work includes sentence clustering to exclude redundant information, query expansion by WordNet and applying additional syntactic features.

5 References

1. **Eric SanJuan, Patrice Bellot, Veronique Moriceau, Xavier Tannier.** Overview of the 2010 QA Track: Preliminary results. 2010.

2. **Horacio Saggion, Guy Lapalme.** Generating Indicative-Informative Summaries with SumUM. *Association for Computational Linguistics.* 2002.

3. **Jorge Vivaldi, Iria da Cunha, Javier Ramırez.** The REG summarization system at QA@INEX track 2010. 2010.

4. **Juan-Manuel Torres-Moreno, Michel Gagnon.** The Cortex automatic summarization system at the QA@INEX track 2010. 2010.

5. **Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz.** Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics.* 1993, Vol. 19, 2.

6. **MURDOCK, VANESSA GRAHAM.** ASPECTS OF SENTENCE RETRIEVAL. *Dissertation.* 2006.

7. **Jay M. Ponte, W. Bruce Croft.** A Language Modeling Approach to Information Retrieval. *Proceedings of the 21st Annual Conference on Research and Development in Information Retrieval (ACM SIGIR).* 1998.

8. **Franz Josef Och, Hermann Ney.** Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL).* 2002.

9. **Abraham Ittycheriah, Martin Franz, Salim Roukos.** IBM's Statistical Question Answering System - TREC-10. *Proceedings of the Tenth Text Retrieval Conference (TREC).* 2001.

10. **Buchholz, Sabine.** Using grammatical relations, answer frequencies and the World Wide Web for TREC question answering. *Proceedings of the Tenth Text Retrieval Conference (TREC).* 2001.

11. **Hristo Tanev, Milen Kouylekov, Bernardo Magnini.** Combining linguistic processing and web mining for question answering: ITC-irst at TREC-2004. *Proceedings of the Thirteenth Text Retrieval Conference (TREC).* 2004.

12. **Deepak Ravichandran, Eduard Hovy.** Learning surface text patterns fora question answering system. *Proceedings of the ACL Conference.* 2002.

13. **Michael Kaisser, Tilman Becker.** Question answering by searching large corpora with linguistic methods. *Proceedings of the Thirteen Text Retrieval Conference (TREC).* 2004.

14. **Stanley F. Chen, Joshua Goodman.** An Empirical Study of Smoothing Techniques for Language Modeling. 1998.

15. **Edmundo-Pavel Soriano-Morales, Alfonso Medina-Urrea, Gerardo Sierra, Carlos-Francisco Mendez-Cruz.** The GIL-UNAM-3 summarizer: an experiment in the track QA@INEX'10. 2010.

16. **Andrea Carneiro Linhares, Patricia Velazquez.** Using Textual Energy (Enertex) at QA@INEX track 2010. 2010.

Overview of the 2011 QA Track: Querying and Summarizing with XML

Killian Janod, Olivier Mistral

LIA, Université d'Avignon et des Pays de Vaucluse (France)
{killian.janod,olivier.mistral}@etd.univ-avignon.fr

Abstract. In this paper, we describre our participation in the INEX 2011 Question Answering track. There is many ways to answer this track, we focused on how to use meta-data XML in the summarizing process. Firstly, we used XML in the querying process in order to get the most relevant document thanks to the Indri search engine. Secondly, XML enabled us to find sentences that are useful in our probabilistic summarizing process.

Keywords: XML, INEX, QA track, Indri

1 Introduction

For this first participation at the Initiative for the Evaluation of XML retrieval [1], we started working on this track and we thought about this: How can a summarizer answers a question when the corpus which it works on doesn't have this answer? We thought the summarizer have to select only very relevant documents in the Wikipedia and work with them. Our summarizer is able to do so, thanks to the tools given to work on this track :

- An XML document which contains the 132 topics of the track of the form : id topic, title and first sentence of the news,
- A recent cleaned dump of the Wikipedia[1] whose pages are plain XML corpus,
- A baseline XML-element retrieval system powered by Indri [2].

We obviously understood that we have to improve the most we can our using of the XML-element retrieval system. We wanted to try a summarizer which do not only use a probabilistic method. To do so we needed an other way to have information about the corpus. At his point using XML was logical.

Therefore we decided to focus on the two following tasks to carry out this track with such limited time:

1. The improvement of requests with XML,
2. The integration of XML in the summarizer.

The rest of the paper is organized as follows. In Section 2, we present the improvements we did on the requests with XML. Then, we develop the summarizer. Finally, the Section 3 is a conclusion on our works.

[1] http://qa.termwatch.es/data/enwiki-latest-pages-articles.xml.gz
[2] http://qa.termwatch.es/qa.html

2 Improved queries with XML

2.1 Indri [3]

For this track, we experimented an XML-element retrieval system powered by Indri. Indri is a search engine that provides state-of-the-art text search and a rich structured query language for text collections. We relied on three elements of this language which are the following operators: *#combine*, *#weight* and *#1*:

- *#combine* is a query likelihood method used for retrieval in ad-hoc retrieval, the results are ranked by :

$$P(Q|D) = \Pi_q P(q_i|D)^{1/|Q|} \quad (1)$$

  It's possible to use [section] in order to evaluate a query only with the section context, for example: "*#combine[title](query)*" will return titles as results, and rank by query. By default, an Indri query without operators uses the *#combine* on all terms.
- *#weight*: with this operator, we can assign varying weights to certain expressions. This allows us to control how much of an impact each expression within our query has on the final score,
- *#1*: For compound words such as Comic-con for example, this operator allows us to keep the order of the words and within one term of each other.

Combined with the use of XML, this syntax is the heart of the queries to express.

2.2 Methodology

Here is the diagram of the method we used:

The first step in improving queries is to extract the topic of the XML file containing the 132 topics. As this file is XMLised, extraction becomes easy and we end up with a topic like this:

```
<xml>
  <topic id="numId">
    <title>tweet title</title>
    <txt>first sentence of the news</txt>
  </topic>
</xml>
```

Then we clean the topic of everything that is not a word. We clean unnecessary spaces and we use a stop list to keep only the most relevant unigrams.

The fact that the text of the topic is the first sentence of the news is very important, because it has statistically more weight than others. So there is more chance of finding key words in that sentence. Thus, with our clean topic, we

[3] http://www.lemurproject.org/indri.php

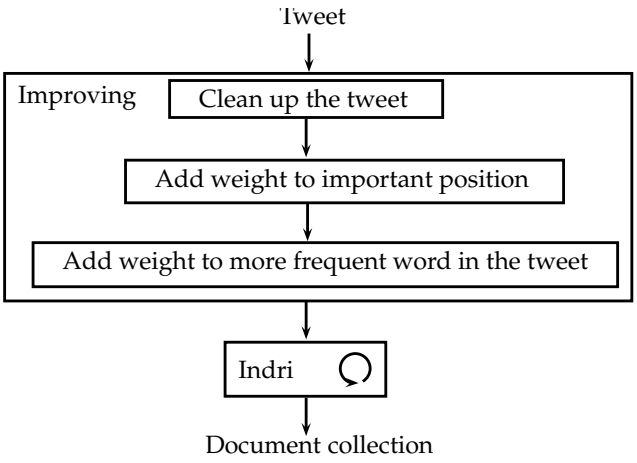


Fig. 1. Improving queries diagram

count the number of iterations of unigrams or compound words by hyphens of the title in both title and text. This will allow us to give weight to the unigrams of the title. For each one, we search them in abstracts and sections and for the two most weighted unigrams (weight must be greater than 1). We also search them in titles. At last, we put in all unigrams of the text non included in the title in a general #combine with a weight equal to 1.

Finally, we have a query of the form:

$$\#weight(n_j\ \#1(U_j).title\ n_i\ \#1(U_i).a\ n_i\ \#1(U_i).s\ \#combine(text\ unigrams))$$

Where U_i: unigram title at i position, n_i: number of iterations of U_i, i from 1 to final unigram position and j from 1 to 2.

Improved queries allowed us to drastically reduce the number of Wikipedia documents returned, while increasing the relevance of these.

3 Summarizing and XML

The summarizing is the second part in our process of automatic answering. This part is composed by four majors steps. The first step "Filtering" will aim at removing every irrelevant element inside the corpus.

Then the second part will edit each word in the corpus in order to keep only the meaning of the word to make the summarizing more efficient. Step number three "Scoring" will give a score to each sentence depending on words contained in the sentence.

To finish a Decision algorithm give a rank to each sentence based on the score made earlier. The top-ranked sentence is kept in order to make the answer and to meet the five hundred word. We can decompose the path like this:

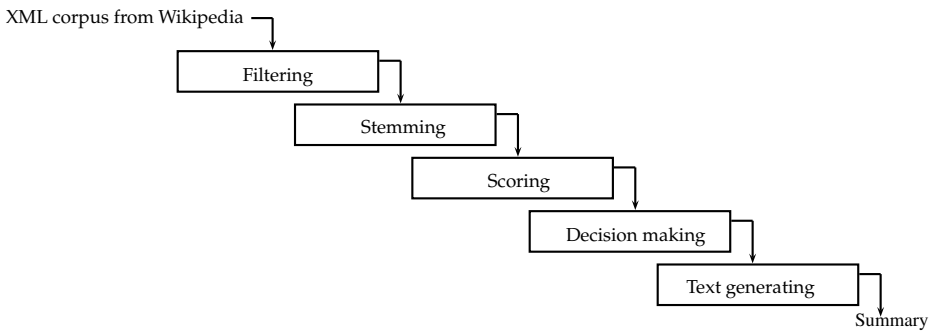


Fig. 2. Summarizing diagram

3.1 Filtering

The filtering step is the first one in the summarizing process we use. The aim in this step is to clean up the corpus. That means we are going to remove every element which can't be useful in the summarizing process. At first the corpus is segmented by sentence. Each line in the corpus was made to be a sentence in order to make editing texts easier. Our corpus contains XML, so we have to deal with tag during the whole process. To handle that tag are segmented as a sentence, as it's showed by the example below.

```
<tag>
  <tag2>
    sentence
  </tag2>
  sentence
  sentence
</tag>
```

In the corpus, we noticed that pages contain only a file and a file's title. We realized these pages never contain any information that can be used by the summarizer. So we decided to remove this kind of pages. It has appeared that there are paragraphs which only contain a picture or a drawing. Those elements are useless for a text summarizer, so we obviously chose to remove them. So, this is the first part of filtering: removing noise in the corpus that used to contain key word but never usable sentences.

The next step is Character filtering. The summarizer looks for hardly usable Character in the corpus and removes them. In this process numbers are removed because it's hard to detect if a number means a quantity, a date or something else. And the summarizer can't deal with numbers as they increase the noise. Punctuation marks are removed too. A meaning could be deduced from these marks but the model used here only uses words. During this process quotes, accents, brackets, etc, are all removded.

The corpus now only contain words and spaces. So, here comes the last step. A stop list is now used in order to keep only words which express something in the corpus. The stop list contains every word which appears too often and can't caracterize a sentence.

3.2 Stemming

The second step is the stemming. In this step, each word is reduced to his stem. We need to find a relation between sentences. The answer is to link words which have the same root and must be talking about the same thing. This step is done thanks to the Porterh[4] algorithm.

3.3 Scoring

When the stemming is finished, the third step begins. Each line is transformed in a vector in order to have a matrix which represents the entire corpus. Then three metrics are done on those vectors in order to give to each of them a score.

In the matrix made, each line represents a sentence in the corpus and each column represents a word from the corpus.

$$Corpus = \begin{matrix} TF_{1,1}\, TF_{1,2} \\ TF_{2,1}\, TF_{2,2} \end{matrix} \tag{2}$$

The 'TF' means Term Frequency defined by :

$$TF_{i,j} = \frac{n_i}{\sum_x n_{x,j}} \tag{3}$$

where S_j is the sentence number j,
and W_i is the word number i,
and $\sum_x n_{x,j}$ is the count of words in S_j,
and n_i is the number of W_i in the entire corpus.

When the matrix is done, but still, sentences can't be ranked, we need to make a score with sentences's vectors. In order to do this score, three metrics were used. The first metrics is the Sum frequency defined as below

$$Score_j = \sum_k TF_{k,j} \tag{4}$$

[4] http://tartarus.org/martin/PorterStemmer/

This metric give a good score to sentences which have a lot of important words but long sentences have better marks than shorter but they are not more important just because they are longer. We need to use another metric. This one is called entropy, defined like this :

$$E_j = -\sum_k p_{kj} \times \log p_{k,j} \tag{5}$$

We use two metrics XML based. The first metric is "t tag count". This metric can be explained by the more a sentence contains links the more relevant it is.

That can be defined as :

$$t_j = \sum_k q_k \tag{6}$$

where $q_k = 1$ if word number k surround by t tag

and $q_k = 0$ otherwise.

In XML, a document can be shown as a tree. We noticed that the deeper in the tree the text is, the less concise the text should be. In order to score, we give a mark to each sentence depending on where the sentence is in the XML tree. We use a scoring system completely dependent of the context defined as follows :

$$tree_j = Q_j^a + Q_j^h + Q_j^{title} + Q_j^s + Q_j^p \tag{7}$$

Where $Q^a = 20$ if sentence is between ¡a¿ 0 otherwise,

$Q^h = 10$ if sentences is between h tag 0 otherwise,

$Q^{title} = 10$ if sentences is between title tag 0 otherwise,

$Q^s = 5$ if sentences is between s tag 0 otherwise,

$Q^p = 1$ if sentences is between p tag 0 otherwise.

At this point we changed a vector of words against a vector of 4 metrics. it's not easier to make a rank of sentence. That's where the last step begins.

3.4 Decision making algorithm

The last step based on [5] is called Decision making because this step have to define which sentences are most relevant in order to use them in the summary. At first each metrics is standardized between 0 and 1 thanks to the equation :

$$std(score_{j,m}) = \frac{score_{jm} - \min(score_m)}{\max(score_m) - \min(score_m)} \tag{8}$$

Where j is a sentence

and m is a metric. In order to process next steps, each metric is normalized. We have to define an average able to handle those metrics and finally score each sentence. the Decision making algorithm is defined as follows:

$$Avg_{sup} = \sum_{v=1 score_{v,j} > 0.5}^{M} (score_{j,v} - 0.5) \tag{9}$$

$$Avg_{inf} = \sum_{v=1 score_{v,j}<0.5}^{M} (0.5 - score_{j,v}) \tag{10}$$

Where M is the number of metrics used
j is the sentence we are scoring
if $Avg_{sup} > Avg_{inf}$ then

$$FinalMark_j = 0.5 + \frac{Avg_{sup}}{M} \tag{11}$$

otherwise

$$FinalMark_j = 0.5 - \frac{Avg_{inf}}{M} \tag{12}$$

This algorithm return a Single final mark for each sentence. We are now able to sort them and use only the top sentences in order to make the summary. We will keep the maximum sentences we can to be as relevant as possible but we will never have more thant five hundred words in the summary. The summary is ready to be assessed with FRESA [2] [3] [4].

4 Conclusion

To conclude, we made a summarizer able to answer the question "what is this tweet about ?". This answer isn't perfect, the summarizer is far to be too. Using XML in queries improving and summarizing raises the question of the XML capacities in automatic Summarizing. We found a way to use it in the INEX QA track context but there is certainly a way to generalize this use. It could be smart to focus on the tree level as we did with scoring the deep. This brief overview is now ended, we hope to be able go further, maybe in the next INEX.

References

1. Shlomo Geva, Jaap Kamps, Ralf Schenkel, and Andrew Trotman, editors. *Comparative Evaluation of Focused Retrieval - 9th International Workshop of the Inititative for the Evaluation of XML Retrieval, INEX 2010, Vugh, The Netherlands, December 13-15, 2010, Revised Selected Papers*, volume 6932 of *Lecture Notes in Computer Science*. Springer, 2011.
2. Horacio Saggion, Juan-Manuel Torres-Moreno, Iria da Cunha, and Eric SanJuan. Multilingual summarization evaluation without human models. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters (COLING'10)*, pages 1059–1067, Beijing, Chine, 2010. ACL.
3. J.-M. Torres-Moreno, Horacio Saggion, I. da Cunha, P. Velazquez-Morales, and E. SanJuan. Evaluation automatique de résumés avec et sans réferences. In *Proceedings de la conference Traitement Automatique des Langagues Naturelles (TALN'10)*, Montral, QC, Canada, 2010. ATALA.

4. Juan-Manuel Torres-Moreno, Horacio Saggion, Iria da Cunha, and Eric SanJuan. Summary Evaluation With and Without References. *Polibits: Research journal on Computer science and computer engineering with applications*, 42:13–19, 2010.
5. Jorge Vivaldi, Iria da Cunha, Juan Manuel Torres-Moreno, and Patricia Velzquez-Morales. Automatic summarization using terminological and semantic resources. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA).

A graph-based summarization system at QA@INEX track 2011

Ana Lilia Laureano-Cruces[1,2] and Javier Ramírez-Rodríguez[1,2]

[1] Universidad Autónoma Metropolitana-Azcapotzalco
Mexico
[2] LIA, Université d'Avignon et des Pays de Vaucluse
France
{clc,jararo}@correo.azc.uam.mx
http://lia.univ-avignon.fr/

Abstract. In this paper we use REG, a graph-based system 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 2011 task (question-answering). We have extracted the title and some key or related words according to two people from the queries, in order to recover 50 documents from english wikipedia. Using this strategy, REG obtained good results with the automatic evaluation system FRESA.

Key words: Automatic Summarization System, Question-Answering System, graph-based.

1 Introduction

Nowadays automatic summarization using graph-based ranking algorithms has drawn much attention in recent years from the computer science community and has been widely used (Mihalcea, 2004; Sidiropoulos and Manolopoulos, 2006; Torres-Moreno and Ramírez Rodríguez, 2010; Torres-Moreno, 2011). According to (Saggion and Lapalme, 2002: 497) a summary is "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". Summaries has two main approaches "extraction", if they contain the most important sentences extracted from the original text (ex. Torres-Moreno et al., 2002; Torres-Moreno, 2011) and "abstraction", if these sentences are re-written or paraphrased from the source, generating a new text (ex. Ono et al., 1994; Paice, 1990; Radev, 1999). Most of the automatic summarization systems are extractive. These systems have been used in different domains (ex. da Cunha et al., 2007; Vivaldi et al., 2010; Farzindar et al., 2004; Abracos and Lopes, 1997). One of the tasks where these extractive summarization systems could help is question-answering. The objective of the INEX@QA 2011 track is contextualizing tweets, i.e. answering questions of the form "what is this tweet about?"

using a recent cleaned dump of the Wikipedia. In this task is where automatic summarization systems is used. Each of the selected 132 topics for 2011 includes the title and the first sentence of a New York Times paper that were twitted at least two months after the 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-Rodríguez, 2010; Torres-Moreno et al., 2010), an automatic summarization system based on graphs.

The rest of this paper is organized as follows. In the next section we show REG, the graph-based summarization system we have used for the experiments. In Section 3 we explain how we have carried out the terms extraction of the queries. In Section 4 we present the results. In Section 5, some conclusions are given.

2 The graph-based system

REG (Torres-Moreno and Ramírez-Rodríguez, 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 of traversal graph to obtain a desired number of relevant sentences, all if necessary. 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, a desired number of sentences with more score will be selected by the greedy algorithm as the most relevant. Finally, the third module generates the summary, selecting and concatenating the desired number of relevant sentences. The first and second modules use CORTEX (Torres-Moreno et al., 2002), a system that carries out an unsupervised extraction of the relevant sentences of a document using several numerical measures and a decision algorithm.

3 Treatment of queries

The 132 queries obtained from the topics of the New York Times were processed by two persons. The first one has chosen in addition to the title the words of the phrase are, from her point of view, key words. The second has been considered in

addition to the title, words related to the to the title, for example, for the query "Largest Holder of US Debt" we consider the terms: debit, china and USA. The 132 queries were processed by the perl program of inex and sent to INDRI to subsequently obtain 50 documents per query from english wikipedia, of which REG obtained a summary of between 500 and 600 words of each. A random sample was selected from 6 queries of each form of generating it and that were evaluated with the FRESA tool.

4 Results

In this study, we used the document sets made available during the —Initiative for the Evaluation of XML retrieval (INEX) 2011 [1], in particular on the INEX 2011 QA Track (QA@INEX). These sets of documents where provided by the search engine Indri. [2] REG has produced the same number of summaries of approximately 550 words each.

To evaluate the efficiency of REG over the INEX@QA corpus, we have used the FRESA package.

Table 1 shows an example of the best result obtained by REG using 50 documents as input. The query that the summary should answer in this case was the number 2011024. 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 baselines. Then we present the average of the random sample for each one of the people and the Baseline summary, finding that REG is acceptable. To complete the study, we are going to choose a larger random sample of the queries to compare the averages of reg with those of the other baselines.

Table 1. Example of REG results using 50 documents as input.

Distribution type	unigram	bigram	with 2-gap	Average
Baseline summary	20.5418	27.5864	27.6669	25.2650
Empty baseline	29.3267	36.8774	36.9097	34.3712
Random unigram	19.6240	27.2631	27.2464	24.7112
Random 5-gram	19.0643	26.1404	26.3386	23.8478
Submitted summary	20.2178	27.2809	27.3687	24.9558

[1] `https://inex.mmci.uni-saarland.de/`

[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 2. Averge of REG results for each person to generate the query.

Form	Person 1	Person 2	Baseline summary
Average	33.6984	33.9505	32.8402

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 2011 task, extracting the terms from the queries, in order to obtain a list of terms related with the main topic of the question.

Our experiments have shown that the system is always better than the two simple baselines, but in comparison with the first one the performance is variable. We think this is due to the fact that some queries are long and they have several terms we could extract, but there are some queries that are very short and the term extraction is not possible or very limited. Nevertheless, we consider that, over the INEX-2011 corpus, REG obtained good results in the automatic evaluations, but now it is necessary to wait for the human evaluation and the evaluation of other systems to compare with.

References

1. Abracos, J.; Lopes, G. (1997). *Statistical methods for retrieving most significant paragraphs in newspaper articles*. In Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization. Madrid. 51-57.
2. da Cunha, I.; Wanner, L.; Cabré, M.T. (2007). *Summarization of specialized discourse: The case of medical articles in Spanish*. Terminology 13 (2). 249-286.
3. Farzindar, A.; Lapalme, G.; Desclés, J.-P. (2004). *Résumé de textes juridiques par identification de leur structure thématique*. Traitement automatique des langues 45 (1). 39-64.
4. Mihalcea, R. (2004). *Graph-based ranking algorithms for sentence extraction, applied to text summarization*. In Proceedings of the 42nd Annual Meeting of the Association for Computational Lingusitics (ACL 2004) (companion volume), Barcelona, Spain.
5. Ono, K.; Sumita, K.; Miike, S. (1994). *Abstract generation based on rhetorical structure extraction*. In Proceedings of the International Conference on Computational Linguistics. Kyoto. 344-348.
6. Paice, C. D. (1990). *Constructing literature abstracts by computer: Techniques and prospects*. Information Processing and Management 26. 171-186.
7. Radev, D. (1999). Language Reuse and Regeneration: Generating Natural Language Summaries from Multiple On-Line Sources. New York, Columbia University. [PhD Thesis]
8. Saggion, H.; Lapalme, G. (2002). *Generating Indicative-Informative Summaries with SumUM*. Computational Linguistics 28 (4). 497-526.

9. Torres-Moreno, J-M.(2011). *Résumé automatique de documents: Une approche statistique* . Hermès-Lavoisier (Paris).
10. Torres-Moreno, J-M.; Saggion, H. da Cunha, I. SanJuan, E. Velázquez-Morales, P. SanJuan, E.(2010a). *Summary Evaluation With and Without References*. Polibitis: Research Journal on Computer science and computer engineering with applications 42.
11. Torres-Moreno, J.-M.; Saggion, H.; da Cunha, I.; Velázquez-Morales, P.; SanJuan, E. (2010b). *Ealuation automatique de résumés avec et sans référence*. In Proceedings of the 17e Conférence sur le Traitement Automatique des Langues Naturelles (TALN). Université de Montréal et Ecole Polytechnique de Montréal: Montreal (Canada).
12. Torres-Moreno, J-M.; Ramírez, J. (2010). *REG : un algorithme glouton appliqué au résumé automatique de texte*. JADT 2010. Roma, Italia.
13. Torres-Moreno, J-M.; Ramírez, J.; da Cunha, I. (2010). *Un resumeur a base de graphes, indépendant de la langue*. Workshop African HLT 2010. Djibouti.
14. Torres-Moreno, J. M.; Velázquez-Morales, P.; Meunier, J. G. (2002). *Condensés de textes par des méthodes numériques*. En Proceedings of the 6th International Conference on the Statistical Analysis of Textual Data (JADT). St. Malo. 723-734.
15. Vivaldi, J.; da Cunha, I.; Torres-Moreno, J.M.; Velázquez, P. (2010). "Automatic Summarization Using Terminological and Semantic Resources". In proceedings of 7th International Conference on Language Resources and Evaluation (LREC 2010). Valletta, Malta.
16. Sidiropoulos; A.; Manolopoulos, Y. (2006). *Generalized comparison of graph-based ranking algorithms for publications and authors*. In: The Journal of Systems and Software. Volume 79. 1679-1700.

SUMMA Content Extraction for INEX 2011

Horacio Saggion

TALN
Department of Information and Communication Technologies
Universitat Pompeu Fabra
C/Tanger 122/140 Poble Nou - Barcelona (08018) - Spain
horacio.saggion@upf.edu

Abstract. We present a text summarization system prepared for the INEX QA 2011 evaluation. The system constructs an answer summary with sentences extracted from Wikipedia documents deemed relevant to a given natural language topic. The system is based on the SUMMA summarization tool for sentence relevance computation and selection.

1 Introduction

The INEX 2011 Question Answering Track asks participants to create a short answer for the question "What is *this tweet* about?". An instance of the problem could be the following questions:

– What is *"At Comic-Con, a Testing Ground for Toymakers"* tweet about?

Given such a question, relevant documents from Wikipedia are retrieved in order to give context to the tweet. These relevant documents have to be summarized in order to reduce redundancy and provide the best possible answer.

We have developed a simple text summarization system in order to extract relevant sentences from Wikipedia documents. The system is based on available summarization technology provided through the SUMMA software [2] which has been used in other evaluation programs [5, 1, 4, 3].

This short communication provides an overview of the technology used to develop our summarizer for INEX.

2 SUMMA tool

SUMMA is a toolkit for the development of text summarization systems [2]: given a summarization task a set of components can be pipelined to create and deploy a summarization application. SUMMA is composed of a series of modules to assess the relevance of sentences or larger units of text (e.g., paragraphs, documents) in a given context. Additional components provide infrastructure for document analysis, document representation, and evaluation. Here we describe SUMMA basic functionalities:

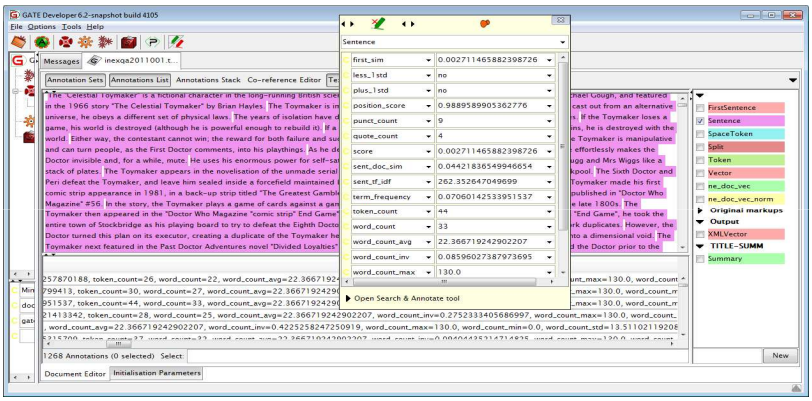


Fig. 1. Document Analised with SUMMA

- Statistics computation: SUMMA computes basic frequency statistics for words or other linguistic units such as named entities: these statistics can be normalized relying on inverted document frequency tables computed by specific modules.
- Text representation: The full document and various document units can be represented as vectors of terms and weights (e.g., the statistics). The system is flexible in that the statistics to use for representation are system parameters.
- Text similarity computation: Various similarity measures are provided with the tool such as cosine similarity or normalized word overlap.
- Sentence features: The basic mechanism to assess sentence relevance is feature computation and a number of programs are available that implement well known summarization relevance features such as sentence position, term frequency distribution, sentence title similarity, sentence query similarity, sentence centroid similarity, cue word distribution, etc.
- Feature combination: The computed features can be combined in an easy way (using a user specified weighting schema) to produce a numerical score for each sentence. The features could also be selected based on an statistical feature selection framework. The score can be used to rank sentences.

In Figure 1 we show a document analysed with SUMMA. The figure shows various features computed for a specific sentence such as the title feature, the word feature, the position feature, the centroid feature, etc.

3 Summarization for INEX 2011

The INEX QA task requires the creation of a 500-word summary of documents retrieved from Wikipedia given a query. Figure 2 shows the text retrieved for the tweet "At Comic-Con, a Testing Ground for Toymakers" by the programs provided by the INEX organizers which produces a combined file with all retrieved documents. Note that the first sentence of the retrieved text is the text of the tweet. We use directly this input in our summarizer.

At Comic-Con, a Testing Ground for Toymakers.
The "Celestial Toymaker" is a fictional character in the long-running British science fiction television series, "Doctor Who". He was played by Michael Gough, and featured in the 1966 story "The Celestial Toymaker" by Brian Hayles. ... He also wrote "Short History of Europe", edited "Lady Mary Wortley-Montagu's Letters" (Selection and Life), and was a contributor to "Punch" magazine. Ross died of heart failure at his home in Kensington, London on 11 September 1933 at the age of 73.

Fig. 2. Retrieved Wikipedia Documents

The input text is analysed with a standard tokenization, sentence splitting program provided by the GATE software. After this, word sentence statistics are computed over the document: the frequency of a word in the sentence and the word inverted sentence frequency (i.e., the number of sentences containing the word). The relevance of a word in the sentence is computed as:

$$w(t) = freq_{sent}(t) * \ln(\frac{N_{sent}}{N_{sent}(t)})$$

where $freq_{sent}(t)$ is the frequency of t in the sentence, N_{sent} is the number of sentences in the document, and $N_{sent}(t)$ is the number of sentences containing t. Each sentence is the document is represented in the vector space model using the words and their weights $w(t)$. The document is also represented as a vector but using global word frequency statistics.

For each sentence four features were computed: (i) the similarity between each sentence and the first sentence (i.e., the tweet) computed as the cosine of the two vector representations, (ii) the position of the sentence (note that documents at the beginning are more relevant than those at the end), (iii) the similarity between the the sentence and the document also computed as the cosine between the two vector representations, and (iv) the term frequency feature computed as a normalized sum of the weights of the words in the sentence.

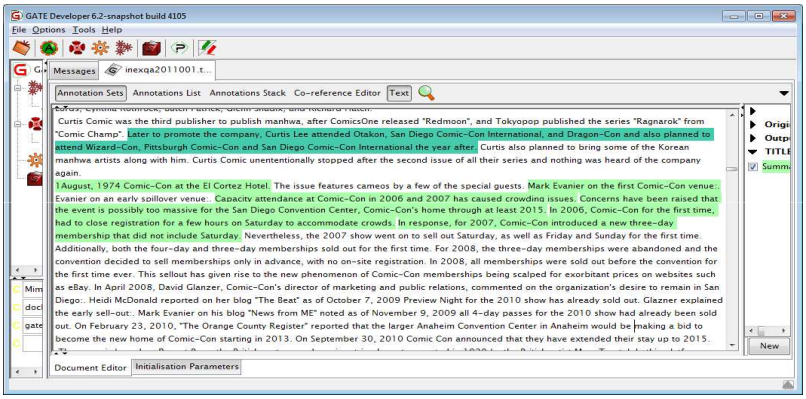


Fig. 3. Query-based Summary

3.1 Selecting the Summary

In order to decide what features to use to produce our answer summary, we relied on the Fresa software [7] that compares an input document with a summary through the computation of divergences. The tool has been proved to produce rankings correlated with rankings produced relying on comparison with human summary models [6]. We used a sample of documents to produce 4 different summaries each using a different feature: similarity to the first sentence, position, similarity to the document, and term frequency. In all cases the best summary according to the Fresa package was one produced using the similarity to the first sentence as scoring feature. We have also tried some feature combinations but none improve the results on the sample. We therefore decide to use this feature alone for scoring and selecting sentences. One example first sentence similarity summary produced by our tool is shown in Figure 3.

4 Outlook

In this short communication we have described the tool we are using for producing a summary that contextualizes a tweet for the INEX 2011 Question Answering task. We have developed a very simple system that scores sentences represented in a vector space model. The scores are produced by comparing the sentence to the initial tweet. This way of scoring sentences was decided after verifying that the "divergence" between summaries and documents was minimized

using this sentence scoring schema. We believe, however, the summarization system is still to simplistic and that it should be improved by taking into account more sophisticated features. We expect to improve the system based on the feedback received after evaluation.

References

1. H. Saggion. Experiments on semantic-based clustering for cross-document coreference. In *Proceedings of the Third Joint International Conference on Natural Language Processing*, pages 149–156, Hyderabad, India, January 7-12 2008. AFNLP, AFNLP.
2. H. Saggion. SUMMA: A Robust and Adaptable Summarization Tool. *Traitement Automatique des Langues*, 49(2):103–125, 2008.
3. Horacio Saggion. Matching texts with summa. In *Défi Fouille de Texte (DEFT 2011)*, Montpellier, France, June 2011 2011. TALN, TALN.
4. Horacio Saggion. Using summa for language independent summarization at tac 2011. In *Text Analysis Conference*, Gaithersburg, Maryland USA, 11/2011 2011. NIST, NIST.
5. Horacio Saggion and Robert Gaizauskas. Multi-document summarization by cluster/profile relevance and redundancy removal. In *Proceedings of the Document Understanding Conference 2004*, Boston, USA, May 6-7 2004. NIST.
6. Horacio Saggion, Juan-Manuel Torres-Moreno, Iria da Cunha, Eric SanJuan, and Patricia Velazquez-Morales. Multilingual summarization evaluation without human models. In *In Proceedings of COLING*, 2010.
7. J.M. Torres-Moreno, Horacio Saggion, I Da Cunha, P. Velazquez-Morales, E. SanJuan, and A. Molina. Summary evaluation with and without references. *Polibits*, In Press, 2010.

Combining relevance and readability for INEX 2011 Question-Answering track

Jade Tavernier and Patrice Bellot

LSIS - Aix-Marseille University
patrice.bellot@lsis.org, jade.tavernier@gmail.com

Abstract. For INEX 2011 QA track, we wanted to measure the impact of some classical measures of readability in the selection of sentences related to topics. We considered the readability as an indicator of the cohesion of the summaries we produced from documents. This is a step towards adaptive information retrieval approaches that take into account the reading skills of users and their level of expertise.

1 Introduction

While many scientific studies and papers deal with information retrieval in context, adapting retrieval to users with limited reading skills is still an open problem. This may include people with language disorders (eg dyslexia makes reading slow and complex) as well as those not proficient enough in the language of a document or that have to read a content whose necessary expertise for understanding is too high. Adaptation of information retrieval with the consideration of individual reading / understanding performance may be a major concern for modern societies where citizens learn online, without human mediation. The issue of measuring the readability of a text is important in many other areas. For example, it allows to estimate the level of difficulty of a text when a child learns reading or learns a foreign language.

Many studies have examined the concept of relevance in trying to define it according to extra-linguistic and contextual parameters that are not explicit in a query [15]. We do not specify in a query our level of expertise (how would we?) and we do not indicate our average speed of reading or difficulties. Moreover, the common retrieval models allow ordering documents according to how much information they convey vis-a-vis what the user expressed in its query, taking into account, possibly, a personal profile (previous queries and consultations) or that of other users who made a similar request. This is an essentially *informational* relevance. This is adequate to a certain extent but does not take into account that the needs are different depending, for example, the level of expertise of the user: someone new to an area will be more interested in a simple document rather than an expert study with very specialized terms and complex structure. Meet such a need for customization requires to define new measures taking into account readability and relevance of a text. In such a case, an information retrieval system could provide users with, for example, simplest documents first.

For INEX 2011 QA track, we wanted to measure the impact of some classical measures of readability in the selection of sentences answering to topics. We considered the readability as an indicator of cohesion of the summaries we extracted from documents. Related work concerns the estimation of linguistic quality, or the fluency, of a text such as employed in some automatic summarization systems [3, 16]. However, we are more interested in detection of text easier to understand than text well written. There are many challenges: determining a good measure of readability for the task, achieving a good balance between a selection of phrases according to their informational relevance (quantity of new information) and their readability.

In Section 2, we present some related work and classical measures of readability. Then, we present and machine learning approaches for adapting readability to users although we have not been able to experience them yet. In Section 3, we describe the experiments we realized and the runs we submitted. In Section 4, we analyze our runs and scores statistically.

2 Relevance and readability

The purpose of an information retrieval system is to provide relevant documents to the user in relation to its information need (topic or *query*). The notion of relevance has been the subject of many studies that attempt to identify the concepts to which it refers [18]. S. Mizzaro proposes a definition of relevance that encompasses several dimensions [15] :

- the need for information, broken down into real need, perceived need, expressed need, and a query written in a formal query language;
- the compounds : the information, the task and its context;
- the time required to retrieve the information;
- the granularity of the answers: comprehensive document, segments of documents or precise answer.

The type of documents, the level of detail, the date of writing are so many criteria that can be included in retrieval models, along with the popularity (PageRank) and the authority. The integration of the criterion of readability also requires to reformulate what is a relevant document. Adapting the retrieval process to the reading skills of the user amounts to considering the readability as a continuous variable that we seek to maximize while selecting only the best documents verifying the other criteria of relevance. Readability can be seen as an additional criterion in the retrieval model or as a new filtering step in the selection of documents.

2.1 Classical measures for estimating readability

W.H. Dubay notes that more than 200 readability formulas can be found in the literature [8]. They exploit linguistic, syntactic, semantic or pragmatic clues. In

this section, we report the most classic formula. Coh-Metrix[1] combines more than 50 measures to calculate the coherence of texts [11]. Some measures of readability still used date back fifty years. For the English language, the most common measures of readability are : FOG [12] (formula 1), SMOG [14] (formula 2) and Flesch [9] (formula 3) which is proposed by software such as Microsoft Word in its statistical tools set. Derivatives of these formulas can also be found such as formula 4 described in [20].

Let d a document (or a text or a sentence), $L(d)$ the readability of d. Let ASL be the average length of sentences (number of words), ASW the average number of syllables per word, MON the frequency of monosyllabic words in d and POL the number of multisyllabic words in the beginning, in the middle and at the end of d.

$$L_{FOG}(d) = 3,068 + 0,877 \times ASL + 0,984 \times MON \tag{1}$$

$$L_{SMOG}(d) = 3 + \sqrt{POL} \tag{2}$$

$$L_{Flesch}(d) = 206,835 - 1,015 \times ASL - 84,6 \times ASW \tag{3}$$

$$L_{Flesch-Kincaid}(d) = 0,39 \times ASL + 11,8 \times ASW - 15,59 \tag{4}$$

Some other measures are reported in [5] that employ closed list of words classified according to their difficulty such as the Caroll-Davies-Richman list [2] and the Dale & O'Rourke dictionary [7] : Lexile [21], Dale-Chall (formula 5) [4] and Fry [10] for very short texts.

$$L_{Dale-Chall}(d) = (0.1579 \times DS) + (0.0496 \times ASL) + 3.6365; \tag{5}$$

where DS is Dale-Chall Lexicon-based score, or % of words not in their dictionary of 3,000 common words, and ASW the average number of syllables per word.

Note that many other factors, particularly sensitive for visually impaired people, could be considered: spacing and font size, contrast, paragraph alignment, the presence of images, physical structure of documents... These criteria are taken into account to measure the accessibility of Web pages by *Web Accessibility Initiative* (WAI).

2.2 Machine Learning for readability measures adapted to users

The availability of large corpora and the organization of large-scale experiments have enabled the development of new measures of readability that does not involve numerical parameters selected manually. They are used to better reflect certain contexts than do generic measures such as those defined above. L. Si

[1] http://cohmetrix.memphis.edu/cohmetrixpr/index.html

and J. Callan have proposed to combine linguistic criteria and surface analysis (unigram language model) [20].

They conclude that the rarity of words (estimated through a generic unigram model) and the length of sentences are meaningful criteria for estimating the difficulty of reading a web page but not the number of syllables per word. This is consistent with psycholinguistic studies on the reading process: a long word is not necessarily more difficult to recognize if it has already been met by the reader.

As recommended by W3C, K. Inui and S. Yamamoto looked at the issue of simplification of text for easy reading. Solving this problem requires the identification of the most complex sentences in the texts [13]. The authors note that the usual measures of readability do not take into account the literacy skills of persons to whom the texts were intended. They propose to learn automatically from manually labeled examples a ranking function of sentences. In their first experiments only POS-tags are taken into account. R. Barzilay and M. Lapat propose a similar solution by learning a ranking function estimating the local consistency of the texts by means of an analysis of transitions between sentences [1]. They show that their method can be effectively applied to the detection of complex sentences in that it connects cohesion and complexity to the number of common words and entities between sentences to the number of anaphoric relations and to some syntactic information.

More recently, Tanaka-Ishii have addressed the problem of building a training corpus clustered in different classes of difficulty. They propose to see it as a classification task by only tagging pairs of texts for which is easily possible to determine, for each pair, which of both texts is more complex than the other [22]. K. Collins-Thompson and J. Callan have developed an approach to automatically determine the grade level associated with a web page using a multinomial naive Bayesian approach which assumes that the different levels of difficulty are not independent (grade 1 is closest to grade 2 than to grade 3). They lead to a better prediction quality than that obtained by means of a measure such Flesch [5]. S. Schwarm and M. Ostendorf have extended this method to n-gram models that, to some extent, capture the syntactic complexity of sentences [19]. They combine, using a SVM classifier, the scores obtained by the Flesch-Kincaid formula, the number of new words and some information such as the average number of noun phrases and verbal phrases by sentence. An evaluation of their model to discriminate documents from encyclopedia and journals, for children and adults, showed results significantly higher than those achieved with the Flesch or Lexile measures alone. E. Pitler and A. Nenkova brought together previous approaches by including an analysis of discourse relations between sentences of a text and of lexical cohesion, unigram models and numerical values such length of text, number of pronouns, definite articles, noun phrases... [17].

3 Experiments

3.1 Informational Based Summarization

For QA track, we started by querying The New York Times in the Termwatch platform[2]. Each queries returned a set of Wikipedia documents. Then, we had to summarize every retrieved document.

Relevance scores for initial selection of sentences. At first, for each set of documents, we made a preprocessing : sentence splitting, filtering (punctuation, parentheses...), stemming.

Then, we calculated three relevance scores and we employed a decision formula to attribute a final score to each sentence.

1. Entropy :

$$E(s) = \sum tf. \log tf \tag{6}$$

where tf is term-frequency:

$$tf_{i,s} = \frac{n_{i,s}}{\sum_k n_{k,s}} \tag{7}$$

with i a word, s a sentence, $n_{i,j}$ the number of occurrences of i in j.
2. Similarity (dot product) between the sentence s and the title of the document it belongs to,
3. Sum Frequency F:

$$F(s) = \frac{\sum tf}{N} \tag{8}$$

with N the total number of words in the sentence s.

Initial scoring of the sentences. In order to obtain a score for each sentence s, we followed the following steps:

1. Calculate two average values $\sum_\alpha^\mu$ and $\sum_\beta^\mu$

$$\alpha(s) = \sum_{v=0/\lambda_{s,v}>0.5}^{\gamma-1} (\lambda_{s,v} - 0.5) \tag{9}$$

$$\beta(s) = \sum_{v=0/\lambda_v<0.5}^{\gamma-1} (0.5 - \lambda_{s,v}) \tag{10}$$

where $\lambda_{s,v}$ is the score of a sentence s by employing a scoring function v (one the three scores defined above) and γ the number of scoring functions (in our case, $\gamma = 3$).

[2] http://http://qa.termwatch.es/

2. Make decision : let $SP(s)$ be the final score of sentence s:

$$SP(s) = \begin{cases} 0.5 + \frac{\alpha(s)}{\gamma} \text{ if } \alpha(s) > \beta(s) \\ 0.5 - \frac{\beta(s)}{\gamma} \text{ else} \end{cases} \quad (11)$$

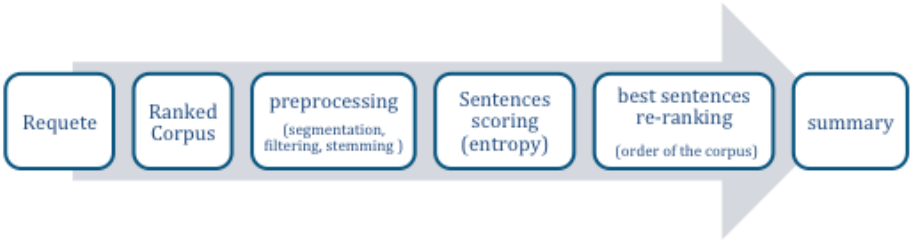


Fig. 1. Steps producing the initial set of ranked sentences.

3.2 Combining relevance score and readability

In the following, we explain how we incorporated readability in the summarization process. We chose to combine linearly the inital score with a readability score. Then we chose to produce a summary by ranking the top scored sentences by following the order they appear in the set of documents retrieved by TermWatch.

The final score of sentence s combining original score SP and readability SL is:

$$S(s) = \lambda.SP(s) + (1 - \lambda).SL(s) \quad (12)$$

with $\forall s$, $0 \leq SP \leq 1$ and $0 \leq SL(s) \leq 1$. According to our previous experiments, we chose to set $\lambda = 0.7$.

3.3 Runs submitted

We submitted 3 runs. Each run combines readability and original scoring function. Figure 2 show the beginning of a summary ignoring readability. It corresponds to Run 1 of University of Avignon - LIA team. This will allow to compare our results with a baseline.

Run 1. For this first run we decided to use the Flesch readability formula (3)[3].

Figure 3 shows the summary produced by using Flesch on the summary presented in Figure 2.

[3] We use the perl library Lingua::EN::Syllable to calculate the number of syllables per sentence.

Humans first landed on the Moon on July 20, 1969.
During the same period, Lincoln Towing issued 139
This Sentry is the source of the infection in "Marvel Zombies".
Trogden's father said, "I call myself Heat Moon, your elder brother is Little Heat Moon.
Elusive Heat Elusive Heat (foaled April 30, 2006) is a dark bay mare by Elusive Quality.
Private equity firms, with their investors,
Because private equity firms are continuously in the process of raising,
Rydquist became interested in drifting upon moving to California.
Flack has been inducted into the Australian Sports and Athletics Hall of Fame
The following is an outdated ranking of the largest private equity firms.
These same principals guide Goodman today.
A stub for private equity and venture capital firms.

Fig. 2. Example of summary without taking into account readability.

Humans first landed on the Moon on July 20, 1969.
Trogden's father said, "I call myself Heat Moon, your elder brother is Little Heat Moon.
Elusive Heat Elusive Heat (foaled April 30, 2006) is a dark bay mare by Elusive Quality.
Other Foals Xtra Heat has three additional foals from 2008, 2009, and 2010 who have not yet raced
Because private equity firms are continuously in the process of raising,
Rydquist has been crew on a Swedish sail racing team.
The following is an outdated ranking of the largest private equity firms.
A stub for private equity and venture capital firms.

Fig. 3. A summary of the document presented in Figure 2 by taking into account Flesch readability. We have highlighted the sentences that did not appear in the initial summary.

Run2. For this run, we used the Dale Chall formula (5). This formula does not take into account the number of syllables per sentence, but checks if the words are in the list of Dall-Chall (list of 3000 words)[6]. Figure 4 shows the summary produced by using Flesch on the summary presented in Figure 2.

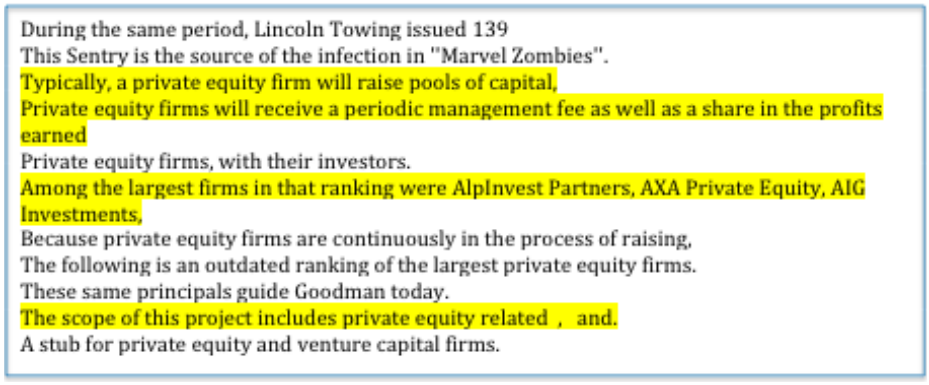

During the same period, Lincoln Towing issued 139
This Sentry is the source of the infection in "Marvel Zombies".
Typically, a private equity firm will raise pools of capital,
Private equity firms will receive a periodic management fee as well as a share in the profits earned
Private equity firms, with their investors.
Among the largest firms in that ranking were AlpInvest Partners, AXA Private Equity, AIG Investments,
Because private equity firms are continuously in the process of raising,
The following is an outdated ranking of the largest private equity firms.
These same principals guide Goodman today.
The scope of this project includes private equity related , and.
A stub for private equity and venture capital firms.

Fig. 4. A summary of the document presented in Figure 2 by taking into account Dale-Chall readability measure. We have highlighted the sentences that did not appear in the initial summary.

Run 3. For the third run we have decided to not score each sentence independently to others. We computed a score by using a sliding window of three sentences and by employing Flesch formula (3). This allowed us to take into account the context of each sentence.

Let $SP'(s_n)$ be the new "informational relevance" score of a sentence s in position n in a document :

$$SP'(s_n) = \max\{SP(s_{n-2,..,n}), SP(s_{n-1,...,n+1}), SP(s_{n,...,n+2})\} \quad (13)$$

with $SP(s_{n-1,...,n+1})$ the score (formula 11) of the aggregation of the sentences in position $n-1$, n and $n+1$.

We did the same for computing a new readability score SL':

$$SL'(s_n) = \max\{SL(s_{n-2,..,n}), SL(s_{n-1,...,n+1}), SL(s_{n,...,n+2})\} \quad (14)$$

As previously, we combined SP' and SL' linearly to obtain a new final score S' :

$$S'(s) = \lambda.SP'(s) + (1-\lambda).SL'(s) \quad (15)$$

Figure 5 shows the summary produced by using these new scoring functions on the summary presented in Figure 2.

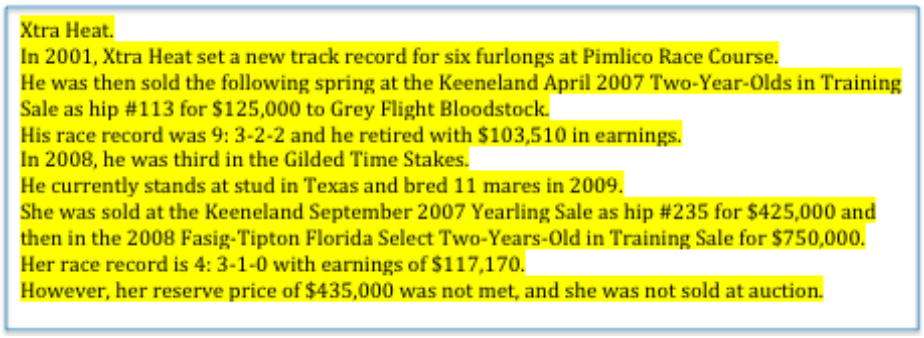


Fig. 5. A summary of the document presented in Figure 2 by taking into account our Window-based measures. We have highlighted the sentences that did not appear in the initial summary.

4 Results

At the time of writing this paper, we have not the results of the task. We studied statistically the association between the different scoring functions we have tested. More precisely, we computed the Kendall rank correlation coefficient for each topic between pairs of ranking of the top 50 best scored sentences.

Table 1 shows the proportions of queries for which the assumption of independence between the rankings should be rejected (p-value ¡ 0.1) i.e. on which there is not significant difference in the ranking of top 50 sentences. It indicates these proportions:

- between the original relevance score and each of the three readability formulas we used,
- between final rankings (combinations S of relevance score SP and readability SL with different values of λ) and each of the three readability formulas.

	Readability measure								
	Flesch (3)			Dale-Chale (5)			Window-based (15)		
SP only	0.121			0.167			0.136		
Final score S	$\lambda = 0.6$	$\lambda = 0.7$	$\lambda = 0.8$	$\lambda = 0.6$	$\lambda = 0.7$	$\lambda = 0.8$	$\lambda = 0.6$	$\lambda = 0.7$	$\lambda = 0.8$
	0.659	0.811	0.947	0.477	0.598	0.758	0.893	0.992	1

Table 1. Proportions of queries for which the assumption of independence between the rankings (SP only or S and Flesch or Dale-Chale or Window-based) should be rejected for different values of λ (based on Kendall's tau coefficient).

As expected, the association between readability scores and the original score SP is very weak since only about 12-16% of the topics lead to dependent rank-

ings. The Dale-Chale measure is the most correlated with the initial scoring SP, but it is the one that changes the final ranking the most.

5 Conclusion and Future work

Our contributions for the INEX 2011 question-answering track have involved the combination of informational relevance scores and readability scores during extraction of sentences from documents. We submitted three runs. The first two by using two classical measures and the third one by proposing a smoothed window-based readability function. In future work, we plan to improve our approaches by refining readability according to experiments with real users searching for documents in a digital library. Our aim is to provide them with navigation facilities : contextual routes through the collection of documents and user-adapted recommendation.

References

1. R. Barzilay and M. Lapata. Modeling local coherence: An entity-based approach. In 43rd Annual Meeting of the ACL, pages 141–148. Association for Computational Linguistics, 2005. Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics.
2. D. Carroll, P. Davies, and B. Richman. Word frequency book. American Heritage, New York. 1971.
3. J. Chae. Predicting the fluency of text with shallow structural features: case studies of machine translation and human-written text. In 12th Conference of the European Chapter of the ACL (EACL 2009), pages 139–147, 2009.
4. J. S. Chall and E. Dale. Readability revisited: The new Dale-Chall readability formula. Brookline Books Cambridge, MA, 1995.
5. K. Collins-Thompson and J. Callan. A language modeling approach to predicting reading difficulty. In HLT-NAACL, volume 4, 2004. Proceedings of HLT/NAACL.
6. Dale and Chall. A formula for predicting readability. 1948.
7. E. Dale and J. O'Rourke. The living word vocabulary. Chicago: World Book-Childcraft International. 1981.
8. W. H. DuBay. The principles of readability. Impact Information, pages 1–76, 2004.
9. R. Flesch. A new readability yardstick. Journal of applied psychology, 32:221–233, 1948.
10. E. Fry. A readability formula for short passages. Journal of Reading, 33(8):594–597, 1990.
11. A. C. Graesser, D. S. McNamara, M. M. Louwerse, and Z. Cai. Coh-metrix: Analysis of text on cohesion and language. Behavior Research Methods, Instruments, and Computers, 36:193–202, 2004.
12. R. Gunning. The Teaching of Clear Writing. New York: McGraw Hill, 1952.
13. Kentaro Inui, Satomi Yamamoto, and Hiroko Inui. Corpus-based acquisition of sentence readability ranking models for deaf people. In Sixth Natural Language Processing Pacific Rim Symposium (NLPRS 2001), pages 159–166, Tokyo, Japan, 2001.
14. G. H. McLaughlin. Smog grading: A new readability formula. Journal of reading, 12(8):639–646, 1969.

15. S. Mizzaro. Relevance : the whole history. Journal of the American Society for Information Science, 48(9):810–832, 1997.
16. E. Pitler, A. Louis, and A. Nenkova. Automatic evaluation of linguistic quality in multi-document summarization. In 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010), pages 544–554, 2010.
17. E. Pitler and A. Nenkova. Revisiting readability: A unified framework for predicting text quality. In EMNLP 2008, pages 186–195, 2008. Proceedings of the Conference on Empirical Methods in Natural Language Processing.
18. T. Saracevic. Relevance: A review of and a framework for the thinking on the notion in information science. Journal of the American Society for Information Science, 26(6):321–343, 1975.
19. S. E. Schwarm and M. Ostendorf. Reading level assessment using support vector machines and statistical language models. In 43rd Annual Meeting of the ACL, pages 523–530. Association for Computational Linguistics, 2005. Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics.
20. L. Si and J. Callan. A statistical model for scientific readability. In Proceedings of the tenth international conference on Information and knowledge management, pages 574–576. ACM, 2001.
21. A. J. Stenner, I. Horablin, D. R. Smith, and M. Smith. The lexile framework, MetaMetrics. Inc., Durham, NC. 1988.
22. K. Tanaka-Ishii, S. Tezuka, and H. Terada. Sorting texts by readability. Computational Linguistics, 36(2):203–227, 2010.

The Cortex and Enertex summarization systems at the QA@INEX track 2011

Juan-Manuel Torres-Moreno[1] and Patricia Velázquez-Morales[2]
and Michel Gagnon[1]

[1]École Polytechnique de Montréal - Département de génie informatique
CP 6079 Succ. Centre Ville H3C 3A7 Montréal (Québec), Canada.
[2] VM Labs, 84000 Avignon, France.
juan-manuel.torres@univ-avignon.fr,patricia_velazquez@yahoo.com,michel.gagnon@polymtl.ca
http://www.polymtl.ca

Abstract. The Enertex system is based on a statistical physics approach. It transforms a document on a spins (words present/absent) system. A summary is obtained using the sentences with large spins interactions. In another way, 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 2011 task of Questions, Cortex strategy system obtained very good results in the automatic evaluations FRESA.

Key words: INEX, Automatic summarization system, Question-Answering system, Cortex, Enertex.

1 Introduction

Automatic text 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, 2]. We adopted a simpler method, usually called *extraction*, that allow to generate summaries by extraction of pertinence sentences [3–5, 2]. 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 [6, 7] has been used in information extraction, information retrieval, question-answering, and it may also be used in text summarization. CORTEX[1] is an automatic summarization system, recently developed [8] 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

[1] *CORTEX es Otro Resumidor de TEXtos* (CORTEX is anotheR TEXt summarizer).

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 [9]:

- 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 [10–12]. 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 [13–15]. In the systems described in [13], 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 [14], 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 [15] 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 INEX Task Question Answering 2011. In Section 3 we explain the methodology of our work. Experimental settings and results obteined with Enertex and Cortex summarizers are presented in Section 4. Section 5 exposes the conclusions of the paper and the future work.

2 The INEX initiative and the QA INEX Track

The Initiative for the Evaluation of XML Retrieval (INEX) is an established evaluation forum for XML information retrieval (IR) [16]. This in initiative "...aims to provide an infrastructure, in the form of a large structured test collection and appropriate scoring methods, for the evaluation of focused retrieval systems".

The Question Answering track of INEX 2011 (QA) is about contextualizing tweets, i.e. answering questions of the form "What is this tweet about?" using a recent cleaned dump of the english Wikipedia [2]

[2] See the officiel web site of INEX 2011 Track: `https://inex.mmci.uni-saarland.de/tracks/qa/`

2.1 Document collection

The document collection has been rebuilt based on a recent dump of the English wikipedia from April 2011. Since we target a plain xml corpus for an easy extraction of plain text answers, all notes and bibliographic references were removed. Theses informations are difficult to handle. Then only 3,217,015 non empty Wikipedia pages (pages having at least on section) are used as corpus.

2.2 Topics

The commitee of INEX has defined 132 topics for the Track 2011. Each topic includes the title and the first sentence of a New York Times paper that were twitted at least two months after the wikipedia dump we use. Each topic was manually checked that there is related information in the document collection.

3 The summarization systems used

3.1 Cortex summarization system

Cortex [17, 18] is a single-document extract summarization system. It uses 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 2011 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 a perl program supplied by INEX organizers[3]. This program is coupled to Indri system[4] to obtain for each query, 50 documents from the whole corpus.

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 [19, 18] for a detailed description of these metrics) to evaluate the sentence's relevance.

1. The frequency of words.
2. The overlap between the words of the query (R).
3. The entropy the words (E).
4. The shape of text (Z).
5. The angle between question and document vectors (A).

[3] See: `http://qa.termwatch.es/data/getINEX2011corpus.pl.gz`

[4] 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. See: `http://www.lemurproject.org/indri/`

6. The sum of Hamming weights of words per segment times the number of different words in a sentence.
7. The sum of Hamming weights of the words multiplied by word frequencies.
8. The words interaction (I).
9. ...

By exemple, the topic-sentence overlap measure 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.

$$\text{Overlap}(T, S) = \frac{card(S \cap T)}{card(T)}$$

The system scores each sentence with a decision algorithm that relies on the normalized metrics. Before combining the votes of the metrics, these are partitionned 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 s given a query q is calculated with:

$$\begin{aligned} &\text{if}(\alpha > \beta) \\ &\qquad \text{then} \;\; \text{Score}(s, q) = 0.5 + \alpha/\Gamma \\ &\qquad \text{else} \;\; \text{Score}(s, q) = 0.5 - \beta/\Gamma \end{aligned}$$

The Cortex system is applied to each document of a topic and the summary is generated by concatenating higher score sentences.

3.2 The Enertex system

[21] shows that the energy of a sentence s reflects its weight, related to the others μ sentences $\mu = 1, ...P$, in the document. We applied

$$E_{s,\mu} = (S \times S^T)^2 \tag{1}$$

where S is the matrix sentences by terms and S^T their transposed, to summarization by extraction of sentences. The summarization algorithm includes three

modules. The first one makes the vectorial transformation of the text with filtering, lemmatisation/stemming and standardization processes. The second module applies the spins model and makes the calculation of the matrix of textual energy (1). We obtain the weighting of a sentence s by using its absolute energy values, by sorting according to

$$\sum_{\mu} |E_{s,\mu}| \tag{2}$$

So, the relevant sentences will be selected as having the biggest absolute energy. Finally, the third module generates summaries by displaying and concatenating of the relevant sentences. The two first modules are based on the Cortex system[5].

In order to calculate the similarity between every topic and the phrases we have used Textual Energy (1). Consequently the summary is formed with the sentences that present the maximum interaction energy with the query.

At first, the first 10 documents[6] of the cluster are concatenated into a single multi-document in chronological order. Placing the topic q (enriched or not) like the title of this long document. The Textual Energy between the topic and each of the other sentences in the document is computed using 1. Finally, we are only interested in recovering the row ρ of matrix E which corresponds to interaction energy of the topic q vs. the document. We construct the summary by sorting the most relevant values of ρ.

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)[7], in particular on the INEX 2011 QA Track (QA@INEX) `https://inex.mmci.uni-saarland.de/tracks/qa/`.

To evaluate the efficacity of Cortex and Enertex systems on INEX@QA track, we used the FRESA package[8].

4.1 INEX queries modification

Two different strategies were employed to generate the 132 queries from topics:

1. No pre-processing of topic.
2. Enrichment of the topic by manual definitions of terms.

1) No pre-processing or modification was applied on queries set. Summarizers use the query as a title of a big multi-document retrieved by Indri engine.

[5] See next section.
[6] We used only 10 documents by limits of memory.
[7] `http://www.inex.otago.ac.nz/`
[8] FRESA package is disponible at web site: `http://lia.univ-avignon.fr/fileadmin/axes/TALNE/downloads/index_fresa.html`

2) Enrichment of topic. The query has been mannualy enriched using the title and the definitions of each term present.

Table 1 shows an example of the results obtained by Cortex and Enertex systems using 50 or 10 documents respectively as input. The topic that the summary should answer in this case was the number 2011001:

```
<topic id="2011001">
<title>At Comic-Con, a Testing Ground for Toymakers</title>
  <txt>
       THIS summer's hottest toys won't be coming to a toy aisle
       near you. The only place to get them will be at Comic-Con
       International in San Diego.
  </txt>
</topic>
```

For query 2011001, strategy 2 give the next query:

q = "`At Comic-Con, a Testing Ground for Toymakers a place or area used for testing a product or idea a company that manufactures toys`"

Table 1 presents Cortex and Enertex results (queries enriched or not) in comparison with the INEX baseline (Baseline summary), and three baselines, that is, summaries including random n-grams (Random uni-grams) and 5-grams (Random 5-grams) and empty baseline. We observe that Cortex system is always better than others summarizers.

Table 1. Example of Summarization results on topic 2011001.

Summary type	uni-grams	bi-grams	SU4 bi-grams	FRESA average
Baseline summary	27.73262	35.17448	35.17507	32.69406
Empty baseline	36.20351	43.96798	43.93392	41.36847
Random unigrams	28.99337	36.80240	36.74110	34.17896
Random 5-grams	25.36871	32.78485	32.90045	30.35133
Cortex (Query=Title)	25.62371	32.93572	32.97384	**30.51109**
Cortex (Query=Title+Definition)	31.50112	39.18660	39.14994	36.61255
Enertex (Query=Title)	27.93538	35.36765	35.38713	32.89672

Table 2 presents the average over 15 queries of our three systems (queries numbers are: 2011001, 2011005, 2011010, 2011015, 2011020, 2011025, 2011030, 2011035, 2011040, 2011050, 2011055, 2011060, 2011071, 2011080 and 2011090).

5 Conclusions

We have presented the Cortex and Enertex summarization systems. The first one is based on the fusion process of several different sentence selection metrics.

Table 2. Averages (FRESA) of our three systems over 15 sampled queries.

System	FRESA averages
Cortex summary (Query = Title)	**47.6282**
Cortex summary (Query = Title + Definitons)	50.7814
Enertex summary (Query = Title)	50.5001

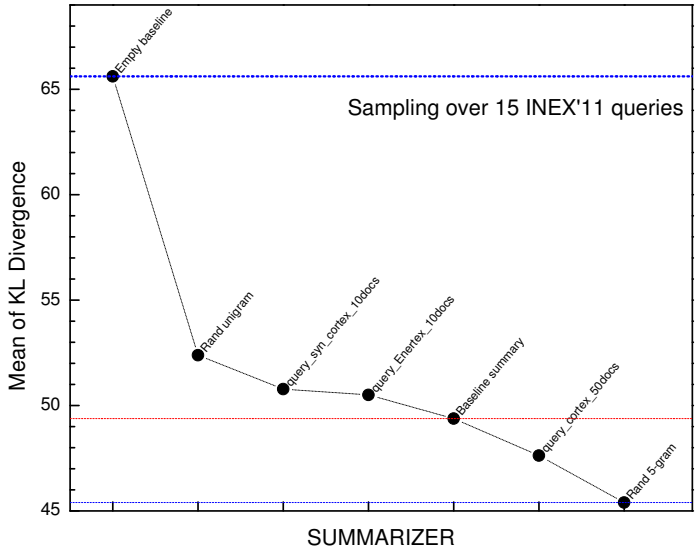


Fig. 1. Recall FRESA results for the 7 systems on INEX 2011 corpora.

The decision algorithm obtains good scores on the INEX 2011 task (the decision process is a good strategy without training corpus). The second one is based in Statistical mechanical concepts of spins models. On INEX 2011 track, Cortex summarizer has obtained very good results in the automatic FRESA evaluations. The Enertex summarizer results are less good than Cortex system. We explain this because the Enertex summarizer has used only sets of 10 documents (the Cortex system uses data sets of 50 documents). Others tests, with 50 documents, are in progress using Enertex system. Manually query enrichment has dissapointed in this task. May be the set of documents recuperated by Indri are less pertinents if terms and their (several) definitions are used as queries.

6 Appendix

We presents the Cortex summary of topic 2011001.

With his partner, artist Jack Kirby, he co-created Captain America, one of comics" most enduring superhero es, and the team worked extensively on such features at DC Comics as the 1940s Sandman and Sandy the Golden Boy, and co-created the Newsboy Legion, the Boy Commandos, and Manhunter. The New York Comic Con is a for-profit event produced and managed by Reed Exhibitions, a division of Reed Business, and is not affiliated with the long running non-profit San Diego Comic-Con, nor the Big Apple Convention, now known as the Big Apple Comic-Con, which is owned by Wizard World. With the relaunch, "The Swots and the Blots" became a standard bearer for sophisticated artwork, as Leo Baxendale began a three year run by adopting a new style, one which influenced many others in the comics field, just as his earlier "Beano" . Some of the strips from "Smash" survived in the new comic, including "His Sporting Lordship", "Janus Stark" and "The Swots and the Blots", but most were lost, although the "Smash" Annual continued to appear for many years afterwards . Other work includes issues of Marvel's "Captain America", "Captain Marvel", "The Power of Warlock", Ka-Zar in "Astonishing Tales", Ant-Man in "Marvel Feature", and "The Outlaw Kid", writing a short-lived revival of Doug Wildey 's Western series from Marvel's . Powell also did early work for Fox's "Wonderworld Comics "and" Mystery Men Comics"; Fiction House's "Planet Comics", where his strips included Gale Allen and the Women's Space Battalion ; Harvey 's "Speed Comics", for which he wrote and drew the feature Ted Parrish, ; Timely 's one-shot "Tough Kid Squad . His work in the 1950s included features and covers for Street and Smith 's "Shadow Comics" ; Magazine Enterprises " " Bobby Benson's B-Bar-B Riders ", based on the children's television series, and all four issues of that publisher's "Strong Man"; and, for Harvey Comics, many war, romance, and horror stories, as well as work for the comics "Man . The winner of each round receives the same contract deal as Comic Creation Nation winners, ie a team of professional artists creating a 22-page comic using the writer's script and background elements, shared ownership of development rights, a publicity campaign built around their property, and the Zeros 2 Heroes team pushing the property towards other entertainment venues. The duo left the comic book business to pursue careers in feature films and were involved producing the feature film adaptation of "Mage" by legendary comic book creator Matt Wagner with Spyglass Entertainment, and had various projects with Mike Medavoy, Mark Canton, Akiva Goldsman, and Casey Silver. The company has been very active by participating in various events, including the L_A Times Festival of Books, Heroes Con, San Diego Comic Con, Toronto Fan Expo, D23 Disney Convention, and Baltimore Comic Con. The ECBACC STARS Workshop is an ECBACC initiative designed to use comic book art and imagery as a vehicle to foster creativity and promote literacy, with a secondary focus on introducing participants to the various career options that exist within the comic book industry. Co-presenting with comics author and scholar Danny Fingeroth during a Comics Arts Conference panel at 2008's Comic-Con International in San Diego, California, the creators explained how the first Rocket Llama story evolved into a webcomic. In addition to "The Ongoing Adventures of Rocket Llama", e-zine features from the "Rocket Llama

Ground Crew" "include" "Action Flick Chick" movie reviews by G4TV 's Next Woman of the Web, cosplayer "Katrina Hill"; "The Action Chick" webcomic; "Marko's Corner" comics, cartoon arts, and podcasts by "Marko Head"; "Reddie Steady" comics for college newspapers; "The Workday Comic", the 8-hour . After the redrawn number #112's online publication came the serialized time travel story #136-137, Time Flies When You"re on the Run, appearing one page at a time throughout each week and expanding the cast with characters like the scientist Professor Percival Penguin and cavedogs who joined them by stowing away in the heroes" time rocket during the supposed previous . Other tie-in products were produced, including lunchboxes, 3-D puffy stickers, party supplies, paintable figurines, Underoos, coloring and activity books, Stain-A-Sticker, Justice League of America Skyscraper Caper game, sunglasses, playhouses, belt buckles, sneakers, cufflinks, signature stamp sets, coloring play mats, drinking glasses/tumblers, model kits, soap, stain painting sets, calendars, Play-Doh sets, jointed wall figures, wrist watches, jigsaw puzzles (Jaymar and .

References

1. ANSI. American National Standard for Writing Abstracts. Technical report, American National Standards Institute, Inc., New York, NY, 1979. (ANSI Z39.14.1979).
2. Juan Manuel Torres-Moreno. *Resume automatique de documents: une approche statistique*. Hermes-Lavoisier, 2011.
3. H. P. Luhn. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, 2(2):159, 1958.
4. H. P. Edmundson. New Methods in Automatic Extracting. *Journal of the ACM (JACM)*, 16(2):264–285, 1969.
5. I. Mani and M. Mayburi. *Advances in automatic text summarization*. The MIT Press, U.S.A., 1999.
6. Gregory Salton. *The SMART Retrieval System - Experiments un Automatic Document Processing*. Englewood Cliffs, 1971.
7. Gregory Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
8. Juan-Manuel Torres-Moreno, Patricia Velazquez-Morales, and Jean-Guy Meunier. Condensés automatiques de textes. *Lexicometrica. L'analyse de données textuelles : De l'enquête aux corpus littéraires*, Special(www.cavi.univ-paris3.fr/lexicometrica), 2004.
9. C. Jacquemin and P. Zweigenbaum. Traitement automatique des langues pour l'accès au contenu des documents. *Le document en sciences du traitement de l'information*, 4:71–109, 2000.
10. Jose Abracos and Gabriel Pereira Lopes. Statistical Methods for Retrieving Most Significant Paragraphs in Newspaper Articles. In Inderjeet Mani and Mark T. Maybury, editors, *ACL/EACL97-WS*, Madrid, Spain, July 11 1997.
11. Simone Teufel and Marc Moens. Sentence Extraction as a Classification Task. In Inderjeet Mani and Mark T. Maybury, editors, *ACL/EACL97-WS*, Madrid, Spain, 1997.
12. Eduard Hovy and Chin Yew Lin. Automated Text Summarization in SUMMARIST. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 81–94. The MIT Press, 1999.

13. Julian Kupiec, Jan O. Pedersen, and Francine Chen. A Trainable Document Summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 68–73, 1995.
14. Anastasios Tombros, Mark Sanderson, and Phil Gray. Advantages of Query Biased Summaries in Information Retrieval. In Eduard Hovy and Dragomir R. Radev, editors, *AAAI98-S*, pages 34–43, Stanford, California, USA, March 23–25 1998. The AAAI Press.
15. Judith D. Schlesinger, Deborah J. Backer, and Robert L. Donway. Using Document Features and Statistical Modeling to Improve Query-Based Summarization. In *DUC'01*, New Orleans, LA, 2001.
16. Shlomo Geva, Jaap Kamps, Ralf Schenkel, and Andrew Trotman, editors. *Comparative Evaluation of Focused Retrieval - 9th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2010, Vugh, The Netherlands, December 13-15, 2010, Revised Selected Papers*, volume 6932 of *Lecture Notes in Computer Science*. Springer, 2011.
17. J.M. Torres-Moreno, P. Velazquez-Moralez, and J. Meunier. *CORTEX, un algorithme pour la condensation automatique de textes.* In *ARCo*, volume 2, page 365, 2005.
18. Juan Manuel Torres-Moreno, Pier-Luc St-Onge, Michel Gagnon, Marc El-Bèze, and Patrice Bellot. Automatic summarization system coupled with a question-answering system (qaas). *in CoRR*, abs/0905.2990, 2009.
19. J.M. Torres-Moreno, P. Velazquez-Morales, and J.G. Meunier. *Condensés de textes par des méthodes numériques. JADT*, 2:723–734, 2002.
20. G. Salton. *Automatic text processing*, chapter 9. Addison-Wesley Longman Publishing Co., Inc., 1989.
21. Silvia Fernández, Eric SanJuan, and Juan Manuel Torres Moreno. Textual energy of associative memories: Performant applications of enertex algorithm in text summarization and topic segmentation. In *MICAI*, pages 861–871, 2007.

The REG summarization system with question expansion and reformulation at QA@INEX track 2011

Jorge Vivaldi and Iria da Cunha

Institut Universitari de Lingüística Aplicada - UPF
Barcelona
{iria.dacunha,jorge.vivaldi}@upf.edu
http://www.iula.upf.edu

Abstract. In this paper, our strategy and preliminary results for the INEX@QA 2011 question-answering task are presented. In this task, a set of 50 documents is provided by the search engine Indri, using some queries. The initial queries are titles associated wih tweets. A reformulation of these queries is carried out using terminological and names entities information. To design the queries to obtain the documents with INDRI, the full process is divided into 2 steps: a) both titles and tweets are POS tagged, and b) queries are expanded or reformulated, using: terms and name entities included in the title, terms and name entities found in the tweet related to those ones, and Wikipedia redirected terms and name entities from those ones included in the title. In our work, the automatic summarization system REG is used to summarize the 50 documents obtained with these queries. The algorithm models a document as a graph, to obtain weighted sentences. A single document is generated, considered as the answer of the query. This strategy, combining summarization and question reformulation, obtains preliminary good results with the automatic evaluation system FRESA.

Key words: INEX, Question-Answering, Terms, Name Entities, Wikipedia, Automatic Summarization, REG.

1 Introduction

The Question-Answering (QA) task can be related to two types of questions: very precise questions (expecting short answers) or complex questions (expecting long answers, including several sentences). The objective of the QA track of INEX 2011 (Initiative for the Evaluation of XML retrieval) is oriented to the second one. Specifically, the QA task to be performed by the participating groups of INEX 2011 is contextualizing tweets, i.e. answering questions of the form "what is this tweet about?" using a recent cleaned dump of the Wikipedia (WP). The general process involves: tweet analysis, passage and/or XML elements retrieval and construction of the answer. Relevant passages segments should contain relevant information but contain as little non-relevant information as possible. The

used corpus in this track contains all the texts included into the English WP. The expected answers are short documents of less than 500 words exclusively made of aggregated passages extracted from the WP corpus.

Thus, we consider that automatic extractive summarization systems could be useful in this QA task, taking into account that a summary can be defined 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.

To carry out this task, we have decided to use REG (Torres-Moreno and Ramírez, 2010; Torres-Moreno et al., 2010), an automatic extractive summarization system based on graphs. We have performed some expansions and reformulations of the initial INEX@QA 2011 queries, using terms and name entities, in order to obtain a list of terms related with the main topic of all the questions.

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.).

This paper is organized as follows. In Section 2, the summarization system REG is shown. In Section 3, queries expansions and reformulations are explained. In Section 4, experimental settings and results are presented. Finally, in Section 5, some preliminary conclusions are exposed.

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 and name entities (T&NE) included into the titles and the associated tweets are representative of the main subject of these texts. If this assumption is true, the results of quering the search engine with an optimized list of T&NE should be better that simply to use the title of the tweet as search query.

In order to demonstrate such hypothesis, we have decided to generate 3 different queries to Indri:

a) Using the initial query string (the title of the tweet).

b) Enriching the initial query with a list of those T&NE from the tweet that are related to the T&NE already present in the initial query. Redirections from WP are also considered.

c) Using only the above mentioned list of T&NE obtained from the previous step.

The procedure for obtaining this list from the tweet may be sketched as follows:

1. To find, in both query and tweet strings, for T&NE and verify that such strings are also present in WP. This procedure is again splitted in two stages: first finding the T&NE, and then looking for such unit in WP. The last step is close to those presented in Milne and Witten (2008), Strube and Ponzetto (2006) or Ferragina and Scaiella (2010).
2. To compare each unit in the tweet with all the units found in the query. Such comparison is made using the algorithm described in Milne and Witten (2007).
3. To choose only those units whose relatedness value are higher than a given threshold.

Figure 1 shows how the enriched query is built. From the query string we obtain a number of terms: (t_{tm}); we repeat the procedure with the tweet string (t_{tm}). We look for such terms in the WP; only the terms (or a substring of them) that have an entry in WP are considered. Then, we calculate the semantic relatedness among each term of the tweet (t_{tn}) with each term of the query. Only those terms of the tweets whose similarity with some of the term of the query is higher that a threshold value are taken into account. Assuming a query and tweet string as shown in Figure 1, each t_{tm} is compared with all t_{qn}. As a result of such comparisons, only t_{t2} and t_{t4} will be inserted in the enriched query because t_{t1} and t_{t3} will be rejected.

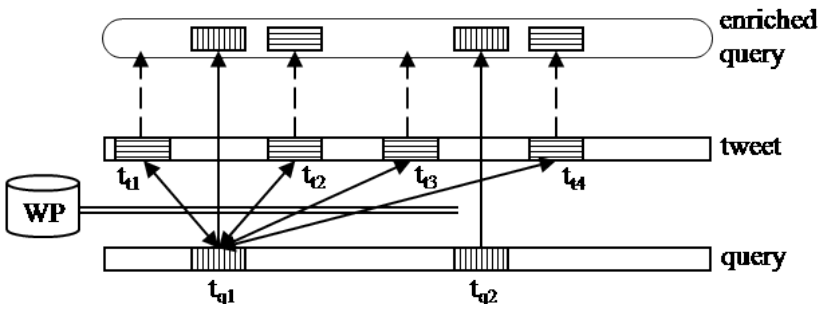


Fig. 1. Advanced query terms selection.

As mentioned above, the comparison among WP articles is done by using the algorithm described in Milne and Witten (2007). The idea is pretty simple and it is based in the links extending each article: higher is the number of number of such links shared by both article higher is their relatedness. Figure 2 shows an outline about how to calculate the relatedness among the WP pages "automobile" and "global warming". It is clear that some outgoing links ("air pollution", "alternative fuel", etc.) are shared by both articles while other links not ("vehicle", "Henry Ford", "Ozone"). From this idea it is possible to build such relatedness measure (see Milne and Witten, 2007 for details).

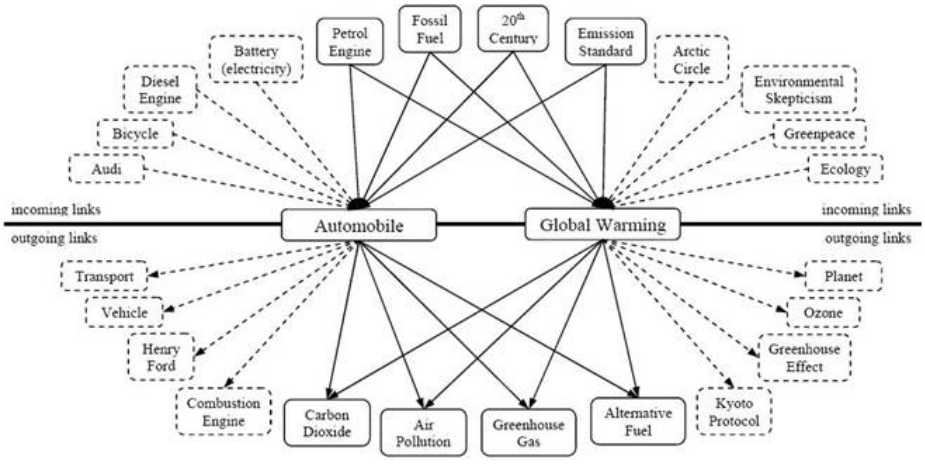


Fig. 2. Looking for the relation among WP articles (reprinted from Milne and Witten, 2007).

Let's see an example of some queries generated in our experiment. For the initial query (the title of the tweet) "Obama to Support Repeal of Defense of

Marriage Act", we extract the term "defense" and "marriage act", and the name entity "Obama". Moreover, we add the name entity "Barack Obama", since there is a redirection link in WP from "Obama" to "Barack Obama". Finally, some terms ("law", "legal definition", "marriage union", "man", "woman", "support" and "gay rights") and name entities ("President Obama" and "White House") semantically related with the units of the title are selected.

The process of building of the 3 different queries for this same example is the following:

1. The initial query is the title of the tweet:
   Obama to Support Repeal of Defense of Marriage Act.
   In this title the following T&NEs have been found: "Obama", "Defense of Marriage Act".
2. The expanded query is built from the body of the tweet:
   WASHINGTON - President Obama will endorse a bill to repeal the law that limits the legal definition of marriage to a union between a man and a woman, the White House said Tuesday taking another step in support of gay rights.
   The T&NE found in this string are: "Obama", "Defense of Marriage Act", "Barack Obama", "law", "legal definition", "marriage", "union", "man", "woman", "step", "support", "gay rights", "President Obama", "White House" and "Tuesday". The built expanded query contains the following query terms: "Obama to Support Repeal of Defense of Marriage Act", "Obama", "Barack Obama", "President Obama", "White House", "marriage", "union", "man", "gay rights" and "woman". Note that some terms are dropped (like "step" and "Tuesday") because they do no have any relation to the T&NE found in the title of the tweet, and some new query terms have been added ("President Obama") using WP redirection links.
3. The reformulated query is built using only the list of T&NE: "Obama", "Barack Obama", "President Obama", "White House", "marriage", "union", "man", "gay rights" and "woman".

The term and name entity extraction was carried out manually. 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 2011 are several, that is, they do not correspond to an unique domain. Also the relatedness among WP pages is manually done because our implementation of the relatedness measure relies in a relatively old dump of English WP.

4 Experiments Settings and Results

In this study, we used the document sets made available for the INEX 2011 QA Track (QA@INEX). These sets of documents where provided by the search

engine Indri.[1] REG produced multidocument summaries using the set of 50 documents provided by Indri using all the initial queries of the track and the expansions and reformulations following our strategy.

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`.

Tables 1, 2 and 3 show an example (document ID = 2011041) of the results obtained by REG with 50 documents as input and using the 3 different queries (a, b and c, respectively). These tables present REG results in comparison with an intelligent baseline (Baseline summary) and 3 other baselines: summaries including random n-grams (Random unigram), 5-grams (Random 5-gram) and empty words (Empty baseline). In this example, the reformulated query obtains better results (56.58465) than the initial query (59.04529) using FRESA.

Table 1. Example of REG results over the document 2011041 using query a.

Distribution type	unigram	bigram	with 2-gap	Average
Baseline summary	51.88447	60.37075	60.67855	57.64459
Empty baseline	74.53241	83.71035	83.95558	80.73278
Random unigram	55.86427	65.06233	65.25039	62.05900
Random 5-gram	47.71473	56.34721	56.71558	53.59251
Submitted summary	53.09912	61.89060	62.14615	59.04529

In table 4, the average of the results of 6 summaries selected at random from the 50 summaries is presented (IDs = 2011144, 2011026, 2011041, 2011001, 2011183, 2011081). In this case, the situation regarding the best query changes, and the initial query (the title) obtains the best results. With regard to the baselines, Baseline summary and Random 5-gram are always better than our system. However, in general, our system is better than Random unigram and Empty baseline. Nevertheless, we consider that this evaluation, including only 6 texts, is very preliminary and that it is necessary to wait for the final official evaluation of INEX 2011, in order to obtain a complete evaluation of the results.

[1] 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 2. Example of REG results over the document 2011041 using query b.

Distribution type	unigram	bigram	with 2-gap	Average
Baseline summary	48.75833	57.14385	57.36983	54.42401
Empty baseline	70.35451	79.43793	79.60188	76.46477
Random unigram	52.37199	61.48490	61.59465	58.48385
Random 5-gram	45.73679	54.36620	54.71034	51.60444
Submitted summary	52.08416	60.80215	61.04838	57.97823

Table 3. Example of REG results over the document 2011041 using query c.

Distribution type	unigram	bigram	with 2-gap	Average
Baseline summary	49.62939	58.06352	58.40945	55.36745
Empty baseline	73.65646	82.85850	83.14989	79.88829
Random unigram	53.94457	63.17036	63.39601	60.17031
Random 5-gram	45.36140	53.99297	54.45397	51.26945
Submitted summary	50.67031	59.34556	59.73806	56.58465

The 6 selected summaries can be divided in 2 sets of 3 summaries using: a) reformulated queries with a high quantity of terms and/or name entities (IDs = 2011144, 2011026, 2011041) and b) reformulated queries with a low quantity of terms and/or name entities (IDs = 2011001, 2011183, 2011081). The longest query cotains 11 units and the shortest includes 3 units. Table 5 includes a comparative evaluation between both results. It is interesting to observe that the summaries obtained using queries with a high quantity of terms and name entities obtain better results with the query c) (that is, using the reformulated query). However, when the queries do not include lots of terms, the best results are obtained with the initial queries (that is, the titles).

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 have applied this approach to the INEX@QA 2011 task, using 3 types of queries, the initial ones (titles of tweets) and other ones extracting T&NE from titles, and selecting those units that are semantically related to T&NE present in the associated tweets. Semantic relatedness is obtained directly from WP.

Our preliminary experiments have shown that our system is always better than the 2 simple baselines, but in comparison with the 2 more intelligent baselines the performance is variable. Morevoer, this preliminary evaluation shows that the reformulated queries obtain bestter results than the initial queries when the quantity of extracted terms and name entities is high.

Table 4. Average of results using the 3 queries and REG.

Average	Query a	Query b	Query c
Baseline summary	45.313355	48.297996	48.668063
Empty baseline	59.050726	63.367683	66.331401
Random unigram	46,563255	48.538538	49.83195
Random 5-gram	41.434218	43.78428	43.813228
Submitted summary	45.530785	48.5865	49.421386

Table 5. Comparison between summaries obtained with short and long queries.

Average	Query a	Query b	Query c
Summaries with short query	39.882106	43.721356	47.69209
Summaries with long query	51.179463	53.451656	51.150683

We consider that, over the INEX-2011 corpus, REG obtained good results in the automatic evaluations, but now it is necessary to wait for the human evaluation and the evaluation of other systems to compare with.

References

1. Edmunson, H. P. (1969). *New Methods in Automatic Extraction.* Journal of the Association for Computing Machinery 16. 264-285.
2. Ferragina, P. and Scaiella, U. (2010). *TAGME: On-the-fly Annotation of Short Text Fragments (byWikipedia Entities).* 19th International Conference on Information and Knowledge Management. Toronto, Canada.
3. Gaizauskas, R.; Herring, P.; Oakes, M.; Beaulieu, M.; Willett, P.; Fowkes, H.; Jonsson, A. (2001). *Intelligent access to text: Integrating information extraction technology into text browsers.* En Proceedings of the Human Language Technology Conference. San Diego. 189-193.
4. Lal, P.; Reger, S. (2002). *Extract-based Summarization with Simplication.* In Proceedings of the 2nd Document Understanding Conference at the 40th Meeting of the Association for Computational Linguistics. 90-96.
5. Lin, C.-Y. (2004). *ROUGE: A Package for Automatic Evaluation of Summaries.* In Proceedings of Text Summarization Branches Out: ACL-04 Workshop. 74-81.
6. Milne, D.; Witten, I.H. (2007). *An effective , low-cost measure of semantic relatedness obtained from Wikipedia links Obtaining Semantic Relatedness from.* Association for the Advancement of Artificial Intelligence.
7. Milne, D.; Witten, I.H. (2008). *ALearning to link with wikipedia.* Proceedings of the 17th ACM conference on Information and knowledge mining. New York.
8. Nanba, H.; Okumura, M. (2000). *Producing More Readable Extracts by Revising Them.* In Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000). Saarbrucken. 1071-1075.
9. Ono, K.; Sumita, K.; Miike, S. (1994). *Abstract generation based on rhetorical structure extraction.* In Proceedings of the International Conference on Computational Linguistics. Kyoto. 344-348.

10. Paice, C. D. (1990). *Constructing literature abstracts by computer: Techniques and prospects*. Information Processing and Management 26. 171-186.
11. Radev, D. (1999). Language Reuse and Regeneration: Generating Natural Language Summaries from Multiple On-Line Sources. New York, Columbia University. [PhD Thesis]
12. Saggion, H.; Lapalme, G. (2002). *Generating Indicative-Informative Summaries with SumUM*. Computational Linguistics 28(4). 497-526.
13. Saggion, H.; Torres-Moreno, J-M.; da Cunha, I.; SanJuan, E.; Velázquez-Morales, P.; SanJuan, E. (2010). *Multilingual Summarization Evaluation without Human Models*. In Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010). Pekin.
14. Strube, M.; Ponzetto, S.P. (2006). *WikiRelate! Computing Semantic Relatedness Using Wikipedia*. Association for Artificial Intelligence.
15. Torres-Moreno, J-M.; Saggion, H. da Cunha, I. SanJuan, E. Velázquez-Morales, P. SanJuan, E.(2010a). *Summary Evaluation With and Without References*. Polibitis: Research journal on Computer science and computer engineering with applications 42.
16. Torres-Moreno, J-M.; Saggion, H.; da Cunha, I.; Velázquez-Morales, P.; SanJuan, E. (2010b). *Ealuation automatique de résumés avec et sans référence*. In Proceedings of the 17e Conférence sur le Traitement Automatique des Langues Naturelles (TALN). Université de Montréal et Ecole Polytechnique de Montréal: Montreal (Canada).
17. Torres-Moreno, J-M.; Ramírez, J. (2010). *REG : un algorithme glouton appliqué au résumé automatique de texte*. JADT 2010. Roma, Italia.
18. Torres-Moreno, J-M.; Ramírez, J.; da Cunha, I. (2010). *Un resumeur a base de graphes, indépendant de la langue*. In Proceedings of the International Workshop African HLT 2010. Djibouti.
19. Torres-Moreno, J. M.; Velázquez-Morales, P.; Meunier, J. G. (2002). *Condensés de textes par des méthodes numériques*. En Proceedings of the 6th International Conference on the Statistical Analysis of Textual Data (JADT). St. Malo. 723-734.

Overview of the INEX 2011 Focused Relevance Feedback Track

Timothy Chappell[1] and Shlomo Geva[2]

[1] Queensland University of Technology,
t.chappell@student.qut.edu.au
[2] Queensland University of Technology,
s.geva@qut.edu.au

Abstract. The INEX 2011 Focused Relevance Feedback track was run in mostly identical form to the INEX 2010 Focused Relevance Feedback track[2]. Due to the limited number of submissions, this overview is being presented as a preproceedings paper only and is not intended to be published with the conference proceedings. As such, this paper is virtually identical to the track overview presented at INEX 2010 with the main changes being differences between the submissions for the two tracks.

1 Introduction

This paper presents an overview of the INEX 2011 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 2009 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 *focused*. This means that both the search results and the feedback are specified as passages, or as XML elements, in documents - rather than as whole documents. Second, the evaluation is performed over a closed set of documents and assessments, and hence the evaluation is exhaustive, reliable and less dependent on the specific search engine in use. By reusing the relatively small topic assessment pools, having only several hundred documents per topic, the search engine quality can largely be taken out of the equation. Third, the evaluation is performed over executable implementations of relevance feedback algorithms rather than being performed over result submissions. Finally, the entire evaluation platform is reusable and over time can be used to measure progress in focused relevance feedback in an independent, reproducible, verifiable, uniform, and methodologically sound manner.

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 qrels 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 RFInterface {
    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.

Before the track submission date, participants were also provided with an optional binary interface JAR file to allow participants to supply an algorithm

in the form of native client code. The JAR file acts as a driver for the native client, passing information back and forth using pipes.

5 Results

5.1 Submissions

Two groups submitted a total of four Relevance Feedback Modules to the INEX 2011 Relevance Feedback track- down from nine submissions to the INEX 2010 Relevance Feedback track. QUT resubmitted the reference Relevance Feedback Module described in the next paragraph while the University of Otago submitted three native client submissions using the supplied driver.

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[3] as a base for a modified Rocchio[4] 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 *getXPath* methods.

The University of Otago made three submissions of a native client that uses the ATIRE search engine with various settings.

5.2 Evaluation

To evaluate the results, the first 20 topics from the INEX 2009 Ad Hoc track were chosen as the data set used for the evaluation. This was chosen due to the fact that the Ad Hoc track was not run in 2011, the INEX 2010 Ad Hoc track results were already used in the INEX 2010 Focused Relevance Feedback track and the INEX 2008 Ad Hoc track were used as training data for the reference submission.

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.

Trec eval reports results using a variety of different metrics, including interpolated recall-precision, average precision, exact precision and R-precision. Recall-precision reports the precision (the fraction of relevant documents returned out of the documents returned so far) at varying points of recall (after a given portion of the relevant documents have been returned.) 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.

As the Otago submissions do not make use of relevance feedback, alternative no-feedback results for them have not been listed for these submissions. The reference run therefore appears twice; as Reference when feedback is used and Reference (NF) without applying feedback. The Otago submissions have abbreviated names for clarity, but Otago (BM25) refers to untuned BM25 applied as-is. Otago (Rocchio) refers to BM25 with Rocchio pseudo-relevance feedback. Otago (Tuned) refers to BM25 with Rocchio, stemming and tuning.

Run	Average Precision	R-Precision
Reference	0.4219	0.4126
Reference (NF)	0.3376	0.3361
Otago (BM25)	0.3580	0.3597
Otago (Rocchio)	0.3576	0.3597
Otago (Tuned)	0.3656	0.3573

Table 1. Average precision and R-precision for submitted modules

The following table shows the exact precision of the submitted modules in the form of P@N precision, referring to the average proportion of relevant documents that have been returned after N documents have been returned. For example, a P@5 value of 0.5 means that, on average, 50% (or 2.5) of the first 5 documents returned were relevant.

	Reference	Ref (nf)	Otago (BM25)	Otago (Rocchio)	Otago (Tuned)
P@5	0.5	0.5	0.54	0.54	0.52
P@10	0.515	0.435	0.445	0.445	0.485
P@15	0.49	0.4	0.4167	0.4167	0.45
P@20	0.4675	0.385	0.3975	0.3975	0.3975
P@30	0.4367	0.35	0.3583	0.3583	0.3533
P@100	0.3095	0.242	0.226	0.226	0.2175
P@200	0.2327	0.174	0.183	0.1828	0.1788
P@500	0.1224	0.1182	0.1154	0.1154	0.1152
P@1000	0.0636	0.0636	0.0636	0.0636	0.0636

Table 2. Exact (P@N) precision

Another way of plotting the results is the previously described interpolated recall-precision curve. This has the downside of producing occasionally unex-

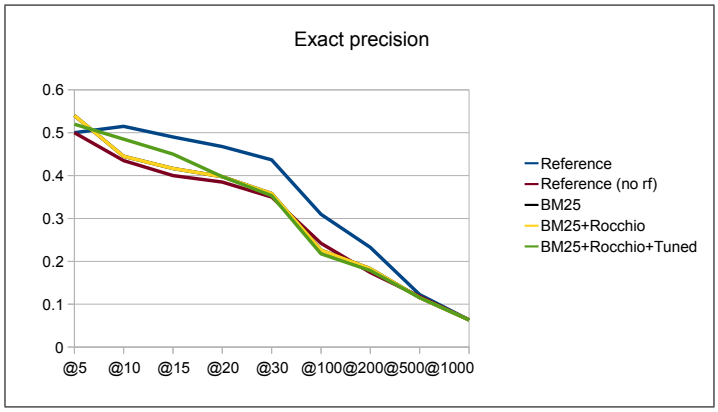


Fig. 1. Comparison of P@N precision of submitted Relevance Feedback modules

pected results due to the smoothing being enough to show improvement in the reference run even at 0.0, despite the fact that improvements don't occur in the first 5 results as shown in the exact precision plot.

In this case we present recall-precision data at 11 points from 0.0 to 1.0 to compare the submitted modules.

	Reference	Ref (nf)	Otago (BM25)	Otago (Rocchio)	Otago (Tuned)
0.0	0.87	0.8418	0.8481	0.8481	0.8679
0.1	0.6903	0.5741	0.5782	0.5782	0.5916
0.2	0.5877	0.5105	0.494	0.494	0.516
0.3	0.5132	0.433	0.4354	0.4354	0.4518
0.4	0.4718	0.3986	0.3943	0.3943	0.3916
0.5	0.4329	0.3078	0.3537	0.3537	0.3632
0.6	0.3912	0.2703	0.3128	0.3134	0.3305
0.7	0.3557	0.2488	0.279	0.279	0.2977
0.8	0.3015	0.2053	0.241	0.241	0.2407
0.9	0.217	0.1716	0.1761	0.1761	0.1753
1.0	0.1528	0.1291	0.1369	0.1369	0.1368

Table 3. Interpolated recall-preecision

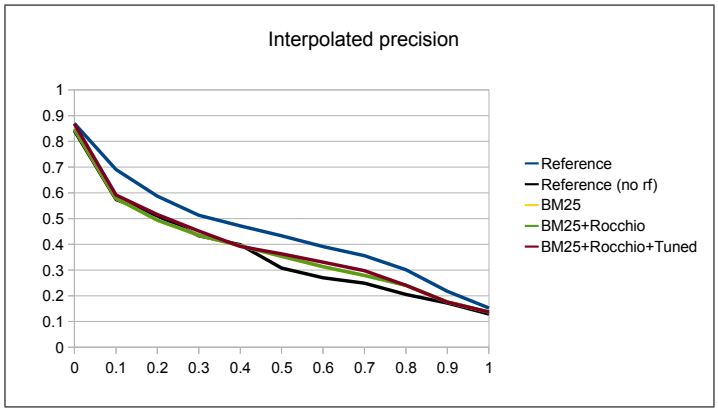


Fig. 2. Recall-precision comparison of Relevance Feedback Modules

6 Conclusion

We have presented the Focused Relevance Feedback track at INEX 2011. Despite the limited pool of participating organisations, the track has provided optimistic results, with improved results on INEX 2010.

7 Acknowledgements

We would like to thank all the participating organisations for their contributions and hard work.

References

1. C. Buckley. The trec eval IR evaluation package. *Retrieved January*, 1:2005, 2004.
2. T. Chappell and S. Geva. Overview of the inex 2010 relevance feedback track. *Comparative Evaluation of Focused Retrieval*, pages 303–312, 2011.
3. B. Goetz. The Lucene search engine: Powerful, flexible, and free. *Javaworld http://www. javaworld. com/javaworld/jw-09-2000/jw-0915-lucene. html*, 2002.
4. J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice-Hall Series in Automatic Computation, chapter 14, pages 313–323. Prentice-Hall, Englewood Cliffs NJ, 1971.

Snip!

Andrew Trotman, Matt Crane

Department of Computer Science
University of Otago
Dunedin, New Zealand

Abstract. The University of Otago submitted runs to the Snippet Retrieval Track and the Relevance Feedback tracks at INEX 2011. This paper discusses those runs. At the time of writing the snippet results had not been released. In the relevance feedback track no improvement was seen in whole-document retrieval when Rocchio blind relevance feedback was used.

Keywords: Snippet Generation, Relevance Feedback, Procrastination.

1 Introduction

In 2011 the University of Otago participated in two tracks it had not previously experiment with: the Snippet Retrieval Track and the Relevance Feedback Track. Six snippet runs were submitted and three relevance feedback runs were submitted. This contribution discusses those runs.

For details of the INEX document collection and the "rules" for the tracks the interested reader is referred to the track overview papers.

2 Snippets

2.1 Runs

A total of six runs were submitted:

First-p: in this run the snippet was the first 300 characters of the first <p> element in the document. This run was motivated by the observation that the start of a Wikipedia article typically contains an overview of the document and is therefore a good overview of the paper. In this run and all submitted runs the snippet was constructed so that it always started at the beginning of a word and ended at the end of a word and all XML tag content was discarded.

Tf-paragraph: in this run the snippet was the first 300 characters from the paragraph (<p>) element with the highest sum of query term occurrences (that is, for a two word query it is sum of the tf's of each term).

Tf-passage : in this run the snippet was the 300 character (word aligned) sliding window with the highest sum of query term occurrences. Since there are usually many such possible windows, the window was centered so that the distance from the start of the window to the first occurrence was the same (within rounding errors) as the distance from the last occurrence to the end of the window (measured in bytes).

These three runs together form experiment 1 in which the aim is to determine whether a snippet is better formed from an element or a passage.

Tficf-paragraph: in this run the snippet was the first 300 characters of the paragraph with the highest *tf* * *icf* weight were $icf_t = \log(C/c_t)$ were C is the length of the collection (in term occurrences) and c is the number of times term t occurs.

Tficf-passage: in this run the snippet was the first 300 character (word aligned) sliding window with the highest tf * icf score.

The tficf runs along with the tf runs form experiment 2 in which the aim is to determine the most effective way of choosing a passage or paragraph.

The final run was

KL-word-cloud: in this run the KL-divergence between each term in the document and the collection was used to order all terms in the document. From this ordering the top n were chosen as the snippet so that the snippet did not exceed 300 characters.

This final run forms experiment 3 in which the aim is to determine whether snippets are better form using extractive techniques (phrases) are better than summative techniques (word clouds).

2.2 Results

At the time of writing no results have been posted and so the results of the experiments remain unknown.

2.3 Observations from assessing

In total 6 topics were assessed by participants at the University of Otago. A post-assessment debriefing by the four assessors resulted in the following observations:

Snippets that included the title of the document were easier to assess than those that did not. It is subsequently predicted that those runs will generally score better than runs that did not. A recommendation is made to the track chairs to either automatically include the document title in the assessment tool, or to make it clear that the snippet may include the document title.

Snippets that were extractive from multiple parts of the document (included ellipses) generally contained multiple snippets each of which was too short to be useful and collectively not any better.

Snippets made from word clouds were not generally helpful.

Snippets that contained what appeared to be the section / subsection "path" through the document generally took so much space that the remaining space for the extractive snippet was too short for a useful snippet.

2.4 Further work

If the track is run in 2012 then from the observations it would be reasonable to submit a run that contains the document title, a single snippet extracted from the document, and the title of the section from which the snippet was extracted. The method of extraction is unclear and would depend on the results of the experiments submitted to this track in 2011.

3 Relevance Feedback

The purpose of the Otago relevance feedback runs was twofold.

The first purpose was to experiment with the INEX relevance feedback infrastructure. In particular, the infrastructure was written in Java but the search engine Otago uses is written in C++. The gateway from Java to C++ was provided by INEX.

The second purpose was to determine whether or not blind relevance feedback is an effective method of improving whole-document search results on Wikipedia. To this end the runs submitted by Otago ignored the human assessments and only returned whole documents.

3.1 Runs

A total of three runs were submitted:

BM25: in this run the documents are ranked using BM25 (k1=0.9, b=0.4). No relevance feedback was performed. This runs forms an out-of-the-box baseline. Is it the result of running the untrained ATIRE search engine over the documents.

BM25-RF: in this run the documents are ranked using BM25 (as above), then from the top 5 document the top 8 terms were selected using KL-divergence. These were then added to the query (according to Rocchio's algorithm) and BM25 was again used to get the top results. Terms added to the query had an equal weight to those already there, and terms already in the query could be added. The parameters 5 and 8 were chosen through learning over the training data.

BM25-RF-S: in this run the documents are ranked using BM25, then from the top 5 document the top 8 terms were selected using KL-divergence (as above). Additionally the s-stemmer was used in the initial and second query. Additional to this the parameters for BM25 were learned using a grid search (k1=0.5 b=0.5). Again training was on the INEX supplied training data.

In all runs blind relevance feedback was used and the user's assessments were ignored. As such these runs form a good baseline for ignoring the user.

3.2 Results

A subset of the official INEX published results are presented in Table 1 and the Precision / Recall graph is presented in Figure 1. The focused retrieval results are not presented as whold-document retrieval was used.

From the results, it appears as though relevance feedback has no effect on the performance of the search engine, but stemming does. It is already know that stemming works on the INEX Wikipedia collection, but unexpected that Rocchio Feedback does not. This result needs to be verified as it could be a problem with the run or a problem with the assessment method.

Table 1: INEX Published Results (from INEX)

Precision	Reference	Reference no feedback	Otago BM25	Otago BM25-RF	Otago BM25-RF-S
P@5	0.500	0.500	0.540	0.540	0.520
P@10	0.515	0.435	0.445	0.445	0.485
P@15	0.490	0.400	0.417	0.417	0.450
P@20	0.468	0.385	0.398	0.398	0.398
R-Precision	0.413	0.336	0.360	0.360	0.357

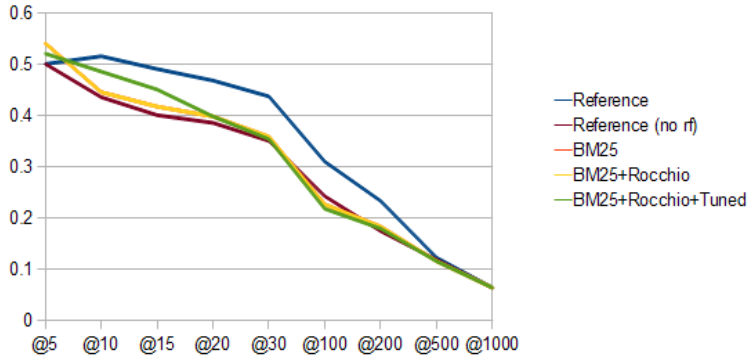


Figure 1: Official INEX Relevance Feedback Result (from INEX)

3.4 Further work

If the track is run in 2012 then it is reasonable to build on the baseline by including the user's assessments in the run. This could be done by performing a process similar to run BM25-RF-S at each assessment point and returning the top as-to un-seen document.

However, before any further work is done it is important to understand why relevance feedback does not appear to have an effect on this collection.

4. Conclusions

The University of Otago submitted six snippet runs and three feedback runs. These runs form baselines for experiments in improving the quality of the results in the search engine.

It is not clear why the relevance feedback method had no effect on precision. In further work this will be investigated.

References

The interested reader is referred to the overview papers of INEX 2011, especially the overview of the snippet and relevance feedback tracks.

Overview of the INEX 2011 Snippet Retrieval Track

Matthew Trappett[1], Shlomo Geva[1], Andrew Trotman[2], Falk Scholer[3], and Mark Sanderson[3]

[1] Queensland University of Technology, Brisbane, Australia
matthew.trappett@qut.edu.au, shlomo.geva@qut.edu.au
[2] University of Otago, Dunedin, New Zealand
andrew@cs.otago.ac.nz
[3] RMIT University, Melbourne, Australia
falk.scholer@rmit.edu.au, mark.sanderson@rmit.edu.au

Abstract. This paper gives an overview of the INEX 2011 Snippet Retrieval Track. The goal of the Snippet Retrieval Track is to provide a common forum for the evaluation of the effectiveness of snippets, and to investigate how best to generate snippets for search results, which should provide the user with sufficient information to determine whether the underlying document is relevant. We discuss the setup of the track, and the preliminary results.

1 Introduction

Queries performed on search engines typically return far more results than a user could ever hope to look at. While one way of dealing with this problem is to attempt to place the most relevant results first, no system is perfect, and irrelevant results are often still returned. To help with this problem, a short text snippet is commonly provided to help the user decide whether or not the result is relevant.

The goal of snippet generation is to provide sufficient information to allow the user to determine the relevance of each document, without needing to view the document itself, allowing the user to quickly find what they are looking for.

The INEX Snippet Retrieval track was run for the first time in 2011. Its goal is to provide a common forum for the evaluation of the effectiveness of snippets, and to investigate how best to generate informative snippets for search results.

2 Snippet Retrieval Track

In this section, we briefly summarise the snippet retrieval task, the submission format, the assessment method, and the measures used for evaluation.

2.1 Task

The task is to return a ranked list of documents for the requested topic to the user, and with each document, a corresponding text snippet describing the document. This text snippet should attempt to convey the relevance of the underlying document, without the user needing to view the document itself.

Each run is allowed to return up to 500 documents per topic, with a maximum of 300 characters per snippet.

2.2 Test Collection

The Snippet Retrieval Track uses the INEX Wikipedia collection introduced in 2009 — an XML version of the English Wikipedia, based on a dump taken on 8 October 2008, and semantically annotated as described in [1]. This corpus contains 2,666,190 documents.

The topics have been reused from the INEX 2009 Ad Hoc Track [2]. Each topic contains a short content only (CO) query, a content and structure (CAS) query, a phrase title, a one line description of the search request, and a narrative with a detailed explanation of the information need, the context and motivation of the information need, and a description of what makes a document relevant or not.

To avoid the 'easiest' topics, the 2009 topics were ranked in order of the number of relevant documents found in the corresponding relevance judgements, and the 50 with the lowest number were chosen.

For those participants who wished to generate snippets only, and not use their own search engine, a reference run was generated using BM25.

2.3 Submission Format

An XML format was chosen for the submission format, due to its human readability, its nesting ability (as information was needed at three hierarchical levels — submission-level, topic-level, and snippet-level), and because the number of existing tools for handling XML made for quick and easy development of assessment and evaluation.

The submission format is defined by the DTD given in Figure 1. The following is a brief description of the DTD fields. Each submission must contain the following:

- participant-id: The participant number of the submitting institution.
- run-id: A run ID, which must be unique across all submissions sent from a single participating organisation.
- description: a brief description of the approach used.

Every run should contain the results for each topic, conforming to the following:

- topic: contains a ranked list of snippets, ordered by decreasing level of relevance of the underlying document.

```
<!ELEMENT inex-snippet-submission (description,topic+)>
<!ATTLIST inex-snippet-submission
  participant-id CDATA #REQUIRED
  run-id CDATA #REQUIRED
>
<!ELEMENT description (#PCDATA)>
<!ELEMENT topic (snippet+)>
<!ATTLIST topic
  topic-id CDATA #REQUIRED
>
<!ELEMENT snippet (#PCDATA)>
<!ATTLIST snippet
  doc-id CDATA #REQUIRED
  rsv CDATA #REQUIRED
>
```

Fig. 1. DTD for Snippet Retrieval Track run submissions

- topic-id: The ID number of the topic.
- snippet: A snippet representing a document.
- doc-id: The ID number of the underlying document.
- rsv: The retrieval status value (RSV) or score that generated the ranking.

2.4 Assessment

To determine the effectiveness of the returned snippets at their goal of allowing a user to determine the relevance of the underlying document, manual assessment has been used. The documents for each topic were manually assessed for relevance based on the snippets alone, as the goal is to determine the snippet's ability to provide sufficient information about the document.

Each topic within a submission was assigned an assessor. The assessor, after reading the details of the topic, read through the top 100 returned snippets, and judged which of the underlying documents seemed relevant based on the snippets.

To avoid bias introduced by assessing the same topic more than once in a short period of time, and to ensure that each submission is assessed by the same assessors, the runs were shuffled in such a way that each assessment package contained one run from each topic, and one topic from each submission.

2.5 Evaluation Measures

Submissions are evaluated by comparing the snippet-based relevance judgements with the existing document-based relevance judgements, which are treated as a ground truth. This section gives a brief summary of the specific metrics used. In all cases, the metrics are averaged over all topics.

We are interested in how effective the snippets were at providing the user with sufficient information to determine the relevance of the underlying document, which means we are interested in how well the user was able to correctly determine the relevance of each document. The simplest metric is the mean precision accuracy (MPA) — the percentage of results that the assessor correctly assessed, averaged over all topics.

$$\text{MPA} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{1}$$

Due to the fact that most topics have a much higher percentage of irrelevant documents than relevant, MPA will weight relevant results much higher than irrelevant results — for instance, assessing everything as irrelevant will score much higher than assessing everything as relevant.

MPA can be considered the raw agreement between two assessors — one who assessed the actual documents (i.e. the ground truth relevance judgements), and one who assessed the snippets. Because the relative size of the two groups (relevant documents, and irrelevant documents) can skew this result, it is also useful to look at positive agreement and negative agreement to see the effects of these two groups.

Positive agreement (PA) is the conditional probability that, given one of the assessors judges a document as relevant, the other will also do so. This is also equivalent to the F_1 score.

$$\text{PA} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{2}$$

Likewise, negative agreement (NA) is the conditional probability that, given one of the assessors judges a document as relevant, the other will also do so.

$$\text{NA} = \frac{2 \cdot \text{TN}}{2 \cdot \text{TN} + \text{FP} + \text{FN}} \tag{3}$$

Mean normalised prediction accuracy (MNPA) calculates the rates for relevant and irrelevant documents separately, and averages the results, to avoid relevant results being weighted higher than irrelevant results.

$$\text{MNPA} = 0.5\frac{\text{TP}}{\text{TP} + \text{FN}} + 0.5\frac{\text{TN}}{\text{TN} + \text{FP}} \tag{4}$$

This can also be thought of as the arithmetic mean of recall and negative recall. These two metrics are interesting themselves, and so are also reported separately. Recall is the percentage of relevant documents that are correctly assessed.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5}$$

Negative recall (NR) is the percentage of irrelevant documents that are correctly assessed.

$$\mathrm{NR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{6}$$

The primary evaluation metric, which is used to rank the submissions, is the geometric mean of recall and negative recall (GM). A high value of GM requires a high value in recall and negative recall — i.e. the snippets must help the user to accurately predict both relevant and irrelevant documents. If a submission has high recall but zero negative recall (e.g. in the case that everything is judged relevant), GM will be zero. Likewise, if a submission has high negative recall but zero recall (e.g. in the case that everything is judged irrelevant), GM will be zero.

$$\mathrm{GM} = \sqrt{\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \cdot \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}} \tag{7}$$

3 Participation

Table 1. Participation in the Snippet Retrieval Track

ID	Institute	Runs
14	University of Otago	6
16	KASETSART UNIVERSITY	3
20	QUT	3
23	RMIT University	3
31	Radboud University Nijmegen	6
65	University of Minnesota Duluth	4
72	Jiangxi University of Finance and Economics	8
73	Peking University	8
77	Room 2318, Science Buildings 2, Peking University	6
83	ISMU	3

In this section, we discuss the participants and their approaches.

In the 2011 Snippet Retrieval Track, 50 runs were accepted from a total of 56 runs submitted. These runs came from 10 different groups, based in 7 different countries. Table 1 lists the participants, and the number of runs accepted from them.

Participants were allowed to submit as many runs as they wanted, but were required to rank the runs in order of priority, with the understanding that some runs may not be assessed, depending on the total number of runs submitted. To simplify the assessment process, 50 runs were accepted, to match the number of topics. This was achieved by capping the number of runs at 8 runs per partipating institute, and discarding any runs ranked below 8.

3.1 Participant approaches

The following is a brief description of the approaches used, as reported by the participants.

Queensland University of Technology The run 'QUTFirst300' is simply the first 300 characters of the documents in the reference run.

The run 'QUTFocused', again using the reference run, ignored certain elements, such as tables, images, templates, and the reference list. The tf-idf values were calculated for the key words found in each document. A 300 character window was then moved along the text, counting the total key words found in each window, weighted by their tf-idf scores. The highest scoring window was found, then rolled back to the start of the sentence to ensure the snippet did not start mid-sentence.

The run 'QUTIR2011A' selects snippets, using the reference run to select the appropriate documents. A topological signature is created from the terms of the query. Snippets are determined as 300 character passages starting from the `<p>` tag that is used to delineate paragraphs in the documents. Signatures are created for these snippets and compared against the original query signature. The closest match is used.

RMIT University The snippet generation algorithm was based on selecting highly ranked sentences which were ranked according to the occurrence of query terms. Nevertheless, it was difficult to properly identify sentence boundaries due to having multiple contributors with different writing styles. The main exception was detected when a sentence included abbreviations such as "'Dr. Smith"'. We did not do an analysis of abbreviations to address this issue in detail.

We processed Wikipedia articles before constructing snippets. Specifically, information contained inside the `<title>` and `<bdy>` was used to narrow the document content. We suggest that snippets should include information of the document itself instead of sources pointing to other articles. Therefore, the Reference section was ignored in our summarisation approaches. The title was concatenated to the leading scored sentences.

We used the query terms listed in the title, and we expanded them by addressing a pseudo relevance feedback approach. That is, the top 5 Wikipedia articles were employed for selecting the first 25 and 40 terms.

Radboud University Our previous study found that topical language model improves document ranking of ad-hoc retrieval. In this work, our attention is paid on snippets that are extracted and generated from the provided ranked list of documents.

In our experiments of the Snippet Retrieval Track, we hypothesize that the user recognizes certain combinations of terms in created snippets which are related to their information needs. We automatically extract snippets using terms

as the minimal unit. Each term is weighted according to their relative occurrance in its article and in the entire Wikipedia. The top K scoring terms are chosen for inclusion in the snippet. The term-extraction based snippets are then represented differently to the user. One is a cluster of words that indicate the described topic. Another is a cluster of semi-sentences that contains the topic information while preserving some language structure.

Jiangxi University of Finance and Economics p72-LDKE-$m_1m_2m_3m_4$, where m_i $(1 \leq i \leq 4)$ equals to 0 or 1, employs four different strategies to generate a snippet. Strategy 1 is dataset selection: using documents listed in reference runs ($m_1 = 0$) or Wikipedia 2009 dataset ($m_1 = 1$). Strategy 2 is snippet selection: using baseline method ($m_2 = 0$) or window method ($m_2 = 1$). According to the baseline method, after the candidate elements/nodes being scored and ranked, only the first 300 characters are extracted as snippet from the element/node has the highest score. Remain part of this snippet are extracted from the successive elements/nodes in case of the precedents are not long enough. While in the window method, every window that contain 15 terms are scored and those with higher scores are extracted as snippets. Strategy 3 is whether using ATG path weight ($m_3 = 1$) or not ($m_3 = 0$) in element retrieval model. The element retrieval model used in our system is based on BM25 and the works about ATG path weight has been published in CIKM 2010. Strategy 4 is whether reordering the XML document according to the reference runs ($m_4 = 0$) or not ($m_4 = 1$) after elements/nodes being retrieved.

Peking University In the INEX 2011 Snippet Retrieval Track, we retrieve XML documents based on both document structure and content, and our retrieval engine is based on the Vector Space Model. We use Pseudo Feedback method to expand the query of the topics. We have learned the weight of elements based on the cast of INEX2010 to enhance the retrieval performance, and we also consider the distribution of the keywords in the documents and elements, the more of the different keywords, the passage will be more relevant, and so is the distance of the keywords. We used method of SLCA to get the smallest sub-tree that satisfies the retrieval. In the snippet generation system, we use query relevance, significant words, title/section-title relevance and tag weight to evaluate the relevance between sentences and a query. The sentences with higher relevance score will be chosen as the retrieval snippet.

4 Snippet Retrieval Results

In this section, we present and discuss the preliminary evaluation results for the Snippet Retrieval Track.

At the time of writing, 39 of the 50 assessment packages have been completed. As not all of the assessments have been completed yet, the results presented here are preliminary results only. Each submission has been evaluated on a slightly

Table 2. Ranking of all runs in the Snippet Retrieval Track, ranked by GM (preliminary results only)

Rank	Run	Score
1	p20-QUTFirst300	0.5925
2	p72-LDKE-1111	0.5774
3	p72-LDKE-0101	0.5686
4	p73-PKU_ICST_REF_11a	0.5647
5	p23-baseline	0.5612
6	p65-UMD_SNIPPET_RETRIEVAL_RUN_3	0.5606
7	p72-LDKE-1110	0.5580
8	p14-top_tficf_passage	0.5539
9	p72-LDKE-1101	0.5497
10	p77-PKUSIGMASR01CLOUD	0.5479
11	p72-LDKE-1121	0.5467
12	p23-expanded-25	0.5446
13	p77-PKUSIGMASR03CLOUD	0.5415
14	p23-expanded-40	0.5411
15	p20-QUTFocused	0.5358
16	p77-PKUSIGMASR05CLOUD	0.5354
17	p72-LDKE-0111	0.5241
18	p72-LDKE-1011	0.5175
19	p72-LDKE-1001	0.5141
20	p35-97-ism-snippet-Baseline-Reference-run_01	0.5135
21	p73-PKU_106	0.5116
22	p73-PKU_105	0.5102
23	p20-QUTIR2011A	0.5082
24	p73-PKU_ICST_REF_11b	0.5027
25	p77-PKUSIGMASR04CLOUD	0.5020
26	p73-PKU_102	0.5018
27	p14-top_tf_passsage	0.4979
28	p35-98-ism-snippet-Baseline-Reference-run_01	0.4925
29	p65-UMD_SNIPPET_RETRIEVAL_RUN_4	0.4914
30	p31-SRT11DocTXT	0.4899
31	p31-SRT11ParsDoc	0.4878
32	p77-PKUSIGMASR02CLOUD	0.4863
33	p73-PKU_100	0.4852
34	p77-PKUSIGMASR06CLOUD	0.4837
35	p73-PKU_107	0.4831
36	p31-SRT11DocParsedTXT	0.4799
37	p14-top_tficf_p	0.4792
38	p14-top_tf_p	0.4764
39	p35-ism-snippet-Baseline-Reference-run_02	0.4708
40	p65-UMD_SNIPPET_RETRIEVAL_RUN_1	0.4667
41	p65-UMD_SNIPPET_RETRIEVAL_RUN_2	0.4610
42	p31-SRT11ParsStopDoc	0.4556
43	p14-first_p	0.4178
44	p73-PKU_101	0.4030
45	p31-SRT11ParsStopTerm	0.3731
46	p14-kl	0.3703
47	p31-SRT11ParsTerm	0.3563
48	p16-kas16-MEXIR-ALL	0.0000
49	p16-kas16-MEXIR-ANY	0.0000
50	p16-kas16-MEXIR-EXT	0.0000

Table 3. Additional metrics of all runs in the Snippet Retrieval Track (preliminary results only)

Run	MPA	MNPA	Recall	NR	PA	NA
p14-first_p	0.7885	0.5816	0.2562	0.9069	0.2717	0.8633
p14-kl	0.7579	0.5706	0.2771	0.8641	0.2001	0.8358
p14-top_tf_p	0.7669	0.5922	0.3414	0.8431	0.3016	0.8426
p14-top_tf_passsage	0.7572	0.6140	0.4065	0.8214	0.3123	0.8252
p14-top_tficf_p	0.7670	0.6055	0.3521	0.8589	0.3109	0.8471
p14-top_tficf_passage	0.7787	0.6375	0.4281	0.8468	0.3709	0.8548
p16-kas16-MEXIR-ALL	0.5692	0.2846	0.0000	0.5692	0.0000	0.6360
p16-kas16-MEXIR-ANY	0.8942	0.4471	0.0000	0.8942	0.0000	0.9355
p16-kas16-MEXIR-EXT	0.8786	0.4393	0.0000	0.8786	0.0000	0.9286
p20-QUTFirst300	0.8069	0.6625	0.4600	0.8651	0.4067	0.8703
p20-QUTFocused	0.7572	0.6223	0.4064	0.8383	0.3385	0.8311
p20-QUTIR2011A	0.7823	0.6190	0.3681	0.8699	0.3327	0.8529
p23-baseline	0.7928	0.6459	0.4219	0.8698	0.3739	0.8629
p23-expanded-25	0.7803	0.6248	0.3867	0.8629	0.3316	0.8582
p23-expanded-40	0.7555	0.6202	0.4034	0.8369	0.3376	0.8354
p31-SRT11DocParsedTXT	0.7938	0.6135	0.3345	0.8926	0.3194	0.8674
p31-SRT11DocTXT	0.8023	0.6211	0.3672	0.8750	0.3315	0.8589
p31-SRT11ParsDoc	0.7885	0.6206	0.3570	0.8843	0.3241	0.8631
p31-SRT11ParsStopDoc	0.8041	0.6213	0.3439	0.8986	0.3078	0.8723
p31-SRT11ParsStopTerm	0.7562	0.5759	0.3025	0.8493	0.2396	0.8222
p31-SRT11ParsTerm	0.7803	0.5726	0.2566	0.8885	0.2330	0.8495
p35-97-ism-snippet-Baseline-Reference-run_01	0.7969	0.6271	0.3834	0.8709	0.3552	0.8622
p35-98-ism-snippet-Baseline-Reference-run_01	0.7874	0.6161	0.3483	0.8840	0.3179	0.8576
p35-ism-snippet-Baseline-Reference-run_02	0.8000	0.6088	0.3222	0.8954	0.3055	0.8700
p65-UMD_SNIPPET_RETRIEVAL_RUN_1	0.7564	0.5899	0.3342	0.8457	0.2938	0.8335
p65-UMD_SNIPPET_RETRIEVAL_RUN_2	0.7577	0.5865	0.3284	0.8446	0.2900	0.8364
p65-UMD_SNIPPET_RETRIEVAL_RUN_3	0.7726	0.6293	0.4212	0.8373	0.3673	0.8445
p65-UMD_SNIPPET_RETRIEVAL_RUN_4	0.7864	0.6041	0.3305	0.8776	0.3162	0.8573
p72-LDKE-0101	0.7400	0.6328	0.4710	0.7946	0.3678	0.8120
p72-LDKE-0111	0.7567	0.6127	0.4115	0.8139	0.3357	0.8258
p72-LDKE-1001	0.7441	0.6162	0.3999	0.8325	0.3242	0.8202
p72-LDKE-1011	0.7926	0.6254	0.3752	0.8755	0.3466	0.8624
p72-LDKE-1101	0.7582	0.6280	0.4416	0.8144	0.3548	0.8265
p72-LDKE-1110	0.7582	0.6324	0.4635	0.8012	0.3838	0.8114
p72-LDKE-1111	0.7315	0.6370	0.4859	0.7881	0.3634	0.8063
p72-LDKE-1121	0.7925	0.6288	0.4107	0.8469	0.3403	0.8617
p73-PKU_100	0.7838	0.6114	0.3630	0.8597	0.3293	0.8569
p73-PKU_101	0.7744	0.5701	0.2836	0.8566	0.2312	0.8518
p73-PKU_102	0.7482	0.6190	0.4139	0.8240	0.3148	0.8279
p73-PKU_105	0.7782	0.6174	0.3986	0.8362	0.3247	0.8481
p73-PKU_106	0.7900	0.6173	0.3741	0.8606	0.3271	0.8585
p73-PKU_107	0.7946	0.6148	0.3704	0.8593	0.2975	0.8616
p73-PKU_ICST_REF_11a	0.7634	0.6345	0.4321	0.8369	0.3837	0.8369
p73-PKU_ICST_REF_11b	0.7821	0.6083	0.3474	0.8693	0.3277	0.8561
p77-PKUSIGMASR01CLOUD	0.7982	0.6389	0.4008	0.8769	0.3717	0.8624
p77-PKUSIGMASR02CLOUD	0.7482	0.6007	0.3866	0.8148	0.3225	0.8151
p77-PKUSIGMASR03CLOUD	0.8163	0.6467	0.3882	0.9052	0.3666	0.8837
p77-PKUSIGMASR04CLOUD	0.7975	0.6278	0.3712	0.8845	0.3428	0.8618
p77-PKUSIGMASR05CLOUD	0.7915	0.6269	0.3799	0.8739	0.3632	0.8632
p77-PKUSIGMASR06CLOUD	0.7949	0.6075	0.3287	0.8863	0.3240	0.8654

different subset of the full set of 50 topics. Because of this, and because the difficulty level of the 50 topics varies substantially, the results presented here are not completely accurate. They do, however, give a reasonable idea of how well each run went, and which approaches work best. The final results will be released at a later date.

Table 2 gives the ranking for all of the runs. The run ID includes the ID number of the participating organisation; see Table 1 for the name of the organisation. The runs are ranked by geometric mean of recall and negative recall.

The highest ranked run, according to the preliminary results, is 'p20-QUTFirst300', in which the snippet consists of the first 300 characters of the underlying document.

Table 3 lists additional metrics for each run, as discussed in Section 2.5. One statistic worth noting is the fact that no run scored higher than 50% in recall, with an average of 35%. This indicates that poor snippets are causing users to miss over 50% of relevant results. Negative recall is high, with 94% of runs scoring over 80%, meaning that users are easily able to identify most irrelevant results based on snippets.

5 Conclusion

This paper gave an overview of the INEX 2011 Snippet Retrieval track. The goal of the track is to provide a common forum for the evaluation of the effectiveness of snippets. The paper has discussed the setup of the track, and presented the preliminary results of the track. The preliminary results indicate that in all submitted runs, poor snippets are causing users to miss over 50% of relevant results, indicating that there is still substantial work to be done in this area. The final results will be released at a later date, once all of the assessment has been completed.

References

1. Schenkel, R., Suchanek, F.M., Kasneci, G.: YAWN: A semantically annotated Wikipedia XML corpus. In: 12. GI-Fachtagung fr Datenbanksysteme in Business, Technologie und Web (BTW 2007), pp. 277–291 (2007)
2. Geva, S., Kamps, J., Lehtonen, M., Schenkel, R., Thom, J.A., Trotman, A.: Overview of the INEX 2009 ad hoc track. In: Geva, S., Kamps, J., Trotman, A. (eds.) Focused Retrieval and Evaluation. LNCS, pp. 4–25. Springer, Heidelberg (2010)

Focused Elements and Snippets

Carolyn J. Crouch, Donald B. Crouch,
Natasha Acquilla, Radhika Banhatti, Reena Narendavarapu

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 to produce competitive (i.e., highly ranked) focused elements in response to the various tasks of the INEX 2010 Ad Hoc Track. These experiments are based on an entirely new analysis and indexing of the INEX 2009 Wikipedia collection. Using this indexing and our basic methodology for dynamic element retrieval [1, 3], described herein, yields highly competitive results for all the tasks involved. These results are reported and compared to the top-ranked base runs with respect to significance. We also report on our current work in snippet production, which is still in very early stages. Our system is based on the Vector Space Model [5]; basic functions are performed using Smart [4].

In 2010, our INEX investigations centered on integrating our methodology for the dynamic retrieval of XML elements [1, 3] with traditional article retrieval to facilitate in particular the retrieval of *good focused elements*—i.e., elements which when evaluated are competitive with those in the top-ten highest ranked results. Earlier work [2] had convinced us that our approach was sound, but the scaling up of the document collection (i.e., moving from the small Wikipedia collection used in earlier INEX competitions to the new, much larger version made available in 2009) clarified an important point for us. The new (2009) Wiki documents are much more complex in structure than their predecessors. Because our methodology depends on being able to recreate the Wiki document at execution time, every tag (of the more than 30,000 possible tags within the document set) must be maintained during processing in order for the xpath of a retrieved element to be properly evaluated. (In earlier work, we had omitted some tags—e.g., those relating to the format rather than structure—for the sake of convenience, but this was no longer possible in the new environment.) Clearly, we needed to analyze the new collection with respect to the kinds of elements we wanted to retrieve (most specifically, the terminal nodes of the document tree). We spent some time on this process and then parsed and indexed the collection based on this analysis. All the experiments described herein are applied to this data.

In this paper, we describe our methodology for producing good focused elements for the INEX 2010 Ad Hoc tasks. The experiments performed using this approach are detailed and their results reported. To retrieve good focused elements in response to a

query, we use article retrieval (to identify the articles of interest) combined with dynamic element retrieval (to produce the elements) and then apply a focusing strategy to that element set. Experimental results confirm that this approach produces highly competitive results for all the specified tasks. We use the focused elements produced by this methodology as the basis for the snippets required by the INEX 2011 Ad Hoc Track. The results of these experiments to date are also reported.

References

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

[2] Crouch, C., Crouch, D., Vadlamudi, R., Cherukuri, R., Mahule, A.: A useful method for producing competitive ad hoc task results. S. Geva, J. Kamps, R. Schenkel, Trotman, A. (eds.), LNCS 6932, Springer, 63-70 (2011)

[3] 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

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

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

RMIT at INEX 2011 Snippet Retrieval Track

Lorena Leal, Falk Scholer, James Thom

RMIT University, Melbourne, Australia

Abstract. This report describes our participation in the Snippet retrieval track. Snippets were constructed by selecting sentences according to the occurrence of query terms. We followed a pseudo-relevance feedback approach in order to expand the original query. Preliminary results showed that a large number of extra terms may harm sentence selection for short summaries.

1 Introduction

In many IR systems the standard answer after submitting a query consists of a ranked list of results. Each one of these results is presented with three textual key elements: the title, the snippet and the URL. The retrieved documents are returned by the IR system because they a have a certain similarity with the users' query terms. However, not all documents in the answer list are likely to actually be relevant for a user. Therefore, users carry out a triage process, selecting which documents they wish to view in full by scanning these key elements. The snippet is generally the most indicative component when users need to review multiple documents for fulfilling their information needs. Snippets are short fragments extracted from the document, and their aim is to provide a glimpse of the document content. Common practices for constructing snippets include the selection of either metadata information, leading sentences of a document, or sentences containing query terms. Our approach focuses on the latter method.
In this regard, the INEX initiative launched the Snippet Retrieval Track to study not only system retrieval, but also snippet generation effectiveness. We describe the conducted experiments and results for this latter task.

2 Methodology

Given that we do not participate in the system retrival task, we used a baseline run distributed by the track organizers. It involved, for each topic, the first 500 Wikipedia articles retrieved by applying the BM25 similarity function ($K1 = 0.7, b = 0.3$). The snippet generation task consists of constructing succint summaries for those documents. Snippets were limited in terms of length to not exceed 300 characters.
In the following subsection, we briefly describe the collection and topics for the track. We have divided the conducted experiments in two parts: *query expasion* and a *snippet generation*.

2.1 Collection and Topics

Documents for the snippet track are part of the INEX Wikipedia collection. This collection is a snapshot of Wikipedia contructed in 2008 of English articles which are enriched with semantic annotations (obtained from YAGO). We did not use those annotations for query expansion or snippet creation experiments, so all markup was ignored from documents. We removed stopwords and applied the Porter stemming algorithm [4] to the remaining terms.

Each topic of the track includes the following fields: title, castitle, phrasetitle, description and narrative. For our experiments, we only used the title terms as they resemble information requests from typical real users. Stopping and stemming was also applied to these terms.

2.2 Query Expansion

Query expansion is a technique that attempts to address the potential vocabulary mismatch between users and authors. That is, users may choose different terms to describe their information needs than authors use when creating documents. Query expansion introduces new and possibly closely related terms to an original query, thus enlarging the set of results. This technique has been explored in terms of retrieval effectiveness of IR systems [2, 6, 9]. However, we employed query expansion to source extra terms for extractive summarisation approaches that we explain later.

We followed Rocchio's approach [5] for expanding the initial query.

$$\boldsymbol{Q_1} = \alpha \cdot \boldsymbol{Q_0} + \frac{\beta}{|R|} \sum_{d \in R} \boldsymbol{d} - \frac{\gamma}{|\overline{R}|} \sum_{d \in \overline{R}} \boldsymbol{d} \tag{1}$$

As can be seen in Equation 1, the influence of the original query ($\boldsymbol{Q_0}$), relevant (R) and irrelevant ($\overline{R}$) documents can be adjusted by the α, β, γ parameters, respectively. For our experiments the value of α was set with a low value to only choose terms that were different from the original query. The value of γ, in contrast, was set to 0 since we did not have negative feedback (or irrelevant documents) from the ranked list of results. Consequently, the value of β can be set to any non-zero value as this will affect the final result in a constant way. Thus, $\boldsymbol{Q_1}$ will contain a new set of terms.

Given that a reference run was provided, we applied Rocchio's formulation by using the top ranked documents (R) for each topic. Subsequently, we selected the leading E tokens as expansion terms. We called this set as Rocchio terms. Based on previous experiments, we fixed $R = 5$ and $E = \{25, 40\}$.

The original query was expanded by concatenating the additional Rocchio terms. However, some refinements were done to the supplementary terms. Numbers were discarded, except numbers of exactly four digits, since we assumed that those may refer to years.

2.3 Snippet Generation

It has been shown that the presence of query terms in short summaries positively influences the finding of relevant documents [7, 8]. If the snippet lacks the key words provided by users, they are generally less able to detect whether the underlying document is relevant or not.
Previous research on extractive summarisation has explored the selection of sentences for succint summaries [1, 3]. The advantage of employing sentences is that they convey simpler ideas. Following this approach, Wikipedia articles were segmented into sentences. Nevertheless, it was difficult to properly identify sentence boundaries due to having multiple contributors with different writing styles. The main exception was detected when a sentence included abbreviations such as *"Dr. Smith"*. We did not do an analysis of abbreviations to address this issue in detail.
We processed Wikipedia articles before constructing snippets. Specifically, information contained inside the <*title*> and <*bdy*> was used to narrow the document content. We suggest that snippets should include information of the document itself instead of sources pointing to other articles. Therefore, the *Reference* section was ignored in our summarisation approaches. With the remaining text, we scored sentences according to the occurrence of query terms in them by using Equation 2.

$$qb_i = \frac{(\text{number of unique query terms in sentence } i)^2}{\text{total number of query terms}} \tag{2}$$

For all snippets, the title of the document was concatenated to the first ranked sentences. It should be noted that snippets should not exceed 300 characters according to the track requirements. Top sentences were presented in snippets depending on document length. In case an article contained less than 3 sentences, the snippet algorithm only provided the first 100 characters of each sentence. In this setting we assumed that the document was very short and was unlikely to provide any relevant information; therefore the summary was reduced. On the contrary, the leading 150 characters (or less) of the first ranked sentence were displayed, when the document had more than 3 sentences. Subsequent highly scored sentences were cut up to 100 characters. This is done until the summary length is reached.
Submitted runs employed the query-biased approach for ranking document sentences in three different settings. The run labelled as "baseline" generated snippets given the title words of each topic, that is, no expansion was applied. The runs defined as "expanded-40" and "expanded-25" used $E = 40$ and $E = 25$ respectively, for applying the query-biased score. Using a newswire collection in a former study, we found that 40 extra terms were sufficient for enlarging the original query. Given the differences among collections, we also experimented with another E value by reducing it to 25.

3 Results

Preliminary results for our runs are shown in Table 1. Of our three submitted runs, the baseline that used no query expansion performed the

Baseline	0.5612
Expanded 25	0.5446
Expanded 40	0.5411
Top run	0.5925
Median run	0.5019

Table 1. Preliminary results.

best, as measured by the geometric mean of recall and negative recall (GM), the offical track measure. Adding more expansion terms hurt performance on this task. However, all runs were more effective than the median run, based on the GM measure.

References

1. Brandow, R., Mitze, K., Rau, L.F.: Automatic condensation of electronic publications by sentence selection. Information Processing & Management 31(5), 675–685 (1995)
2. Buckley, C., Salton, G., Allan, J., Singhal, A.: Automatic query expansion using smart: Trec 3. In: Overview of the Third Text REtrieval Conference (TREC-3). pp. 69–80 (1995)
3. Luhn, H.P.: The automatic creation of literature abstracts. IBM Journal of Research and Development 2(2), 159–165 (1958)
4. Porter, M.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
5. Rocchio, J.J.: Relevance feedback in information retrieval pp. 313–323 (1971)
6. Salton, G., Buckley, C.: Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science 41(4), 288–297 (1990)
7. Tombros, A., Sanderson, M.: Advantages of query biased summaries in information retrieval. In: Proceedings of the 21st annual international ACM SIGIR conference. pp. 2–10. ACM (1998)
8. White, R.W., Jose, J.M., Ruthven, I.: A task-oriented study on the influencing effects of query-biased summarisation in web searching. Information Processing & Management 39(5), 707–733 (2003)
9. Xu, J., Croft, W.B.: Query expansion using local and global document analysis. In: Proceedings of the 19th annual international ACM SIGIR conference. pp. 4–11. ACM (1996)

Topical Language Model for Snippet Retrieval

Rongmei Li and Theo van der Weide

Radboud University, Nijmegen, The Netherlands

Our previous study found that topical language model improves document ranking of ad-hoc retrieval. In this work, our attention is paid on snippets that are extracted and generated from the provided ranked list of documents.

In our experiments of the Snippet Retrieval Track, we hypothesize that the user recognizes certain combinations of terms in created snippets which are related to their information needs. We automatically extract snippets using terms as the minimal unit. Each term is weighted according to their relative occurrance in its article and in the entire Wikipedia. The top K scoring terms are chosen for inclusion in the snippet. The term-extraction based snippets are then represented differently to the user. One is a cluster of words that indicate the described topic. Another is a cluster of semi-sentences that contains the topic information while preserving some language structure.

Indian School of Mines,Dhanbad at INEX 2011 Snippet Retrieval Task

Sukomal Pal
Dept. of CSE, ISM, Dhanbad
sukomalpal@gmail.com

Preeti Tamrakar
Dept. of CSE, ISM, Dhanbad
tamrakarpreeti89@gmail.com

Abstract. This paper describes the work that we did at Indian School of Mines, Dhanbad towards Snippet retrieval for INEX 2011. We pre-processed the XML-ified Wikipedia collection first which was fed to a very simple Snippet Retrieval system that we developed. We submitted 3 runs to INEX 2011. our performance was moderate.

1 Introduction

The usual way of interacting with an IR system is to enter a specific information need expressed as a query. As a result, the system provides a ranked list of retrieved documents. For each of these retrieved documents, the user is typically provided by the system a title and a few sentences from each of these documents. These few sentences are called a "snippet" or "excerpt" of the document and they have the role of helping the user decide which of the retrieved documents are more likely to meet his/her information need. Ideally, it should be possible to make this decision without having to refer to the full document text.

SNIPPET RETRIEVAL(SR):- Retrieving the snippets from the whole document is known as "Snippet Retrieval". The goal of the snippet retrieval track is to determine how best to generate informative snippets for search results. Such snippets should provide sufficient information to allow the user to determine the relevance of each document, without needing to view the document itself. Snippet should be informative enough i.e. a short summary of the document.

In summaries we have two basic kinds of summaries :-

a) Static summary:- These are always the same regardless of the query. A static summary is generally comprised of either or both a subset of the document and metadata associated with the document. The simplest form of summary takes the first two sentences or 50 words of a document, or extracts particular zones of a document, such as the title and author.

b) Dynamic summary:- These are customized according to the user's information need as deduced from a query. Dynamic summaries display one or more "windows" on the document, aiming to present the pieces that have the most utility to the user in evaluating the document with respect to their information need. Dynamic summaries are generally regarded as greatly improving the usability of IR systems, but they present a complication for IR system design.

Though summary generation techniques are often used for snippet retrieval and the problems are seen and attacked from the same angles. However, the point to note that is that summaries are always coherent text, where snippet are not always. Also, snippets can be a set of meaningful phrases without being complete sentences.

SR in the context of INEX:- INEX has organized snippet retrieval task from 2002. In this task, INEX provides three pools, each pool contain 50 topics and document-run p35-97-ism-snippet-Baseline-Reference-run_01. Participants are asked to either use their own system to retrieve a ranked set of snippets. Otherwise, the participants can use the result of reference run provided by the organizer as a seed to their snippet retrieval system.

We took the second approach i.e. we found snippets from the set of documents returned by the reference run. We submitted 3 runs (*ism-baseline-snippet-inex-2011-reference-runX.xml, X =0, 1, 2).*

The paper is organized as follows. In the next section, we review the works done in this field. In section 3 we describe test data followed by our approach. We discuss results in section 4. Finally we conclude with scope of future work.

2 Related Work

The documents to be summarized are articles of the Wall Street Journal (WSJ) taken from the TREC collection (Text REtrieval Conferences) (Harman 1996). In order to decide which aspects of the articles would provide utility to generating a summary, their characteristic were examined in a small scale study. The methodology that was followed involved examining 50 randomly selected articles from the collection and attempting to extract conclusions about the distribution of important information within them. Their title, headings, leading paragraph, and their overall structural organization were studied. This sample collection was used for experimentation with various system parameters, in order to approximate the best settings for the summarization system. Although the sample of the documents was small, there was a strong uniformity in the characteristics of the sample that allowed for a generalization of the conclusions to the entire collection.

Tombros and Sanderson [5] presented the first in-depth study showing that properly selected query-dependent snippets are superior to query-independent summaries with respect to speed, precision, and recall with which users can judge the relevance of a hit without actually having to follow the link to the full document. In this work , we take the usefulness of query-dependent result snippets for granted.

3 Data

Test data introduced in snippet retrieval are:-

Corpus-wiki:- The INEX Snippet Retrieval Corpus is a part of the whole Wikipedia XML Collection. Uncompressed, the corpus size is about 50.7 GB containing images and other multimedia data.

Queries:- The topic-set contains 50 queries (11-60). Queries contained topic id, title, castitle, phrasetitle, description and narrative.

Document-run :- INEX-supplied document-run had 100 documents per query for 50 queries. Some documents were relevant to the query and some were irrelevant.

4 Approaches

a) xml-to-txt conversion:- Corpus-wiki contains XML collection. The XML files are converted into text files before snippet generation. For this, a C program is written in which the XML tags are removed from the XML document. In the conversion of XML to TXT, XML tags and white space(WS) are removed and put only TXT tags.

b) Snippet generation:- After converting XML data into TXT data, snippet is generated by extracting the first 300 characters from the text document and stored into buffer which was considered as our snippet for the document.

c) Submission:- As a very naive approach considered the first 300 characters as snippets. We inserted snippets within the INEX submission format containing participant-id, description, topic-id and snippet. The example is shown below.

<?xml version =”1.0”?>

<!DOCTYPE inex-snippet-submission SYSTEM”inex-snippet-submission.dtd”>

<inex-snippet-submission participant-id=”35” run-id=”ism-snippet-Baseline-Reference-run_02”>

<description>information about Nobel prize </description>

<topic topic-id=”2011011”>

<snippet rsv=”1564.00” doc-id=”21201”>

Title: List of Nobel laureates in Chemistry. References General " All Nobel Laureates in Chemistry". Nobelprize.org. Retrieved on 2008-10-06. " Nobel Prize winners by category (chemistry)". Encyclopædia Britannica. Retrieved on 2008-10-06. Specific " Alfred Nobel – The Man Behind the Nobel Pr </snippet>

<snippet rsv=”1562.00” doc-id=”52502”>

Title: List of Nobel laureates in Physics. References General " All Nobel Laureates in Physics". Nobelprize.org. Retrieved on 2008-10-08. " Nobel Prize winners by category (physics)". Encyclopædia Britannica. Retrieved on 2008-10-08. Specific " The Nobel Prize in Physics 1901". Nobelprize.org. Retr </snippet>.

4 Results

There is only preliminary result for this Snippet Retrieval track. They gave the result till 50 rank. And we have submitted total 3 runs. So we got 20, 28 and 39 ranks. The participant who got 1st rank, has scored 0.5925. Our result is shown below:-

Run-id	Score	Rank
p35-97-ism-snippet-Baseline-Reference-run_01	0.5135	20
p35-98-ism-snippet-Baseline-Reference-run_01	0.4925	28
p35-ism-snippet-Baseline-Reference-run_02	0.4708	39

5 Conclusion

This is our very naive approach. Our immediate task is to make the evaluation program run on our system, reproduce the results reported, analyze the score and try to better our performance with proper parameter-tuning using other approaches . It was our first attempt and our performance is not so satisfactory, but we can say, it was moderate.There is plenty of work to do, some of which will definitely be attempted and addressed in the coming days.

References :-

[1] http://www.inex.mmci.uni-saarland.de
[2] http://www.Xmlsoft.org
[3] S. Brin and L. Page. ***The anatomy of a large-scale hypertextual Web search engine.*** In WWW7, pages 107–117, 1998.
[4] Schenkel, R., Suchanek, F.M., Kasneci, G.: Yawn: A semantically annotated wikipedia xml corpus. In: BTW. (2007) 277-291
[5] Anastasios Tombros and Mark Sanderson. ***Advantages of query biased summaries in information retrieval.*** In 21st Conference on Research and Development in Information Retrieval (SIGIR'98), pages 2–10, 1998.
[6] Andrew Turpin, Yohannes Tsegay, David Hawking, and Hugh E. Williams. ***Fast generation of result snippets in web search***. In 30th Conference on Research and Development in Information Retrieval (SIGIR'07), pages 127–134, 2007.
[7] Jade Goldstein, Mark Kantrowitz, Vibhu O. Mittal, and Jaime G. Carbonell. Summarizing text documents: ***Sentence selection and evaluation metrics.*** In 22nd Conference on Research and Development in Information Retrieval (SIGIR'99), pages 121–128, 1999.
[8] Ryen White, Joemon M. Jose, and Ian Ruthven. ***A task-oriented study on the influencing effects of query-biased summarisation in web searching.*** Information
Processing Management, 39(5):707–733, 2003.
[9] Ryen White, Ian Ruthven, and Joemon M. Jose. ***Finding relevant documents using top ranking sentences***: an evaluation of two alternative schemes. In 25th Conference on Research and Development in Information Retrieval (SIGIR'02), pages 57–64, 2002.

PKU at INEX 2011 XML Snippet Track

Songlin Wang, Yihong Hong, and Jianwu Yang*

Institute of Computer Sci. & Tech., Peking University,
Beijing 100871, China
{wang_sl05, hongyihong, yangjw}@pku.edu.cn

Abstract. Snippets are used by almost every text search engine to complement ranking scheme in order to effectively handle user searches, which are inherently ambiguous and whose relevance semantics are difficult to assess. In this paper, we present our participation in the INEX 2011 Snippet track. A efficiently retrieval system has been used, by which the semi-structured of document has been considered, and present a snippet generate system, which effectively summarize the query results and document content, according to which users can quickly assess the relevance of the query results.

Keywords: XML Retrieval, XML-IR, Snippet, Query

1 Background

Each result in the results list delivered by current WWW search engines contains a short document snippet, and the snippet gives the user a sneak preview of the document contents. Accurate snippets allow users to make good decisions about which results are worth accessing and which can be ignored.

INEX 2011 XML Snippet track contains two parts, XML document retrieval and snippet generation task. XML retrieval presents different challenges than retrieval in text documents due to the semi-structured nature of the data, the goal is to take advantage of the structure of explicitly marked up documents to provide more focused retrieval results. The goal of the snippet generation is to generate informative snippets for search results; such snippets should provide sufficient information to allow the user to determine the relevance of each document, without needing to view the document itself.

The corpus is a subset of the Wikipedia corpus with 144,625 documents, and the snippet retrieval track will use the INEX Wikipedia collection. Topics will be recycled from previous ad hoc tracks. Participating organizations will submit a ranked list of documents, and corresponding snippets. Each submission should contain 500 snippets per topic, with a maximum of 300 characters per snippet.

* Corresponding author

This paper is organized as follows. In section 2, we show the retrieval system. Section 3 describes the approach taken to the snippet task. The paper ends with a discussion of future research and conclusion in section 4.

2 Retrieval System

In snippet track, we direct two parts, the document retrieval and the snippet generation. We first retrieve each element of articles using CO queries. The retrieval model is based on Okapi BM25 scores between the query model and the document (element) model that is smoothed using Dirichlet and Jelinek-Mercer priors.

2.1 Retrieval model

The Vector Space Model is the basic model of this research. The vector space model represents each document and query as an n-dimensional vector of unique terms. The terms are weighted based on their frequency within the document. The relationship of a document to a query is determined by the distance between the two vectors in vector space. The closer the two vectors, the more potentially relevant the document is (cosine similarity is one of the measures used to compute the distance).

Our retrieval engine is based on the Vector Space Model. Inside of use the vector space model represents each document as an n-dimensional vector of unique terms, we used a matrix to represent the document, and vector for each element of the document.

2.2 BM25 Score Formula

Our system is based on the BM25 weights function.

$$\mathrm{Score}(\mathrm{Q},\mathrm{e_i}) = \sum_{j=1}^{m} IDF(q_j) \times \frac{tf(q_j,\mathrm{e_i}) \times (k_1+1)}{tf(q_j,\mathrm{e_i}) + k_1 \times (1 - b + b \times \frac{dl}{avgdl})} \qquad 2\text{-}1$$

$$IDF(q_j) = \log \frac{\mathrm{N} - \mathrm{n}(q_j) + 0.5}{\mathrm{n}(q_j) + 0.5} \qquad 2\text{-}2$$

With:

- $tf(q_j,\mathrm{e_i})$: the frequency of keyword q_j in element e_i.
- N: the number of articles in the collection.
- $n(q)$: the number of articles containing the term q_j.
- $\frac{dl}{avgdl}$: the ratio between the length of element e_i and the average element length.
- k_1 and b: the classical BM25 parameters.

Parameter k_1 is able to control the term frequency saturation. Parameter b allows setting the importance of $\frac{dl}{avgdl}$.

2.3 Score for document(sub-tree)

Each document has lots of elements, and the formula for the document is defined as follow:

$$Score(d) = D(m) \cdot \sum_{i=1}^{m} \mathrm{Score}(\mathrm{Q}, \mathrm{e}_i) \cdot elementweight(Q, e_i) \quad 2\text{-}3$$

$$D(m) = \begin{cases} D_{single} & (m = 1) \\ D_{multiple} & (m > 1) \end{cases} \quad 2\text{-}4$$

With:

- $\mathrm{Score}(\mathrm{Q}, \mathrm{e}_i)$: the score of element e_i in document.
- $elementweight(Q, e_i)$: the weight of element e_i in the document.
- $D(m)$: the reduced via the element decay.

2.4 Weight of the element

We used the INEX2010 croup as a learning set, we split the elements into two part, relevant elements and irrelevant elements, using information gain, and we can get each element weight.

$$Gain(e) = -\sum_{i=1}^{2} P(c_i) \log P(c_i) \quad 2\text{-}5$$

$$= -[P(e)\{-\sum_{i=1}^{2} P(c_i \mid e) \log P(c_i \mid e)\}$$

$$+ P(\bar{e})\{-\sum_{i=1}^{2} P(c_i \mid \bar{e}) \log P(c_i \mid \bar{e})\}] \quad 2\text{-}6$$

With:

- c_i : relevant or irrelevant classification.
- $P(c_i|e)$: the probability of element e belongs to class c_i.

2.5 Pseudo Feedback

In most collections, the same concept may be referred to using different words. This issue, known as synonymy, has an impact on the recall of most information retrieval systems. In this track we use Pseudo Feedback method to expand the query of the topics.

First a query with the recognized phrase is submitted to the retrieval system, and the system will do the first run to rank document and pick the top ranked 50 documents. These top ranked documents are assumed to be relevant by the retrieval system and are combined with the original query through query expansion to do the second run. The retrieval system presents newly ranked documents to the user. And the score of the words that expanded form this method is defined as follow:

$$Score(w|Q) = \sum idf(w) idf(q_i) \log \frac{df}{|D|+1} \quad 2\text{-}7$$

$$idf(w) = \log \frac{\mathrm{N} - \mathrm{n}(w) + 0.5}{\mathrm{n}(w) + 0.5} \quad 2\text{-}8$$

With:

-|D|: the number of terms in document D.

-df : the frequency of term w in document D.

2.6 The Distribution of Keywords

The more of the different keywords, the passage will be more relevant, and so is the distance of the keywords. We used method of SLCA to get the smallest sub-tree that satisfies the retrieval (contain all keywords), and in this sub-tree we can calculate the score of query Q to the position x is defined by:

$$score(i, Q) = \sum_{k=1}^{|Q|} c(kw_k, j) \exp\left[\frac{-(\mathrm{i}-\mathrm{k})^2}{2\,\partial^2}\right] \quad 2\text{-}9$$

With:

- $c(kw_k, j)$: the value of keyword kw_k in the position j.

Some other case also be considered, one keywords in one element of the sub-tree and another is belong to its brother element, the keywords in the same element, and so on. For example,

```
<sec>
    <st> the mountains in Indian</ st>
    <p> …(list of mountains and their distribute)</p>
</sec>
```

We can make sure that the content in element <p> is the retriever need, so when one element contains keywords of the query, its brother node is also important. When general the snippet of the document, the weight of this kind element (<p> as above example) should be increased.

3 Snippet Generation

In the snippet generation system, we use query relevance, significant words, title/section-title relevance and tag weight to evaluate the relevance between sentences and a query. The sentences with higher relevance score will be chosen as the retrieval snippet.

3.1 Query Relevance

The relevance between a query and sentences largely depends on the query terms in a sentence. Function [3-1] is one example to calculate the relevance.

$$Score_{query}(s) = queryC(s) * \sum_{i=1}^{n} Occ(q_i, s) * Weight(q_i) \quad 3\text{-}1$$

$$Weight(q_i) = idf(q_i) \quad 3\text{-}2$$

With:

- $queryC(s)$: the number of query terms category in sentence s.
- $Occ(q_i, s)$: the occurrence frequency of query terms $q_i$in sentence s.
- $Weight(q_i)$: the weight of query term q_i.
- $idf(q_i)$: the IDF value of term $q_i$in the database.

3.2 Significant Words[3]

The frequency of significant words was used to help evaluate the relevance between a query and a sentence. A word is defined as a significant word if it is a non-stop word and its term frequency is larger than a threshold T. T is defined as function [3-3].

$$T = 7 + I * 0.1 * |L - n| \quad 3\text{-}3$$

With:

- n: the number of sentences in the document.
- L: L is 25 for n < 25 and 40 for n > 40.
- I: I is 0 for 25<=n<=40 and 1 otherwise

The significant words score is based on the function [3-4]

$$Score_{sw}(s) = SW/TSW \quad 3\text{-}4$$

Where SW is the number of significant words in the sentence and TSW is the total number of words in the sentence.

3.3 Title Relevance

Since the title is the best summary of a document, the sentence with more title terms is highly probably relevant to the query. The title score of the sentence can be calculated using the function [3-5].

$$Score_{title}(s) = T/N \quad 3\text{-}5$$

Where T is the number of title words in the sentence s and N is the number of title words.

3.4 Sentence Score

The function [3-6] is the formula used to calculate the sentence score.

$$Score_{rel}(s) = \alpha Score_{query}(s) + \beta Score_{sw}(s) + \gamma Score_{title}(s) \quad 3\text{-}6$$

Where$\alpha = 0.7$, $\beta = 0.15$, $\gamma = 0.15$ based on the experiment.

Additionally, we use a section-title weight and a tag-weight calculated in the document retrieval system to help evaluate the sentence relevance. If a sentence is in a section whose title contains the query words, the sentence may be more relevant to the query. The function [3-7] synthesizes all the factors mentioned above, and generates the final sentence relevance score.

$$Score(s) = \big(1 + \mu * stW(s) + \sigma * tagW(s)\big) * Score_{rel}(s) \quad 3\text{-}7$$

With

-stW(s): The frequency of query terms in the section-title of the section containing sentence s.

-tagW(s): The tag weight of the tag containing the sentence s.

4 Conclusions

From the special focus on exploiting structural characteristics of XML document collections, we retrieve XML documents based on both document structure and content. We have learned the weight of elements based on the cast of INEX2010 to enhance the retrieval performance, and we also consider the distribution of the keywords in the documents and elements. In the snippet generation system, we use query relevance, significant words, title/section-title relevance and tag weight to evaluate the relevance between sentences and a query. The sentences with higher relevance score will be chosen as the retrieval snippet.

5 Acknowledgment

The work reported in this paper was supported by the National Natural science Foundation of China Grant 60642001 and 60875033.

References

1. Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai, editors. Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005, Revised Selected Papers, volume 3977 of Lecture Notes in Computer Science. Springer, 2006.
2. Wouter Weerkamp. Optimizing structured document retrieval using focused and relevant in context strategies. Master's thesis, Utrecht University, the Netherlands, 2006.
3. Changhu Wang, Feng Jing, Lei Zhang, Hong-Jiang Zhang. Learning Query-Biased Web Page Summarization. Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, 2007.
4. Jade Goldsteiny, Mark Kantrowitz, Vibhu Mittal, Jaime Carbonelly. Summarizing Text Documents: Sentence Selection and Evaluation Metrics. Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, 1999.
5. Wesley T. Chuang, Jihoon Yang. Extracting Sentence Segments for Text Summarization: A Machine Learning Approach. SIGIR '00 Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval.

Author Index

ISBN 978-90-814485-8-1